AIX เวอร์ชัน 7.2

Performance management

AIX เวอร์ชัน 7.2

Performance management

หมายเหตุ

ก่อนใช้ข้อมูลนี้ รวมถึงผลิตภัณฑ์ที่สนับสนุน โปรดอ่าน ข้อมูลใน “คำประกาศ” ในหน้า 511

This edition applies to AIX Version 7.2 and to all subsequent releases and modifications until otherwise indicated in new editions.

สารบัญ

เกี่ยวกับเอกสารนี้

คอลเล็กชันหัวข้อนี้ให้ข้อมูลแบบครบถ้วนแก่แอ็พพลิเคชันโปรแกรมเมอร์ วิศวกร ของลูกค้า วิศวกรระบบ ผู้ดูแลระบบ ผู้ใช้ที่มี ประสบการณ์ และโปรแกรมเมอร์ระบบ เกี่ยวกับวิธี ดำเนินงานในการประเมิน และการปรับแต่งประสิทธิภาพการทำงานของตัวประมวลผล, ระบบไฟล์, หน่วยความจำ, I/O ดิสก์, Network File System (NFS), Java และ I/O การสื่อสาร คอลเล็กชันหัวข้อนี้ยังอธิบายการออกแบบ ระบบและแอ็พพลิเคชันอย่างมีประสิทธิภาพ รวมถึงการประยุกต์ใช้ คอลเล็กชัน หัวข้อนี้ยังมีอยู่บน CD เอกสารคู่มือที่ รวมมากับระบบปฏิบัติการ

การไฮไลต์

มีการใช้ระเบียบการไฮไลต์ต่อไปนี้ในเอกสารนี้:

ตัวหนา	ระบุคำสั่ง รูทีนย่อย คีย์เวิร์ด ไฟล์ โครงสร้าง ไดเร็กทอรี และไอเท็มอื่นที่เป็นเจ้าของชื่อ ที่กำหนดไว้ล่วงหน้าโดยระบบ และยังระบุอ็อบเจ็กต์รูปภาพ เช่น ปุ่ม เลเบล และไอคอนที่ผู้ใช้เลือก
ตัวเอน	ระบุพารามิเตอร์ซึ่งผู้ใช้จะเป็นผู้ระบุชื่อจริง หรือค่า
`Monospace`	ระบุตัวอย่างของค่าข้อมูลเฉพาะ ตัวอย่างของข้อความที่คล้ายกับที่คุณจะเห็นแสดงขึ้น ตัวอย่างของส่วนของโปรแกรม โค้ดที่คล้ายกับที่คุณอาจบันทึกในฐานะโปรแกรมเมอร์ ข้อความจากระบบ หรือข้อมูลที่คุณควรจะพิมพ์จริง

การตรงตามตัวพิมพ์ใน AIX

ทุกสิ่งในระบบปฏิบัติการ AIX® เป็นแบบตรงตาม ตัวพิมพ์ ซึ่งหมายความว่ามีการแยกแยะความแตกต่างระหว่างตัวอักษรพิมพ์ใหญ่ และพิมพ์เล็ก ตัวอย่างเช่น คุณสามารถใช้คำสั่ง **ls** เพื่อ แสดงรายชื่อไฟล์ ถ้าคุณพิมพ์ `LS` ระบบจะตอบกลับว่า `is not found` คำสั่ง ในลักษณะคล้ายกัน **FILEA**, **FiLea**, และ **filea** คือชื่อไฟล์ที่แตกต่างกันสามไฟล์ แม้ว่า จะอยู่ในไดเร็กทอรีเดียวกันก็ตาม เพื่อหลีกเลี่ยงการทำการดำเนินการที่ไม่ต้องการ ตรวจสอบให้แน่ใจว่าคุณใช้ตัวพิมพ์ที่ถูกต้องเสมอ

ISO 9000

ISO 9000 registered quality systems were used in the development and manufacturing of this product.

การจัดการประสิทธิภาพ

ส่วนนี้แสดงข้อมูลที่สมบูรณ์ให้แอ็พพลิเคชันโปรแกรมเมอร์ วิศวกรลูกค้า วิศวกรระบบ ผู้ดูแลระบบ ผู้ใช้ขั้นปลายที่มีประสบการณ์ และโปรแกรมเมอร์ระบบทราบเกี่ยวกับวิธีการทำภารกิจ เช่น การประเมินและการปรับประสิทธิภาพของตัวประมวลผล ระบบไฟล์ หน่วยความจำ ดิสก์ I/O, NFS, Java และ I/O การสื่อสาร หัวข้อนี้ยังระบุการออกแบบระบบและแอ็พพลิเคชันที่มีประสิทธิภาพ รวมถึง การนำการออกแบบนั้นไปใช้งาน หัวข้อนี้ยังมีอยู่ในแผ่นซีดีส่วนเอกสาร ที่จัดส่งพร้อมกับระบบปฏิบัติการด้วย

หมายเหตุ: เมทริกที่รายงานโดยเครื่องมือสถิติ เช่น **lparstat, vmstat, iostat** และ **mpstat** รวมถึงแอ็พพลิเคชันบน Perfstat application program interface (API) หรือ system performance measurement interface (SPMI) API varies ในบางระดับเมื่อรัน แบบขนานด้วยช่วงเวลาสุ่มตัวอย่างเดียวกันที่อินสแตนซ์ของเวลา ที่กำหนด

มีอะไรใหม่ในการจัดการประสิทธิภาพ

อ่านเกี่ยวกับข้อมูลใหม่หรือข้อมูลที่เปลี่ยนแปลงที่เห็นได้ชัดเจนสำหรับชุดของการจัดการประสิทธิภาพ

วิธีการดู มีอะไรใหม่หรือมีอะไรที่เปลี่ยนแปลง

ในไฟล์ PDF คุณอาจดูแถบการปรับปรุงใหม่ (|) ในขอบด้านซ้าย ของข้อมูลใหม่หรือข้อมูลที่เปลี่ยนแปลง

October 2016

ข้อมูลต่อไปนี้เป็นข้อมูลสรุปของการอัพเดตที่มีในคอลเล็กชันหัวข้อ การปรับแต่งประสิทธิภาพ:

- เพิ่มข้อมูลเกี่ยวกับหัวข้อ “VMM thread interrupt offload” ในหน้า 194

มกราคม 2016

หัวข้อต่อไปนี้ได้รับการอัพเดต:

- “การจัดสรรพื้นที่เพจที่เลื่อนออกไป” ในหน้า 170
- “ขีดจำกัดระบบไฟล์ JFS ขั้นสูง Maxclient” ในหน้า 170
- “พูลหน่วยความจำ” ในหน้า 168
- “LRU แบบอิงรายการ” ในหน้า 169
- “การจัดสรรพื้นที่ว่างหน้า” ในหน้า 170
- “การเปลี่ยนคำสั่ง vmtune และ schedtune” ในหน้า 488
- “การปรับการเปลี่ยนหน้า VMM” ในหน้า 166
- “ค่าสำหรับพารามิเตอร์ minperm และ maxperm” ในหน้า 169
- “ค่าสำหรับพารามิเตอร์ minfree และ maxfree” ในหน้า 167
- “การปรับการเปลี่ยนหน้า VMM” ในหน้า 166

December 2015

ข้อมูลต่อไปนี้เป็นข้อมูลสรุปของอัพเดตที่ทำไว้กับคอลเล็กชันหัวข้อ Performance Management

- เพิ่มพารามิเตอร์ที่สามารถปรับได้ไปยังหัวข้อ AIX_STDBUFSIZ พารามิเตอร์ที่ปรับค่าได้อื่นๆ
- เพิ่มระยะเวลาการซิงโครไนซ์ JFS2 และส่วนที่ทำงานพร้อมกันไปยังหัวข้อ การปรับประสิทธิภาพการทำงาน การซิงโครไนซ์ไฟล์ ข้อมูลนี้อธิบายถึงการใช้งานของความสามารถในการปรับค่าเพื่ออนุญาตให้ระบบไฟล์ JFS2 เพื่อจัดการกับการซิงโครไนซ์โดยไม่ได้ใช้ sync daemon
- เพิ่มลิงก์ข้อมูลที่เกี่ยวข้องให้กับคำสั่ง **caccelstat** ในหัวข้อ คำสั่งการปรับประสิทธิภาพการทำงาน
- อัพเดตข้อมูลเกี่ยวกับตัวแปรสภาวะแวดล้อม AIXTHREAD_SCOPEt ในหัวข้อ "การปรับ Java สำหรับ AIX" ในหน้า 430 และ "การปรับเธรด" ในหน้า 76 ค่าดีฟอลต์ของ ตัวแปรสภาวะแวดล้อมนี้ คือ **S**

พื้นฐานของประสิทธิภาพ

การประเมินประสิทธิภาพของระบบจำเป็นต้องทำความเข้าใจถึง ไดนามิกของการดำเนินการโปรแกรม

เวิร์กโหลดระบบ

คำนิยามที่ถูกต้องและสมบูรณ์ของเวิร์กโหลดของระบบมีความสำคัญมากต่อ การคาดการณ์หรือการทำความเข้าใจกับประสิทธิภาพของระบบ

ความแตกต่างในเวิร์กโหลดอาจก่อให้เกิดความผันแปรใน ประสิทธิภาพที่ประเมินของระบบได้มากกว่าความแตกต่างในความเร็วของนาฬิกา CPU หรือขนาด random access memory (RAM) คำนิยามเวิร์กโหลดต้องรวมไม่เฉพาะแต่ชนิดและอัตราของการร้องขอ ที่ส่งไปยังระบบเท่านั้น แต่ยังต้องมีซอฟต์แวร์แพ็กเกจที่ถูกต้อง และแอ็พพลิเคชันโปรแกรมในบ้านที่จะดำเนินการด้วย

สิ่งสำคัญคือการรวมงานที่ระบบกำลังทำอยู่ในพื้นหลัง ตัวอย่างเช่น ถ้าระบบมีระบบไฟล์ที่มีการติดตั้ง NFS และเข้าถึงบ่อยครั้งโดย ระบบอื่น การจัดการกับการเข้าถึงดังกล่าวอาจเป็นส่วนประกอบสำคัญของ เวิร์กโหลดโดยรวม แม้ว่าระบบไม่ใช่เซิร์ฟเวอร์ที่เป็น ทางการ

เวิร์กโหลดที่มีการกำหนดมาตรฐานให้สามารถทำการเปรียบเทียบระหว่างระบบ ที่ไม่คล้ายกันได้เรียกว่า *เกณฑ์เปรียบเทียบ* อย่างไรก็ตาม เวิร์กโหลดจริงบางรายการทำซ้ำ ขั้นตอนวิธีและสภาพแวดล้อมที่ถูกต้องของเกณฑ์เปรียบเทียบ แม้แต่เกณฑ์เปรียบเทียบมาตรฐาน อุตสาหกรรมที่สร้างขึ้นมาจากแอ็พพลิเคชันจริง มีการปรับให้ง่ายขึ้น และมีลักษณะเหมือนกันเพื่อให้สามารถนำไปใช้กับฮาร์ดแวร์แพล็ตฟอร์มได้หลายแบบ การใช้งานที่ถูกต้องเพียงอย่างเดียวสำหรับเกณฑ์เปรียบเทียบมาตรฐานอุตสาหกรรม คือลดจำนวนของระบบที่ต้องมีการประเมินโดยละเอียด ด้วยเหตุนี้ คุณจึงไม่ควรอาศัยผลของเกณฑ์เปรียบเทียบเพียงอย่างเดียวเมื่อพยายามทำความเข้า ใจกับเวิร์กโหลดและประสิทธิภาพของระบบของคุณ

เวิร์กโหลดสามารถจัดประเภทเป็นหมวดหมู่ต่อไปนี้:

หลายผู้ใช้

เวิร์กโหลดที่ประกอบด้วยผู้ใช้จำนวนมากที่ส่งงานผ่านทาง เทอร์มินัลแต่ละเครื่อง โดยปกติ วัตถุประสงค์ด้านประสิทธิภาพของเวิร์กโหลดชนิดนี้คือ การเพิ่มผลผลิตของระบบให้สูงสุดในขณะที่รักษาเวลาตอบกลับในกรณีแย่ที่สุดซึ่ง ระบุ หรือการได้รับเวลาการตอบกลับที่ดีที่สุดสำหรับเวิร์กโหลด คงที่

เซิร์ฟเวอร์

เวิร์กโหลดที่ประกอบด้วยการร้องขอจากระบบอื่น ตัวอย่างเช่น ไฟล์เซิร์ฟเวอร์เวิร์กโหลดประกอบด้วยการร้องขอการอ่านดิสก์และการบันทึกดิสก์เป็นส่วนใหญ่ เป็นดิสก์ I/O คอมโพเนนต์ของเวิร์กโหลดแบบหลายผู้ใช้ (บวก NFS หรือกิจกรรม I/O อื่น) ดังนั้นจึงมีวัตถุประสงค์เหมือนกันคือผลผลิตสูงสุดภายในขีดจำกัดเวลาการตอบกลับ ที่กำหนด เซิร์ฟเวอร์เวิร์กโหลดอื่นประกอบด้วยไอเท็มต่างๆ เช่น โปรแกรมเน้น คณิตศาสตร์ ธุรกรรมฐานข้อมูล งานของเครื่องพิมพ์

เวิร์กสเตชัน

เวิร์กโหลดที่ประกอบด้วยผู้ใช้รายเดียวที่ส่งงานผ่านทางคีย์บอร์ด และรับผลลัพธ์บนจอแสดงผลของระบบนั้น โดยปกติ วัตถุประสงค์ด้านประสิทธิภาพ ที่สำคัญมากที่สุดของเวิร์กโหลดชนิดนี้คือ การลดเวลาตอบกลับสำหรับการร้องขอ ของผู้ใช้ให้เหลือน้อยที่สุด

วัตถุประสงค์ของประสิทธิภาพ

หลังจากกำหนดเวิร์กโหลดที่ระบบของคุณจะต้องประมวลผลแล้ว คุณสามารถเลือกเกณฑ์ประสิทธิภาพและกำหนดวัตถุประสงค์ของประสิทธิภาพตาม ข้อมูลเกณฑ์เหล่านั้นได้

เกณฑ์ประสิทธิภาพโดยรวมของระบบคอมพิวเตอร์คือเวลาตอบกลับ และผลผลิต

เวลาตอบกลับ คือเวลาที่ผ่านไปตั้งแต่ส่งการร้องขอ จนกระทั่งมีการส่งคืนการตอบกลับจากการร้องขอนั้น ตัวอย่างรวมถึง:

- จำนวนเวลาที่เคียวรีฐานข้อมูลใช้
- จำนวนเวลาที่ใช้ในการกระจายอักขระไปยังเทอร์มินัล
- จำนวนเวลาที่ใช้ในการเข้าถึงเว็บเพจ

ผลผลิต คือการประเมินจำนวนงานที่สามารถทำได้ ในบางหน่วยของเวลา ตัวอย่างรวมถึง:

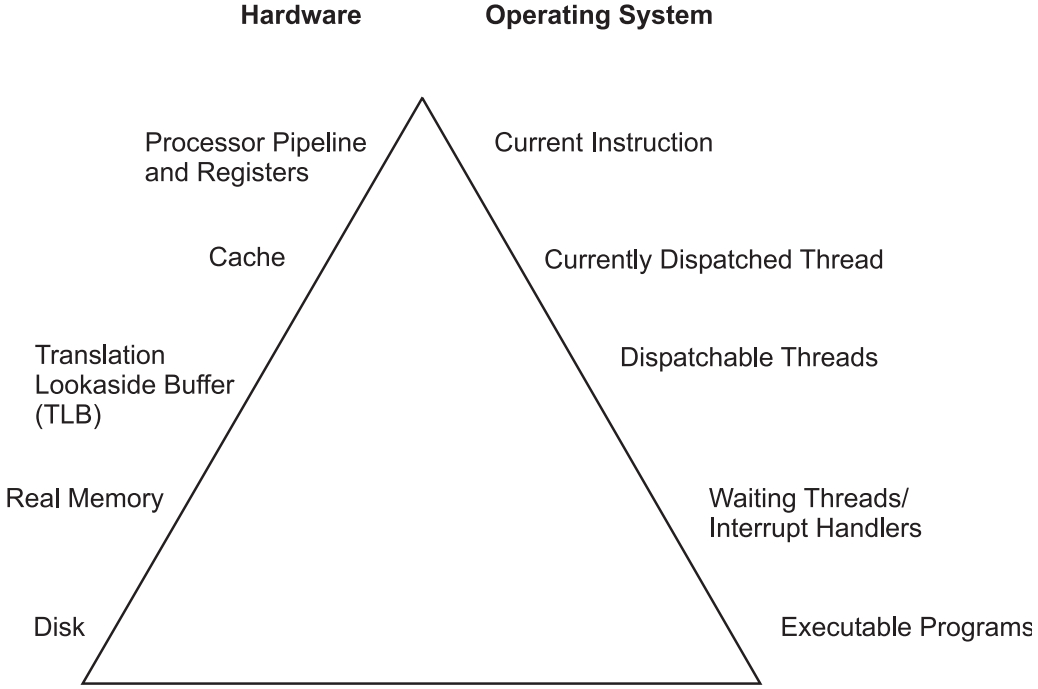
- ธุรกรรมฐานข้อมูลต่อนาที
- กิโลไบต์ของไฟล์ที่โอนย้ายต่อวินาที
- กิโลไบต์ของไฟล์ที่อ่านหรือบันทึกต่อวินาที
- Web server hits ต่อนาที

ความสัมพันธ์ระหว่างเมตริกซ์เหล่านี้มีความซับซ้อน ในบางครั้ง คุณสามารถมีผลผลิต ที่สูงขึ้นในเวลาการตอบกลับที่นานขึ้น หรือเวลาการตอบกลับที่ดีขึ้นในขณะที่ ผลผลิตแย่ลง ในสถานการณ์อื่น การเปลี่ยนแปลงอย่างหนึ่งสามารถพัฒนาทั้งสองอย่างได้ ประสิทธิภาพที่ยอมรับได้ขึ้นอยู่กับผลผลิตที่สมเหตุสมผลรวมกับเวลาการตอบกลับ ที่สมเหตุสมผล

ในการวางแผนหรือการปรับระบบใดๆ ตรวจสอบให้แน่ใจว่าคุณมีวัตถุประสงค์ที่ ชัดเจนสำหรับทั้งเวลาการตอบกลับและผลผลิต ในขณะที่ประมวลผลเวิร์กโหลดที่ระบุ มิฉะนั้น คุณอาจต้องเสียเวลาวิเคราะห์และเสียเงินเพื่อปรับปรุง ประสิทธิภาพระบบในส่วนที่ไม่สมบูรณ์เพียงพอ

โมเดลการดำเนินการโปรแกรม

เพื่อตรวจสอบลักษณะประสิทธิภาพของเวิร์กโหลดอย่างชัดเจน โมเดลแบบไดนามิกแทนที่จะเป็นโมเดลแบบสแตติกของการดำเนินการโปรแกรม จึงเป็นสิ่งจำเป็น ดังแสดงในรูปภาพต่อไปนี้



รูปที่ 1. ลำดับชั้นการดำเนินการโปรแกรม. รูปภาพเป็นรูปสามเหลี่ยม ซึ่งเป็นรูปพื้นฐานของลำดับชั้น ด้านซ้ายแสดงถึงฮาร์ดแวร์เอนทิตีที่ตรงกับเอนทิตีระบบปฏิบัติการที่เหมาะสมบน ด้านขวา โปรแกรมต้องเริ่มต้นจากระดับต่ำสุดของการจัดเก็บ บนดิสก์ไปยังระดับสูงสุดของตัวประมวลผลที่กำลังรันคำสั่ง โปรแกรม ตัวอย่างเช่น จากล่างสุดไปบนสุด ดิสก์ฮาร์ดแวร์เอนทิตีมีโปรแกรม ที่สามารถดำเนินการได้ หน่วยความจำจริงที่มีเธรดระบบปฏิบัติการที่รออยู่ และตัวจัดการขัดจังหวะ บัฟเฟอร์การแปลมีเธรดที่จัดส่งได้ แคชมีเธรดที่จัดส่งในปัจจุบันและไปป์ไลน์ตัวประมวลผล และทะเบียนมีคำสั่ง ปัจจุบัน

เพื่อรัน โปรแกรมต้องพัฒนาขึ้นทั้งลำดับชั้นฮาร์ดแวร์และระบบปฏิบัติการ คู่ขนานกันไป แต่ละองค์ประกอบในลำดับชั้นฮาร์ดแวร์เป็นสิ่งที่หายากกว่า และมีราคาแพงกว่าองค์ประกอบด้านล่าง โปรแกรม ไม่เพียงแต่ต้องช่วงชิงรีซอร์สแต่ละรายการกับโปรแกรมอื่นเท่านั้น การแปลงจากระดับหนึ่งไปยังระดับถัดไปยังต้องใช้เวลานาน เพื่อให้เข้าใจ ไดนามิกของการดำเนินการโปรแกรม คุณต้องมีความเข้าใจพื้นฐานเกี่ยวกับ แต่ละระดับในลำดับชั้น

ลำดับชั้นของฮาร์ดแวร์

โดยปกติ เวลาที่จำเป็นในการย้ายจากระดับของฮาร์ดแวร์หนึ่งไปยังอีกระดับหนึ่ง ประกอบด้วยเวลาแฝงที่เป็นส่วนสำคัญของระดับที่ต่ำกว่า (เวลาจากการออกคำร้องขอ จนกกระทั่งได้รับข้อมูลในครั้งแรก)

ฮาร์ดดิสก์

การดำเนินการที่ช้าที่สุดสำหรับการรันโปรแกรม บนระบบสแตนด์อะโลนคือ การขอรับโค้ดหรือข้อมูลจากดิสก์ สำหรับเหตุผลดังต่อไปนี้:

- การควบคุมดิสก์ต้องไปที่การเข้าถึงบล็อกที่ระบุโดยตรง (การจัดลำดับคิวที่หน่วงเวลา)
- แขนของดิสก์ต้องค้นหาไซลินเดอร์ที่ถูกต้อง (ค้นหาเวลาแฝง)
- หัวอ่าน/เขียนต้องรอจนกว่าบล็อกที่ถูกต้องจะหมุนอยู่ภายใต้หัวเขียน/อ่าน (เวลาแฝงในการหมุน)
- ข้อมูลต้องถูกส่งไปยังการควบคุม (เวลาที่ส่งผ่าน) และส่งไปยังแอ็พพลิเคชันโปรแกรม (เวลา interrupt-handling)

การดำเนินการกับดิสก์ที่ช้าอาจมีสาเหตุเกี่ยวกับคำร้องขอการอ่านหรือการเขียน ในโปรแกรม กิจกรรมการปรับระบบจะพิสูจน์การค้นหา ดิสก์ I/O ที่ไม่จำเป็น

หน่วยความจำที่ใช้จริง

หน่วยความจำที่ใช้จริง อ้างถึง Random Access Memory หรือ RAM ซึ่งจะเร็วกว่าดิสก์ แต่แพงกว่าเมื่อเปรียบเทียบในหน่วยไบต์ ระบบปฏิบัติการจะเก็บอยู่ใน RAM เท่านั้น โค้ดและข้อมูลที่ใช้อยู่ การเก็บสิ่งเกินลงบนดิสก์ หรือการนำกลับมาไว้ใน RAM ในการวางในครั้งแรก

RAM ไม่ได้เร็วกว่า ตัวประมวลผล โดยปกติแล้ว เวลาแฝงของ RAM ของวงรอบตัวประมวลผล จะเกิดขึ้นระหว่างเวลาที่ฮาร์ดแวร์จดจำความต้องการเข้าถึง RAM และเวลาที่ข้อมูลหรือคำสั่งพร้อมใช้งานกับตัวประมวลผล

ถ้าการเข้าถึง ไปยังเพจของหน่วยความจำเสมือนที่เก็บอยู่บนดิสก์ หรือไม่ได้เก็บไว้ ข้อบกพร่องของเพจจะเกิดขึ้น และการประมวลผลของโปรแกรมจะหยุดทำงานชั่วคราวจนกว่าเพจจะถูกอ่านจากดิสก์

Translation Lookaside Buffer (TLB)

โปรแกรมเมอร์ จะถูกครอบด้วยข้อจำกัดเกี่ยวกับฟิสิคัลของระบบด้วยการนำหน่วยความจำเสมือน ไปใช้งาน คุณออกแบบและโค้ดโปรแกรมผ่านหน่วยความจำที่มีขนาดใหญ่ และระบบจะรับผิดชอบต่อการแปลแอดเดรสเสมือนของโปรแกรมสำหรับคำสั่งและข้อมูลลงในแอดเดรสจริง ที่ต้องการรับคำสั่งและข้อมูลจาก RAM เนื่องจากการประมวลผลการแปลแอดเดรส อาจใช้เวลาค่อนข้างนาน ระบบจะเก็บแอดเดรสจริงของการเข้าถึงเพจหน่วยความจำเสมือน ไว้ในแคชที่เรียกว่า translation lookaside buffer (TLB)

ตราบเท่าที่โปรแกรมยังคงรันอยู่ โปรแกรมจะดำเนินการเข้าถึงชุดของโปรแกรม และเพจข้อมูลที่มีขนาดเล็ก การแปล virtual-to-real page-address จะไม่ต้องการให้ทำซ้ำสำหรับการเข้าถึง RAM แต่ละตัว เมื่อโปรแกรมพยายามเข้าถึงเพจของหน่วยความจำเสมือน ที่ไม่มีรายการ TLB ที่เรียกว่า *TLB miss* วงรอบตัวประมวลผลจำนวนมากที่เรียกว่า *TLB-miss latency* จะมีความต้องการดำเนินการกับการแปลแอดเดรส

แคช

หากต้องการลดจำนวนเวลาของโปรแกรม ที่มีประสบการณ์กับเวลาแฝงของ RAM ระบบจะทำงานกับแคชสำหรับคำสั่ง และข้อมูล ถ้าคำสั่งที่ต้องการหรือข้อมูลที่ต้องการมีอยู่ในแคช แคชจะฮิตผลลัพธ์ และคำสั่งหรือข้อมูลจะพร้อมใช้งานกับตัวประมวลผลบนวงรอบถัดไป โดยไม่มีการหน่วงเวลา หรือ การหายไปของแคชจะเกิดขึ้นกับเวลาแฝงของ RAM

ในบางระบบ มีสองถึงสามระดับของแคช ซึ่งโดยปกติจะเรียกว่า L1, L2 และ L3 ถ้าหน่วยเก็บที่อ้างอิงโดยเฉพาะส่งผลลัพธ์ว่า L1 หายไป L2 จะถูกตรวจสอบ ถ้า L2 สร้างการหายไป การอ้างอิงจะไปยังระดับถัดไป นั่นคือ L3 ถ้ายังคงอยู่ หรือ RAM

ขนาดของแคชและโครงสร้างจะผันแปรตามแบบจำลอง แต่หลักการของการใช้แคชจะไม่ซ้ำกัน

ไพพ์ไลน์และเรจิสเตอร์

ไพพ์ไลน์ สถาปัตยกรรมมาตรวัดพิเศษ สร้างความเป็นไปได้ ภายใต้สถานการณ์บางอย่าง การประมวลผลพร้อมเพียงกันของคำสั่งจำนวนมาก ชุดของเรจิสเตอร์อเนกประสงค์ขนาดใหญ่ และเรจิสเตอร์อิงดัชนีจะเก็บจำนวนข้อมูลของโปรแกรมในเรจิสเตอร์ แทนการเก็บและโหลดข้อมูลใหม่อีกครั้ง

คอมไพลเลอร์ที่ออปติไมซ์จะถูกออกแบบมาเพื่อใช้ผลประโยชน์สูงสุดของความสามารถเหล่านี้ การ optimization ฟังก์ชันของคอมไพลเลอร์ควรถูกใช้ เมื่อสร้างโปรแกรมบนระบบที่ใช้จริง อย่างไรก็ตามโปรแกรมที่มีขนาดเล็กก็ถูกกระทำเช่นกัน *คู่มือการใช้ประโยชน์และการปรับสำหรับ XL Fortran, XL C และ XL C++* อธิบายถึงโปรแกรมที่สามารถปรับได้สำหรับผลการทำงานสูงสุด

ลำดับชั้นซอฟต์แวร์

เพื่อรัน โปรแกรมยังต้องคืบหน้าผ่านชุดของขั้นตอนใน ลำดับชั้นซอฟต์แวร์

โปรแกรมเรียกทำงาน

เมื่อคุณร้องขอโปรแกรมให้รัน ระบบปฏิบัติการจะดำเนินการกับ จำนวนของการดำเนินการเพื่อแปลงสภาพโปรแกรมเรียกทำงานบนดิสก์ ไปเป็นโปรแกรมที่รันอยู่

อันดับแรก ไดเร็กทอรีในตัวแปรสภาวะแวดล้อม *PATH* ปัจจุบันของคุณต้องถูกสแกนเพื่อค้นหาสำเนาที่ถูกต้องของโปรแกรม จากนั้น โหลดเดอร์ของระบบ (ไม่ใช่คำสั่ง **ld** ซึ่งเป็นตัวโยง) ต้องแก้ปัญหาเกี่ยวกับการอ้างอิงภายนอกใดๆ จากโปรแกรมกับไลบรารีที่แบ่งใช้

ในการแสดงคำร้องของคุณ ระบบปฏิบัติการจะสร้างการประมวลผล หรือชุดของรีซอร์ส เช่น เซ็กเมนต์แอดเดรสเสมือน ซึ่งต้องการใช้โดยโปรแกรมที่รัน

ระบบปฏิบัติการยังสร้างเธรดเดี่ยวโดยอัตโนมัติภายใน การประมวลผลนั้น *เธรด* คือสถานะการประมวลผลปัจจุบันของอินสแตนซ์ เดี่ยวของโปรแกรม ใน AIX ให้เข้าถึงตัวประมวลผลและรีซอร์สอื่นๆ ที่จัดสรรไว้บนเธรด แทนที่จะเป็นการประมวลผล เธรดจำนวนมากสามารถสร้างขึ้นภายในการประมวลผล โดยแอ็พพลิเคชันโปรแกรม เธรดเหล่านี้จะแบ่งใช้รีซอร์สที่เป็นเจ้าของโดยการประมวลผล ภายในที่เธรดเหล่านั้นกำลังรันอยู่

ท้ายสุด ระบบจะแบรนช์ไปยัง entry point ของโปรแกรม หากเพจของโปรแกรมที่มี entry point ยังไม่ได้อยู่ในหน่วยความจำ (ตามที่ควรจะเป็น หากโปรแกรมคอมไพล์ เรียกใช้งาน หรือทำสำเนาแล้ว) อินเตอร์รัปต์ข้อบกพร่องของเพจที่เป็นผลลัพธ์จะเป็นสาเหตุของเพจที่ถูกอ่านจากหน่วยเก็บ ที่ทำสำรองไว้

Interrupt handlers

กลไกสำหรับการแจ้งเตือนระบบปฏิบัติการที่เข้าแทนที่เหตุการณ์ภายนอก เพื่ออินเตอร์รัปต์เธรดที่รันอยู่ในปัจจุบัน และถ่ายโอนการควบคุมไปยัง interrupt handler

ก่อนที่ interrupt handler จะสามารถรันได้ ต้องมีการบันทึกสถานะของฮาร์ดแวร์ที่เพียงพอ เพื่อมั่นใจว่า ระบบสามารถเรียกคืนบริบทของเธรดได้ หลังจากที่ interrupt handling เสร็จสิ้นแล้ว interrupt handlers ที่เรียกใช้งานใหม่ จะพบกับเวลาหน่วงทั้งหมดของการย้ายลำดับชั้นของฮาร์ดแวร์ขึ้น (ยกเว้นข้อบกพร่องของเพจ) เว้นเสียแต่ interrupt handler ถูกรันเมื่อเร็วๆ นี้ (หรือโปรแกรมที่สอดแทรกมีการประหยัดมาก) ซึ่งไม่เหมือนกับโค้ดหรือข้อมูลใดๆ ที่ยังคงอยู่ใน TLB หรือแคช

เมื่อเธรดที่อินเตอร์รัปต์ถูกจัดส่งอีกครั้ง การประมวลผลบริบท (เช่น เนื้อหาเรจิสเตอร์) จะถูกเรียกคืนแบบโลคัล ดังนั้นการประมวลผลนั้นจึงทำงานได้อย่างถูกต้อง อย่างไรก็ตาม เนื้อหาของ TLB และแคชต้องถูกสร้างขึ้นใหม่บนพื้นฐานของ ความต้องการในลำดับถัดมาของโปรแกรม ดังนั้น ทั้ง interrupt handler และเธรดที่อินเตอร์รัปต์สามารถพบกับเวลาหน่วงของแคชที่หายไป และ TLB ที่หายไป ซึ่งเป็นผลลัพธ์ของอินเตอร์รัปต์

เธรดที่กำลังรอ

เมื่อใดก็ตามที่โปรแกรมดำเนินการทำการร้องขอที่ไม่สามารถให้บริการได้ในทันที เช่น การดำเนินงานซิงโครนัส I/O (ทั้งทางตรงหรือเป็นผลของ page fault) เธรดนั้นจะถูกวางไว้ในสถานะกำลังรอจนกว่าการร้องขอ เสร็จสมบูรณ์

โดยปกติ เธรดที่กำลังรอส่งผลให้เกิด TLB และเวลาแฝงแคชขึ้นอีกชุดหนึ่ง เพิ่มเติมจากเวลาที่ต้องการสำหรับตัวการร้องขอเอง

ความสามารถในการจัดส่งเธรด

เมื่อเธรดมีความสามารถในการจัดส่งแต่ไม่ได้รันไว้ จึงไม่มีประโยชน์ใดๆ ที่จะทำให้บรรลุเป้าหมาย ร้ายไปกว่านั้น เธรดอื่นๆ ที่กำลังทำงานอยู่อาจทำให้แคชของเธรด ถูกนำกลับมาใช้ใหม่ และเพจหน่วยความจำจริงจะถูกเรียกคืน ซึ่งส่งผลทำให้เกิดเวลาหน่วงเพิ่มมากขึ้นเมื่อเธรดถูกจัดส่งในที่สุด

เธรดที่จัดส่งในปัจจุบัน

ตัวกำหนดตารางเวลาจะเลือกเธรดที่มีการรับที่แข็งแรงที่สุด เพื่อใช้กับตัวประมวลผล

ข้อควรพิจารณาที่มีผลกับตัวเลือกได้กล่าวถึงใน “ประสิทธิภาพตัวจัดตารางเวลาตัวประมวลผล” ในหน้า 43 เมื่อเธรดถูกจัดส่งแล้ว สถานะเชิงตรรกะของตัวประมวลผลจะถูกเรียกคืนเป็นสถานะที่ได้รับผลกระทบ เมื่อเธรดถูกอินเตอร์รัปต์

คำสั่งเครื่องในปัจจุบัน

คำสั่งเครื่องส่วนใหญ่จะมีความสามารถในการเรียกใช้งาน ในวงรอบของตัวประมวลผลเดี่ยว หากไม่มี TLB หรือแคชที่หายไปเกิดขึ้น

ในความเปรียบต่าง ถ้าโปรแกรมแตกสาขาไปยังพื้นที่ของโปรแกรมอื่นอย่างรวดเร็ว และเข้าถึงข้อมูลจากจำนวนขนาดใหญ่ของพื้นที่ที่แตกต่างกันซึ่งเป็นสาเหตุทำให้ TLB และอัตราแคชที่หายไปมีค่าสูง จำนวนเฉลี่ยของวงรอบตัวประมวลผลต่อคำสั่ง (CPI) ที่เรียกใช้งานอาจมีค่ามากกว่าจำนวนอื่นๆ โปรแกรมจะถูกบอกให้แสดงตำแหน่ง ที่อ้างอิงที่ไม่ดี ซึ่งอาจใช้จำนวนของคำสั่งเพียงเล็กน้อยเท่าที่จำเป็น เพื่อทำงาน แต่กำลังใช้จำนวนขนาดใหญ่ของวงรอบ ที่ไม่จำเป็น ในส่วนนี้ เนื่องจากความสัมพันธ์ที่แย่นี้ระหว่างจำนวนของคำสั่ง และจำนวนของวงรอบ การตรวจทานรายการโปรแกรมเพื่อคำนวณความยาวของพาธที่ไม่มีผลประโยชน์ใดๆ กับค่าโดยตรง ขณะที่พาธที่สั้นกว่าจะเร็วกว่าพาธที่ยาว อัตราความเร็วสามารถแตกต่างจากอัตราความยาวของพาธ

คอมไพลเลอร์จะจัดเรียงโค้ดด้วยวิธีที่ที่ตนเองมีประสบการณ์เพื่อลดจำนวนของวงรอบ ที่ต้องการสำหรับการประมวลผลโปรแกรม โปรแกรมเมอร์จะค้นหาผลการทำงานสูงสุด ซึ่งต้องเกี่ยวข้องกับคอมไพลเลอร์ ที่มีข้อมูลทั้งหมดที่จำเป็นต่อการออปติไมซ์โค้ดอย่างมีประสิทธิผล แทนที่จะพยายามคาดเดาเทคนิคการออปติไมซ์ของคอมไพลเลอร์ในครั้งที่สอง (โปรดดู การใช้ตัวประมวลผลก่อนและคอมไพลเลอร์ อย่างมีประสิทธิผล) การวัดค่าจริงของการออปติไมซ์อย่างมีประสิทธิภาพ คือผลการทำงานของเวิร์กโหลดที่น่าเชื่อถือ

การปรับระบบ

หลังจากนำแอ็พพลิเคชันโปรแกรมไปใช้อย่างมีประสิทธิภาพแล้ว การปรับปรุง เพิ่มเติมในประสิทธิภาพโดยรวมของระบบกลายเป็นเรื่องของการปรับระบบ

คอมโพเนนต์หลักที่เกี่ยวข้องกับการปรับระดับระบบคือ:

I/O การสื่อสาร

ขึ้นอยู่กับชนิดของเวิร์กโหลดและชนิดของลิงก์การสื่อสาร อาจจำเป็นต้องปรับไดรเวอร์อุปกรณ์การสื่อสารตั้งแต่หนึ่งรายการขึ้นไป : TCP/IP, หรือ NFS

ดิสก์คงที่

Logical Volume Manager (LVM) ควบคุมการวางตำแหน่งของระบบไฟล์ และพื้นที่ว่างการเพจบนดิสก์ ซึ่งอาจส่งผลกระทบอย่างมากต่อจำนวนของเวลาแฝง การค้นหาบนระบบ ไดรเวอร์อุปกรณ์ดิสก์ควบคุมลำดับในการดำเนินการกับ การร้องขอ I/O

หน่วยความจำจริง

Virtual Memory Manager (VMM) ควบคุมพูลของเฟรมหน่วยความจำจริง ที่ว่าง และกำหนดว่าจะขโมยเฟรมเพื่อเติมพูลเมื่อไรและจาก ที่ใด

เธรดที่กำลังรัน

ตัวจัดตารางเวลากำหนดเอนทิตีที่จัดส่งได้ซึ่งควรจะได้รับการควบคุม ถัดไปใน AIX เอนทิตีที่จัดส่งได้คือเธรด โปรด ดู “การสนับสนุนเธรด” ในหน้า 44

การปรับประสิทธิภาพ

การปรับประสิทธิภาพของระบบและเวิร์กโหลดเป็นสิ่งที่สำคัญมาก

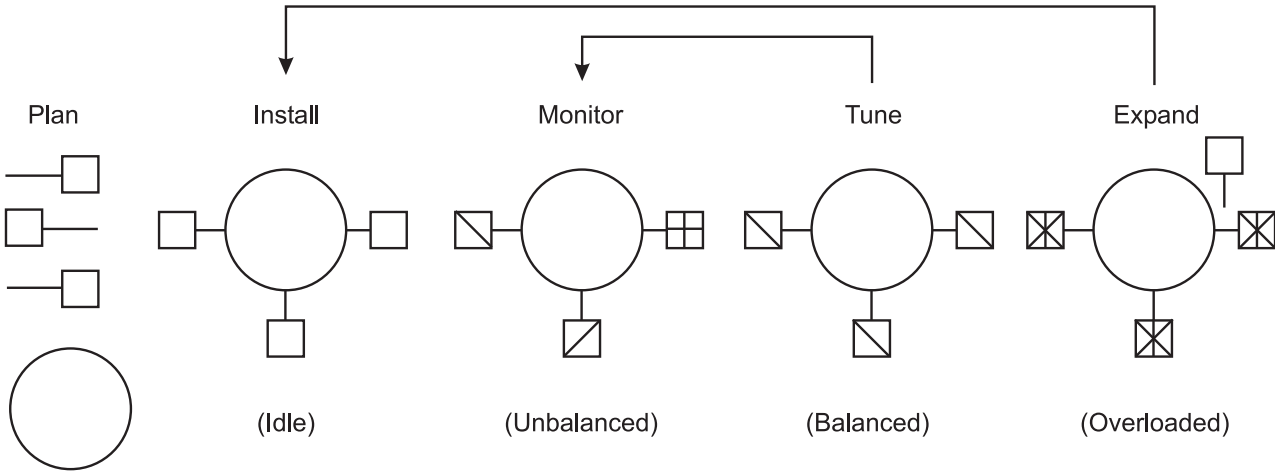
กระบวนการปรับผลการทำงาน

การปรับผลการทำงานคือปัญหาหลักของการจัดการรีซอร์ส และแก้ไขค่าติดตั้งพารามิเตอร์ของระบบ

การปรับเวิร์กโหลดและระบบสำหรับการใช้รีซอร์สให้มีประสิทธิภาพประกอบด้วย ขั้นตอนต่อไปนี้:

1. การระบุเวิร์กโหลดบนระบบ
2. การตั้งค่าวัตถุประสงค์:
   a. การพิจารณาการวัดค่าผลลัพธ์
   b. การรับรองและการกำหนดระดับความสำคัญของวัตถุประสงค์
3. การระบุรีซอร์สที่สำคัญที่จำกัดผลการทำงานของระบบ
4. การลดข้อกำหนดของรีซอร์สที่สำคัญของเวิร์กโหลด:
   a. การใช้รีซอร์สให้เหมาะสมมากที่สุด ถ้ามีตัวเลือก
   b. การลดข้อกำหนดรีซอร์สที่สำคัญของโปรแกรมแต่ละโปรแกรม หรือฟังก์ชันของระบบ
   c. การกำหนดโครงสร้างสำหรับการใช้รีซอร์สแบบขนาน
5. การแก้ไขการจัดสรรรีซอร์สเพื่อสะท้อนถึงระดับความสำคัญ
   a. การเปลี่ยนระดับความสำคัญหรือข้อจำกัดรีซอร์สของโปรแกรมแต่ละโปรแกรม
   b. การเปลี่ยนค่าติดตั้งของพารามิเตอร์การจัดการรีซอร์สของระบบ
6. การทำซ้ำขั้นตอนที่ 3 ถึง 5 จนกว่าจะตรงกับวัตถุประสงค์ (หรือรีซอร์ส ถูกเติมเต็ม)
7. การใช้รีซอร์สเพิ่มเติม ถ้าจำเป็น

มีเครื่องมือที่เหมาะสมสำหรับแต่ละเฟสของการจัดการผลการทำงานของระบบ (โปรดดู “การมอนิเตอร์และปรับคำสั่งและรูทีนย่อย” ในหน้า 448) เครื่องมือบางตัวพร้อมใช้งานจาก IBM® เครื่องมืออื่นๆ คือผลิตภัณฑ์ของกลุ่มที่สาม ภาพประกอบต่อไปนี้แสดงภาพของเฟสการจัดการผลการทำงาน ในสภาวะแวดล้อม LAN แบบง่ายๆ



รูปที่ 2. เฟสผลการทำงาน. ภาพประกอบใช้วงกลมที่มีน้ำหนักห้าวง เพื่อแสดงภาพขั้นตอนของการปรับผลการทำงานของระบบ วางแผน ติดตั้ง มอนิเตอร์ ปรับ และขยาย แต่ละวงกลมจะแสดงระบบ ในสถานะของผลการทำงานต่างๆ; ไม่ทำงาน ไม่สมดุล สมดุล และโหลดมากเกินไป ยิ่งไปกว่านั้น คุณขยายระบบที่โหลดมากเกินไป ปรับระบบจนกว่าจะสมดุล มอนิเตอร์ระบบที่ไม่สมดุล และติดตั้งรีซอร์สเพิ่มเติมเมื่อมีการขยายที่จำเป็น

Identification ของเวิร์กโหลด

สิ่งสำคัญคือ การทำงานทั้งหมดต้องถูกดำเนินการตามระบบ ที่ระบุไว้โดยเฉพาะในระบบที่เชื่อมต่อกับ LAN ชุดของระบบไฟล์ที่ซับซ้อน สามารถพัฒนาด้วยข้อตกลงเฉพาะที่ใช้ในหมู่ผู้ใช้ ของระบบ ระบบไฟล์เหล่านี้ต้องถูกระบุไว้ และพิจารณาเป็นส่วนหนึ่งของกิจกรรมการปรับใดๆ

ด้วยเวิร์กโหลดแบบผู้ใช้หลายคน นักวิเคราะห์ต้องหาจำนวนทั้งอัตราคำร้องขอปกติ และอัตราคำร้องขอสูงสุด ซึ่งเป็นสิ่งสำคัญที่ปฏิบัติได้จริงเกี่ยวกับการแบ่งสัดส่วนของเวลา ซึ่งผู้ใช้โต้ตอบกับเทอร์มินัล

องค์ประกอบที่สำคัญของขั้นตอนการ identification นี้คือ การพิจารณาถึงการวัดค่าและการปรับกิจกรรมที่ได้กระทำบนระบบที่ใช้งานจริง หรือสามารถบรรลุเป้าหมายบนระบบอื่น (หรือ off-shift) ด้วยเวอร์ชันที่จำลองไว้ ของเวิร์กโหลดที่แท้จริง นักวิเคราะห์ต้องวัดน้ำหนักความถูกต้องของผลลัพธ์จาก สภาวะแวดล้อมที่ใช้งานจริงกับความยืดหยุ่นของสภาวะแวดล้อมที่ไม่ได้ใช้งานจริง ซึ่งนักวิเคราะห์สามารถดำเนินการทดสอบการลดระดับผลการทำงานที่เลี่ยง หรือไม่มีค่า

ความสำคัญของวัตถุประสงค์ในการตั้งค่า

แม้ว่าคุณสามารถตั้งค่าวัตถุประสงค์ในแง่ของปริมาณการวัด ผลลัพธ์ที่ต้องการจริงมักจะเป็นตัวกระทำ เช่น เวลาตอบสนองที่พึงพอใจ นอกจากนี้ นักวิเคราะห์ต้องต่อต้านความพยายามในการปรับสิ่งที่สามารถวัดค่าได้ แทนที่จะเป็นสิ่งที่สำคัญ ถ้าไม่มีการวัดค่าที่จัดเตรียมไว้โดยระบบ ซึ่งสอดคล้องกับการปรับปรุงตามความต้องการ การวัดค่านั้นต้องถูกออกแบบขึ้นใหม่

ลักษณะมุมมองที่มีค่าของการแสดงปริมาณตามวัตถุประสงค์จะไม่ใช่การเลือกจำนวนที่บรรลุเป้าหมาย แต่จะทำการตัดสินใจเกี่ยวกับความสำคัญที่เกี่ยวข้องของ วัตถุประสงค์ทั้งหลาย (ตามปกติ) ยกเว้นว่า ระดับความสำคัญเหล่านี้จะถูกตั้งค่าไว้ล่วง

หน้า และมีทุกคนเข้าใจถึงสิ่งที่เกี่ยวข้อง นักวิเคราะห์จะไม่สามารถเปลี่ยนการตัดสินใจได้โดยไม่มีการปรึกษาอย่างต่อเนื่อง นักวิเคราะห์ยังอาจประหลาดใจต่อปฏิกิริยาของผู้ใช้ หรือผู้บริหารถึงลักษณะมุมมองของผลการทำงานที่ถูกละเว้น ถ้าส่วนสนับสนุนและการใช้ระบบอยู่ระหว่างขอบเขตขององค์กร คุณอาจจำเป็นต้องเขียนข้อตกลงเกี่ยวกับระดับของการบริการระหว่างผู้ให้บริการ และผู้ใช้เพื่อมั่นใจว่า มีความเข้าใจในวัตถุประสงค์ของผลการทำงาน และระดับความสำคัญที่ตรงกัน

Identification ของรีซอร์สที่สำคัญ

โดยทั่วไป ผลการทำงานจะมีเวิร์กโหลดที่ต้องถูกพิจารณาโดยสภาพพร้อมใช้งาน และความเร็วของรีซอร์สของระบบที่สำคัญตั้งแต่หนึ่งตัวขึ้นไป นักวิเคราะห์ต้องระบุรีซอร์สเหล่านั้นอย่างถูกต้องหรือความเสี่ยงที่ตกอยู่ในการดำเนินการแบบ trial-and-error ที่ไม่มีจุดสิ้นสุด

ระบบมีรีซอร์สจริง โลจิคัล และรีซอร์สเสมือน รีซอร์สจริงที่สำคัญ จะระบุได้ง่ายขึ้น เนื่องจากทูลด้านประสิทธิภาพการทำงานของระบบเพิ่มเติม จะพร้อมใช้งานเพื่อประเมินผลการใช้ประโยชน์ของรีซอร์สจริง รีซอร์สจริง ที่มีผลกระทบกับผลการทำงานดังต่อไปนี้:

- วงรอบ CPU
- หน่วยความจำ
- บัส I/O
- อะแด็ปเตอร์ต่างๆ
- พื้นที่ดิสก์
- การเข้าถึงเน็ตเวิร์ก

รีซอร์สแบบโลจิคัลจะถูกระบุไว้น้อยที่สุด โลจิคัลรีซอร์ส จะเป็นการโปรแกรมมิงที่พาร์ติชันรีซอร์สจริง การแบ่งพาร์ติชัน จะทำเพื่อแบ่งใช้และจัดการกับรีซอร์สจริง

คุณสามารถใช้รีซอร์สเสมือนบนระบบแบบอิง POWER5 สำหรับ IBM System p ซึ่งประกอบด้วย Micro-Partitioning® อะแด็ปเตอร์อนุกรมเสมือน SCSI เสมือนและอีเทอร์เน็ตเสมือน

ตัวอย่างของรีซอร์สจริงบางตัวและโลจิคัลและรีซอร์สเสมือนจะสร้างไว้ ดังต่อไปนี้:

CPU

- การแบ่งเวลาตัวประมวลผล
- การให้สิทธิ์ CPU หรือ การแบ่งไมโครพาร์ติชัน
- อีเทอร์เน็ตเสมือน

หน่วยความจำ

- กรอบของเพจ
- สแต็ก
- บัฟเฟอร์
- คิว
- ตาราง
- ล็อกและอุปกรณ์สัญญาณ

พื้นที่ดิสก์

- โลจิคัลวอลุ่ม
- ระบบไฟล์
- ไฟล์
- โลจิคัลพาร์ติชัน
- SCSI เสมือน

การเข้าถึงเน็ตเวิร์ก

- เซสชัน
- แพ็กเก็ต
- แชนเนล
- อีเทอร์เน็ตที่แบ่งใช้

สิ่งสำคัญคือ การรับรู้ของโลจิคัลและรีซอร์สเสมือนเช่นเดียวกับ รีซอร์สจริง เธรดสามารถบล็อกได้โดยการขาดแคลนโลจิคัลรีซอร์ส เช่นเดียวกับการขาดแคลนรีซอร์สจริง และการขยายรีซอร์สจริง ไม่ได้ทำให้มั่นใจว่า โลจิคัลรีซอร์สเพิ่มเติมจะถูกสร้างขึ้น ตัวอย่างเช่น เซิร์ฟเวอร์ NFS daemon หรือ **nfsd** daemon บนเซิร์ฟเวอร์ที่ต้องมีการจัดการ NFS ที่คงค้างคำร้องขอ I/O แบบรีโมต จำนวนของ **nfsd** จะจำกัดจำนวนของการดำเนินการ NFS I/O ที่สามารถอยู่ในการดำเนินการอย่างพร้อมเพียงกัน เมื่อขาดแคลน **nfsd** daemon ที่มีอยู่ เครื่องมือของระบบอาจจะบ่งชี้ถึงรีซอร์สจริงที่ต่างๆ เช่น CPU จะถูกใช้เท่านั้น คุณต้องมีความพึงพอใจที่ผิด ซึ่งระบบของคุณจะอยู่ภายใต้การใช้และช้า เมื่อความเป็นจริงที่ว่า คุณขาด **nfsd** ที่เป็นข้อจำกัดของรีซอร์สส่วนที่เหลือ **nfsd** จะใช้วงรอบตัวประมวลผลและหน่วยความ แต่คุณจะไม่สามารถแก้ไขปัญหานี้ได้โดยเพิ่มหน่วยความจำจริงและอัพเกรด CPU ให้เร็วขึ้น โซลูชันคือ สร้างโลจิคัลรีซอร์สเพิ่มเติม **nfsd** daemons

โลจิคัล รีซอร์สและคอขวดสามารถสร้างขึ้นได้ในระหว่างการพัฒนาแอ็พพลิเคชัน เมธอดการส่งผ่านข้อมูลและการควบคุมอุปกรณ์อาจสร้าง โลจิคัลรีซอร์ส เมื่อรีซอร์สที่สร้างขึ้นโดยไม่ตั้งใจ จะไม่มีเครื่องมือในการมอนิเตอร์การใช้ และไม่มีอินเตอร์เฟสเพื่อควบคุมการจัดสรร การมีอยู่อาจไม่มีความต้องการจนกระทั่งปัญหาในเรื่องผลการทำงานเฉพาะ จะถูกกำหนดให้มีความสำคัญ

การลดข้อกำหนดเกี่ยวกับรีซอร์สที่สำคัญ

พิจารณาถึงการลดข้อกำหนดเกี่ยวกับรีซอร์สที่สำคัญ ในสามระดับ

การใช้รีซอร์สที่เหมาะสม:

การเลือกใช้รีซอร์สหนึ่งแทนอีกรีซอร์สหนึ่งควรทำด้วยความระมัดระวัง และมีเป้าหมายเฉพาะชัดเจน

ตัวอย่างของตัวเลือกรีซอร์สในระหว่างการพัฒนาแอ็พพลิเคชันจะเป็น การแลกเปลี่ยนระหว่างการใช้หน่วยความจำที่เพิ่มขึ้นกับการใช้ CPU ที่ลดลง การตัดสินใจ เกี่ยวกับการตั้งค่าคอนฟิกระบบทั่วไปที่แสดงถึงตัวเลือกรีซอร์สคือ จะวางไฟล์แบบโลคัลบนแต่ละเวิร์กสเตชันหรือจะวางแบบรีโมต บนเซิร์ฟเวอร์

การลดความต้องการรีซอร์สที่สำคัญ:

สำหรับแอ็พพลิเคชันที่พัฒนาขึ้นแบบโลคัล สามารถปรับปรุงโปรแกรม เพื่อให้ทำฟังก์ชันเดียวกันได้อย่างมีประสิทธิภาพมากขึ้น หรือเพื่อลบฟังก์ชัน ที่ไม่จำเป็น

ที่ระดับการจัดการระบบ สามารถย้ายเวิร์กโหลดที่มีระดับความสำคัญต่ำซึ่งช่วงชิง รีซอร์สสำคัญไปยังระบบอื่น รันในเวลาอื่น หรือควบคุมได้โดยใช้ Workload Manager

การจัดโครงสร้างสำหรับการใช้รีซอร์สแบบขนาน:

เนื่องจากเวิร์กโหลดต้องใช้รีซอร์สระบบหลายอย่างในการรัน ควรใช้ ประโยชน์จากข้อเท็จจริงที่ว่ารีซอร์สแยกต่างหากกัน และสามารถใช้ ในแบบขนานได้

ตัวอย่างเช่น ขั้นตอนวิธี read-ahead ของระบบปฏิบัติการตรวจพบข้อเท็จจริง ว่า โปรแกรมกำลังเข้าถึงไฟล์ตามลำดับ และจัดตารางเวลาการอ่านตามลำดับ เพิ่มเติมที่จะทำในแบบขนานพร้อมกับการประมวลผลข้อมูลก่อนหน้านี้ของ แอ็พพลิเคชัน ระบบขนานใช้กับการจัดการระบบด้วย ตัวอย่างเช่น ถ้าแอ็พพลิเคชัน เข้าถึงไฟล์ตั้งแต่สองไฟล์ขึ้นไปในเวลาเดียวกัน การเพิ่มดิสก์ไดร์ฟเพิ่มเติม อาจช่วยปรับปรุงอัตราดิสก์ I/O ได้ ถ้าไฟล์ที่มีการเข้าถึงพร้อมกัน วางอยู่บนไดร์ฟต่างกัน

ระดับความสำคัญในการจัดสรรรีซอร์ส

ระบบปฏิบัติการมีวิธีหลายอย่างในการจัดระดับความสำคัญกิจกรรม

ส่วนประกอบบางอย่าง เช่น disk pacing มีการตั้งค่าที่ระดับระบบ สิ่งอื่น เช่น ระดับความสำคัญของกระบวนการ สามารถตั้งค่าโดยผู้ใช้แต่ละรายเพื่อสะท้อน ถึงความสำคัญต่อภารกิจเฉพาะ

การทำซ้ำขั้นตอนการปรับ

หลักของการวิเคราะห์ประสิทธิภาพคือมีปัญหาคอขวดถัดไป เสมอ การลดการใช้รีซอร์สหนึ่งหมายความว่ารีซอร์สอื่นจำกัดผลผลิต หรือเวลาตอบกลับ

ตัวอย่างเช่น สมมุติว่าเรามีระบบซึ่งมีระดับการใช้ประโยชน์ ดังนี้:

CPU: 90% ดิสก์: 70% หน่วยความจำ 60%

เวิร์กโหลดนี้ ยึดอยู่กับ CPU ถ้าเราปรับเวิร์กโหลดได้สำเร็จจน CPU โหลดลดลง จาก 90 เป็น 45 เปอร์เซ็นต์ เราอาจคาดได้ว่าประสิทธิภาพที่พัฒนาขึ้นมีทั้งผลบวกและผลลบ โชคไม่ดี ขณะนี้เวิร์กโหลดมี I/O จำกัด โดยมีการใช้ประโยชน์โดยประมาณดังต่อไปนี้:

CPU: 45% ดิสก์: 90% หน่วยความจำ 60%

การใช้ประโยชน์ CPU ที่ดีขึ้นช่วยให้โปรแกรมสามารถส่งการร้องขอดิสก์ได้เร็วขึ้น แต่จากนั้น เราต้องเจอกับขีดจำกัดความจุของดิสก์ไดร์ฟ บางทีการพัฒนาประสิทธิภาพ อาจเป็น 30 เปอร์เซ็นต์แทนที่จะเป็น 100 เปอร์เซ็นต์ดังที่เราเห็น

มีรีซอร์ส ที่สำคัญใหม่เสมอ คำถามที่สำคัญคือเราปฏิบัติได้ตามวัตถุประสงค์ด้าน ประสิทธิภาพด้วยรีซอร์สที่มีอยู่หรือไม่

ข้อควรสนใจ: การปรับระบบ ที่ไม่เหมาะสมด้วยคำสั่งการปรับ **vmo, ioo, schedo, no,** และ **nfso** อาจส่งผลให้เกิดพฤติกรรมระบบที่ไม่ได้คาดไว้ เช่น ประสิทธิภาพระบบหรือแอ็พพลิเคชันที่ด้อยลง หรือระบบไม่ตอบสนอง ควรใช้การเปลี่ยนแปลงเฉพาะถ้ามีการระบุปัญหาคอขวดโดยการวิเคราะห์ ประสิทธิภาพแล้วเท่านั้น

หมายเหตุ: ข้อแนะนำทั่วไปคือ ประสิทธิภาพขึ้นอยู่กับค่าติดตั้งการปรับ

การใช้รีซอร์สเพิ่มเติม

หลังจากที่วิธีการก่อนหน้านี้ได้ถูกยกเลิก ถ้าผลการทำงานของระบบยังคงไม่ตรงกับวัตถุประสงค์ รีซอร์สที่สำคัญอาจถูกปรับปรุงหรือขยายเพิ่ม

ถ้ารีซอร์สที่สำคัญคือโลจิคัลและภายใต้รีซอร์สที่ใช้จริงมีเพียงพอ โลจิคัลรีซอร์ศสามารถขยายเพิ่มได้โดยไม่มีค่าใช้จ่ายเพิ่มเติม ถ้ารีซอร์สที่สำคัญนี้เกิดขึ้นจริง การวิเคราะห์ต้องตรวจสอบคำถามเพิ่มเติมบางข้อ ต่อไปนี้:

- จำนวนเท่าใดที่ต้องปรับปรุงหรือขยายรีซอร์สที่สำคัญเพิ่มเติม เพื่อให้จบปัญหาเรื่องคอขวด?
- ผลการทำงานของระบบตรงกับวัตถุประสงค์ หรือ รีซอร์สอื่นจะนำมาใช้ก่อน?
- ถ้ามีความต่อเนื่องของรีซอร์สที่สำคัญ ความต่อเนื่องนี้สมเหตุสมผลกับการปรับปรุง หรือขยายเพิ่มรีซอร์สทั้งหมด หรือแบ่งเวิร์กโหลดปัจจุบัน ให้กับระบบอื่น?

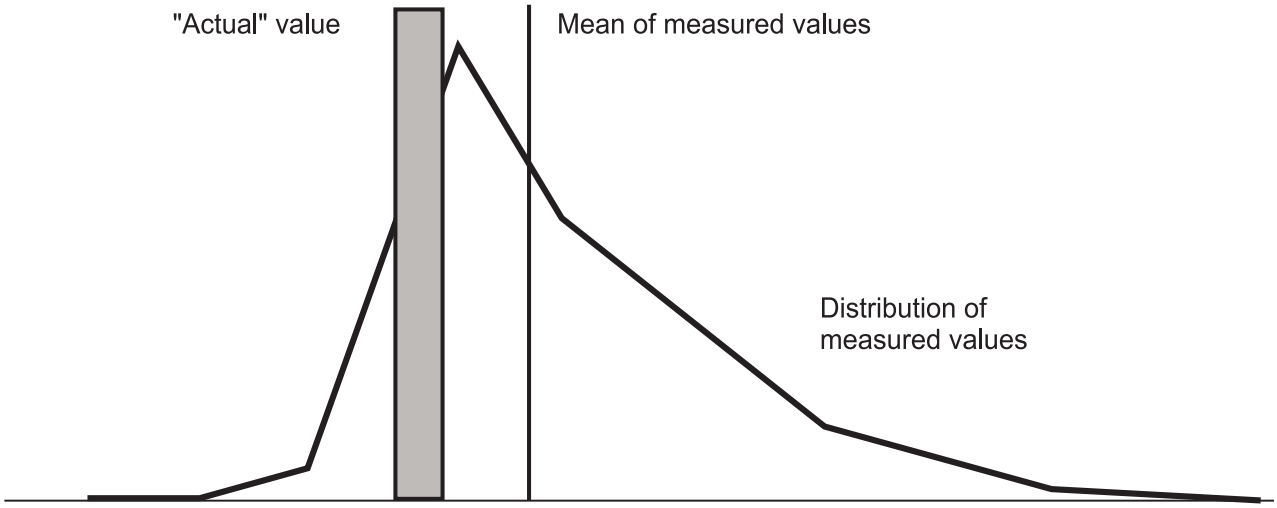
การกำหนดเกณฑ์เปรียบเทียบประสิทธิภาพ

เมื่อเราพยายามเปรียบเทียบประสิทธิภาพของชิ้นส่วนที่กำหนดของ ซอฟต์แวร์ในสภาพแวดล้อมที่แตกต่างกัน เราต้องคำนึงถึงข้อผิดพลาดที่อาจเกิดขึ้น ได้ ข้อผิดพลาดด้านเทคนิค และข้อผิดพลาดด้านหลักการจำนวนมาก ส่วนนี้มีข้อมูล ที่ควรระวังมากที่สุด ส่วนอื่นของชุดหัวข้อนี้จะอธิบายวิธีต่างๆ ที่สามารถวัดเวลาที่ผ่านไปและเวลาเฉพาะกระบวนการ

เมื่อเราประเมินเวลาที่ผ่านไป (นาฬิกาติดผนัง) ซึ่งต้องใช้ในการประมวลผล การเรียกระบบ เราจะได้รับตัวเลขที่ประกอบด้วยข้อมูลดังต่อไปนี้:

- เวลาที่ใช้จริงในระหว่างการดำเนินการคำสั่งเพื่อทำ การบริการ
- จำนวนเวลาที่แตกต่างกันไปซึ่งตัวประมวลผลหยุดในขณะที่รอ คำสั่งหรือข้อมูลจากหน่วยความจำ (นั่นคือต้นทุนของ cache และ TLB misses)
- เวลาที่ต้องใช้ในการเข้าถึงนาฬิกาเมื่อเริ่มต้นและสิ้นสุด การเรียก
- เวลาที่ใช้โดยเหตุการณ์เป็นครั้งคราว เช่น การขัดจังหวะของไทม์เมอร์ระบบ
- เวลาที่ใช้โดยเหตุการณ์แบบสุ่ม เช่น การขัดจังหวะ I/O

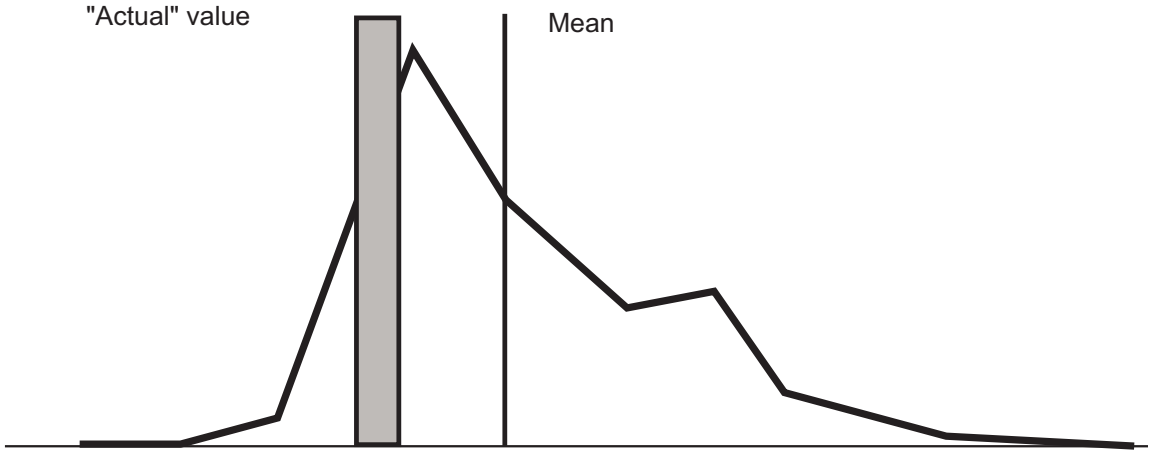
เพื่อหลีกเลี่ยงการรายงานตัวเลขที่ไม่ถูกต้อง โดยปกติแล้ว เราจะประเมิน เวิร์กโหลดหลายๆ ครั้ง เนื่องจากปัจจัยพิเศษทั้งหมดทำให้ เวลาการประมวลผลที่ใช้จริงเพิ่มขึ้น จึงมีการจัดทำ curve สำหรับชุดการ ประเมินปกติของแบบฟอร์มดังแสดงในภาพสาธิตต่อไปนี้



รูปที่ 3. Curve สำหรับชุดการประเมินปกติ.

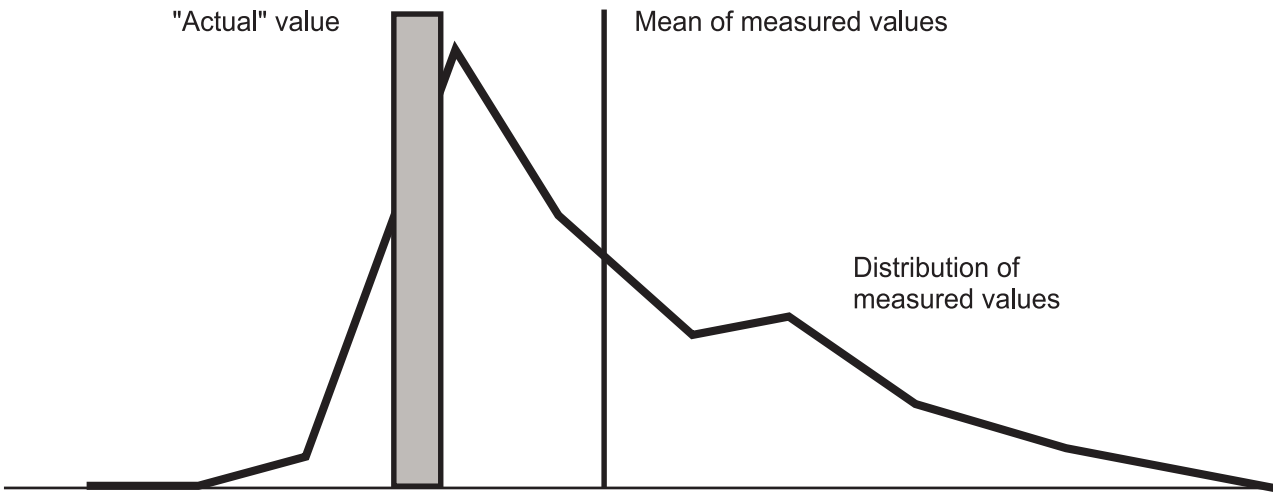
ปลายที่ต่ำมากแสดงถึงสถานการณ์การแคชดีที่สุดที่มีโอกาสต่ำ หรืออาจเป็นผลกระทบจากการปัดเศษ

เหตุการณ์พิเศษที่เกิดขึ้นซ้ำแบบปกติอาจให้ curve รูปแบบทวิฐานนิยม (bimodal) (ค่าสูงสุดสองค่า) ดังแสดงในภาพสาธิตต่อไปนี้



รูปที่ 4. Curve แบบทวิฐานนิยม

การขัดจังหวะที่ใช้เวลานานหนึ่งหรือสองครั้งอาจทำให้ curve เบ้มากขึ้น ดังแสดงในภาพสาธิตต่อไปนี้:



รูปที่ 5. Curve ที่เบ้มากขึ้น

การกระจายตัวของการประเมินค่าที่แท้จริงไม่ได้เป็นแบบสุ่ม และการทดสอบยอดนิยมตลอดกาลของสถิติเชิงอนุมานสามารถใช้ได้ด้วย ความระวังเป็นพิเศษเท่านั้น นอกจากนี้ ขึ้นอยู่กับวัตถุประสงค์ของการประเมิน ทั้งค่าเฉลี่ยและค่าที่แท้จริงอาจไม่ใช่สิ่งชี้วัดที่เหมาะสม ของประสิทธิภาพ

การมอนิเตอร์ประสิทธิภาพระบบ

AIX มีเครื่องมือ และเทคนิคหลายอย่างสำหรับการมอนิเตอร์กิจกรรมระบบที่เกี่ยวข้องกับประสิทธิภาพ

ประโยชน์ของการมอนิเตอร์ผลการทำงานของระบบอย่างต่อเนื่อง

มีประโยชน์อยู่หลายข้อสำหรับการมอนิเตอร์ผลการทำงานของระบบ อย่างต่อเนื่อง

การมอนิเตอร์ผลการทำงานของระบบอย่างต่อเนื่องสามารถทำได้ดังนี้:

- ตรวจพบปัญหาที่เกิดขึ้นก่อนที่จะเกิดผลกระทบในบางครั้ง
- ตรวจพบปัญหาที่กระบทบกับผลการทำงานของผู้ใช้
- เก็บรวบรวมข้อมูลเมื่อเกิดปัญหาขึ้นในครั้งแรก
- อนุญาตให้คุณสร้างเส้นบรรทัดสำหรับการเปรียบเทียบ

การมอนิเตอร์ที่เป็นผลสำเร็จจะเกี่ยวข้องกับสิ่งต่อไปนี้:

- การขอรับข้อมูลที่เกี่ยวข้องกับผลการทำงานเป็นช่วงเวลาจาก ระบบปฏิบัติการ
- การเก็บข้อมูลสำหรับใช้ในอนาคตเพื่อการวินิจฉัยปัญหา
- การแสดงข้อมูลสำหรับผลประโยชน์ที่ผู้ดูแลระบบจะได้รับ
- การตรวจพบสถานการณ์ที่ต้องการเก็บรวบรวมข้อมูลเพิ่มเติม หรือการตอบกลับไปยังคำสั่งจากผู้ดูแลระบบเพื่อเก็บรวบรวมข้อมูล
- การเก็บรวบรวมและการเก็บรายละเอียดข้อมูลที่จำเป็น
- การเปลี่ยนแปลงการติดตามที่ทำกับระบบและแอ็พพลิเคชัน

การมอนิเตอร์ผลการทำงานของระบบอย่างต่อเนื่องด้วยคำสั่ง

คำสั่ง **vmstat, iostat, netstat** และ **sar** จะจัดเตรียมรากฐานที่คุณสามารถสร้าง กลไกการมอนิเตอร์ผลการทำงานได้

คุณสามารถเขียนสคริปต์ shell เพื่อดำเนินการกับการลดทอนข้อมูลเกี่ยวกับเอาต์พุตคำสั่ง คำเตือนเกี่ยวกับปัญหาของผลการทำงาน หรือบันทึกข้อมูลเกี่ยวกับสถานะของระบบ เมื่อปัญหากำลังเกิดขึ้น ตัวอย่างเช่น สคริปต์ shell สามารถทดสอบเปอร์เซ็นต์เวลาสูญเปล่าของ CPU ที่มีค่าศูนย์ เงื่อนไขที่อิ่มตัว และเรียกใช้งานสคริปต์ shell อื่นๆ เมื่อเงื่อนไข CPU ที่อิ่มตัวเกิดขึ้น สคริปต์ต่อไปนี้จะเร็กคอร์ดการประมวลผลที่แอ็คทีฟ 15 กระบวนการซึ่งใช้เวลา CPU ส่วนใหญ่นอกเหนือจากการประมวลผลที่เป็นเจ้าของโดยผู้ใช้ ของสคริปต์:

```
# ps -ef | egrep -v "STIME|$LOGNAME" | sort +3 -r | head -n 15
```

การมอนิเตอร์ผลการทำงานอย่างต่อเนื่องด้วยคำสั่ง vmstat

คำสั่ง **vmstat** จะมีประโยชน์สำหรับการขอรับภาพรวมของ CPU การเพจ และการใช้หน่วยความจำ

ต่อไปนี้คือตัวอย่างรายงานที่สร้างขึ้นด้วยคำสั่ง **vmstat**:

```
# vmstat 5 2
kthr     memory             page              faults        cpu
----- ----------- ------------------------ ------------ -----------
 r  b   avm   fre  re  pi  po  fr   sr  cy  in   sy  cs us sy id wa
 1  1 197167 477552   0   0   0   7   21   0 106 1114 451  0  0 99  0
 0  0 197178 477541   0   0   0   0    0   0 443 1123 442  0  0 99  0
```

โปรดจำไว้ว่า รายงานอันดับแรกจากคำสั่ง **vmstat** จะแสดงกิจกรรมที่สะสมไว้ตั้งแต่การบูตระบบครั้งล่าสุด รายงานที่สองจะแสดงกิจกรรม สำหรับช่วงเวลา 5 วินาทีแรก

สำหรับการอภิปรายโดยละเอียดของคำสั่ง **vmstat** โปรดดู “คำสั่ง vmstat” ในหน้า 112, “การพิจารณาการใช้หน่วยความจำด้วยคำสั่ง vmstat” ในหน้า 139 และ “การประเมินค่าผลการทำงานดิสก์ด้วยคำสั่ง vmstat” ในหน้า 199

การมอนิเตอร์ผลการทำงานอย่างต่อเนื่องด้วยคำสั่ง iostat

คำสั่ง **iostat** มีประโยชน์สำหรับการพิจารณาดิสก์ และการใช้ CPU

ระบบปฏิบัติการ AIX เก็บรักษา ประวัติของกิจกรรมดิสก์ ในตัวอย่างต่อไปนี้ ประวัติดิสก์ I/O ปิดใช้งานเนื่องจากข้อความต่อไปนี้แสดงขึ้น:

```
Disk history since boot not available.

The interval disk I/O statistics are unaffected by this.
```

เมื่อต้องการ เปิดใช้งานประวัติดิสก์ I/O จากบรรทัดรับคำสั่ง ให้ป้อน `smit chgsys` จากนั้นเลือก **true** จากฟิลด์ **เก็บรักษาประวัติดิสก์ I/O อย่างต่อเนื่อง**

รายงาน ตัวอย่างต่อไปนี้แสดงขึ้นเมื่อคุณรันคำสั่ง **iostat**:

```
# iostat 5 2

tty:      tin         tout   avg-cpu:  % user    % sys     % idle    % iowait
          0.1        102.3               0.5      0.2       99.3       0.1

               Disk history since boot not available.
```

```
               The interval disk I/O statistics are unaffected by this.

tty:      tin         tout   avg-cpu:  % user    % sys     % idle    % iowait
          0.2        79594.4               0.6      6.6       73.7      19.2

Disks:        % tm_act     Kbps      tps    Kb_read   Kb_wrtn
hdisk1           0.0       0.0       0.0          0         0
hdisk0          78.2     1129.6     282.4       5648         0
cd1              0.0       0.0       0.0          0         0
```

รายงานแรกจากคำสั่ง **iostat** จะแสดงกิจกรรมที่สะสมไว้ตั้งแต่การรีเซ็ตครั้งสุดท้ายของตัวนับกิจกรรม ของดิสก์ รายงานที่สองจะแสดงกิจกรรม สำหรับช่วงเวลา 5 วินาทีแรก

หลักการที่เกี่ยวข้อง:

"คำสั่ง iostat" ในหน้า 115

คำสั่ง **iostat** จะเป็นวิธีที่เร็วที่สุด ในการขอรับความพึงพอใจในครั้งแรก ไม่ว่าระบบจะมีปัญหาเกี่ยวกับผลการทำงานในขอบเขตของดิสก์ I/O หรือไม่ก็ตาม

งานที่เกี่ยวข้อง:

"การประเมินค่าผลการทำงานดิสก์ด้วยคำสั่ง iostat" ในหน้า 196

เริ่มต้นการประเมินผลโดยรันคำสั่ง **iostat** ด้วยพารามิเตอร์ช่วงเวลาในระหว่างที่มีเวิร์กโหลดสูงสุดของระบบ หรือขณะที่รันแอ็พพลิเคชันที่สำคัญซึ่งคุณต้องการลดเวลาหน่วงของ I/O

การมอนิเตอร์ผลการทำงานอย่างต่อเนื่องด้วยคำสั่ง netstat

คำสั่ง **netstat** จะมีข้อได้เปรียบในการพิจารณา จำนวนของแพ็กเก็ตขาส่งและแพ็กเก็ตที่ได้รับ

ต่อไปนี้คือรายงานตัวอย่างที่สร้างขึ้นด้วยคำสั่ง **netstat**:

```
# netstat -I en0 5
    input    (en0)     output           input   (Total)    output
 packets  errs  packets  errs colls  packets  errs  packets  errs colls
 8305067     0  7784711     0     0 20731867     0 20211853     0     0
       3     0        1     0     0        7     0        5     0     0
      24     0      127     0     0       28     0      131     0     0
CTRL C
```

โปรดจำไว้ว่า รายงานอันดับแรกจากคำสั่ง **netstat** จะแสดงกิจกรรมที่สะสมไว้ตั้งแต่การบูตระบบครั้งล่าสุด รายงานที่สองจะแสดงกิจกรรม สำหรับช่วงเวลา 5 วินาทีแรก

อ็อพชันคำสั่ง **netstat** อื่นๆ ที่มีประโยชน์คือ **-s** และ **-v** สำหรับรายละเอียด โปรดดู "คำสั่ง netstat" ในหน้า 328

การมอนิเตอร์ผลการทำงานอย่างต่อเนื่องด้วยคำสั่ง sar

คำสั่ง **sar** มีประโยชน์ในการพิจารณาการใช้ CPU

ต่อไปนี้คือตัวอย่างรายงานที่สร้างขึ้นด้วยคำสั่ง **sar** :

```
# sar -P ALL 5 2

AIX aixhost 2 5 00040B0F4C00    01/29/04
```

```
10:23:15 cpu    %usr    %sys    %wio   %idle
10:23:20  0        0       0       1      99
          1        0       0       0     100
          2        0       1       0      99
          3        0       0       0     100
          -        0       0       0      99
10:23:25  0        4       0       0      96
          1        0       0       0     100
          2        0       0       0     100
          3        3       0       0      97
          -        2       0       0      98

Average   0        2       0       0      98
          1        0       0       0     100
          2        0       0       0      99
          3        1       0       0      99
          -        1       0       0      99
```

คำสั่ง **sar** ไม่ได้รายงานกิจกรรมสะสม ตั้งแต่การบูตระบบครั้งล่าสุด

สำหรับรายละเอียดเกี่ยวกับคำสั่ง **sar** โปรดดู “คำสั่ง sar” ในหน้า 115 และ “การประเมินค่าผลการทำงานดิสก์ด้วยคำสั่ง sar” ในหน้า 201

การมอนิเตอร์ผลการทำงานของระบบอย่างต่อเนื่องด้วยคำสั่ง topas

คำสั่ง **topas** จะรายงานข้อมูลสถิติที่สำคัญ เกี่ยวกับกิจกรรมบนระบบโลคัล เช่น ขนาดของหน่วยความจำที่ใช้จริง และจำนวนการเรียกระบบสำหรับการเขียน

คำสั่ง **topas** จะใช้ไลบรารี curses เพื่อแสดงเอาต์พุตในรูปแบบที่เหมาะสมสำหรับการดูบนจอแสดงผลแบบอิงอักขระ 80x25 หรือในหน้าต่างที่มีขนาดเหมือนกับบนจอแสดงผลแบบกราฟิก คำสั่ง **topas** จะแตกและแสดงข้อมูลสถิติจากระบบที่มีช่วงเวลาดีฟอลต์ สองวินาที คำสั่ง **topas** นำเสนอหน้าจอสำรองต่อไปนี้ :

- ข้อมูลสถิติของระบบโดยรวม
- รายการของการประมวลผล busiest
- ข้อมูลสถิติ WLM
- รายการของฟิสิคัลดิสก์ที่นิยม
- การแสดงผลโลจิคัลพาร์ติชัน
- มุมมองข้ามพาร์ติชัน

ชุดไฟล์ `bos.perf.tools` และชุดไฟล์ `perfagent.tools` ต้องถูกติดตั้งอยู่บนระบบที่รันคำสั่ง **topas**

สำหรับข้อมูลเพิ่มเติมเกี่ยวกับคำสั่ง **topas** โปรดดู คำสั่ง topas ใน *Commands Reference, Volume 5*

จอภาพสถิติระบบโดยรวม

เอาต์พุตของจอภาพสถิติระบบโดยรวมประกอบด้วย ส่วนคงที่หนึ่งส่วนและส่วนผันแปรหนึ่งส่วน

สองบรรทัดบนสุดที่ด้านซ้ายของเอาต์พุตแสดงชื่อของระบบที่โปรแกรม **topas** กำลังรันอยู่ วันที่และเวลาของการสังเกตครั้งล่าสุด และช่วงเวลาการมอนิเตอร์ ข้างล่างส่วนนี้คือ ส่วนผันแปรซึ่งแสดงรายการส่วนย่อยต่อไปนี้:

- การใช้ประโยชน์ CPU
- อินเตอร์เฟสเครือข่าย
- ฟิสิคัลดิสก์
- WLM คลาส
- กระบวนการ

ทางด้านขวาของส่วนนี้คือส่วนคงที่ซึ่งประกอบด้วย ส่วนย่อยของสถิติดังต่อไปนี้:

- EVENTS/QUEUES
- FILE/TTY
- PAGING
- MEMORY
- PAGING SPACE
- NFS

ข้อมูลต่อไปนี้คือเอาต์พุตตัวอย่างของจอภาพสถิติระบบ โดยรวม:

```
Topas Monitor for host:    aixhost             EVENTS/QUEUES    FILE/TTY
Wed Feb  4 11:23:41 2004   Interval:  2        Cswitch      53  Readch     6323
                                               Syscall     152  Writech     431
Kernel    0.0   |                           |  Reads         3  Rawin         0
User      0.9   |                           |  Writes        0  Ttyout        0
Wait      0.0   |                           |  Forks         0  Igets         0
Idle     99.0   |###########################|  Execs         0  Namei        10
                                               Runqueue    0.0  Dirblk        0
Network  KBPS   I-Pack  O-Pack   KB-In  KB-Out  Waitqueue   0.0
en0       0.8     0.4     0.9     0.0     0.8
lo0       0.0     0.0     0.0     0.0     0.0  PAGING           MEMORY
                                               Faults        2  Real,MB    4095
Disk    Busy%     KBPS     TPS KB-Read KB-Writ  Steals        0  % Comp      8.0
hdisk0    0.0      0.0     0.0     0.0     0.0  PgspIn        0  % Noncomp  15.8
hdisk1    0.0      0.0     0.0     0.0     0.0  PgspOut       0  % Client   14.7
                                               PageIn        0
WLM-Class (Active)     CPU%    Mem%  Disk-I/O%  PageOut       0  PAGING SPACE
System                    0       0         0   Sios          0  Size,MB     512
Shared                    0       0         0                    % Used      1.2
Default                   0       0         0   NFS (calls/sec)  % Free     98.7
Name            PID CPU% PgSp Class         0   ServerV2       0
topas         10442  3.0  0.8 System            ClientV2       0   Press:
ksh           13438  0.0  0.4 System            ServerV3       0   "h" for help
gil            1548  0.0  0.0 System            ClientV3       0   "q" to quit
```

ยกเว้นสำหรับส่วนย่อยกระบวนการผันแปร คุณสามารถเรียงลำดับส่วนย่อย ทั้งหมดตามคอลัมน์ได้ โดยการเลื่อนเคอร์เซอร์ไปที่ด้านบนของคอลัมน์ ที่ต้องการ ส่วนย่อยผันแปรทั้งหมด ยกเว้นส่วนย่อยกระบวนการ มีมุมมองดังต่อไปนี้:

- รายการของผู้ใช้รีซอร์สสูงสุด
- รายงานบรรทัดเดียวที่แสดงถึงผลรวมของกิจกรรม

ตัวอย่างเช่น มุมมองรายงานบรรทัดเดียวอาจแสดงเฉพาะดิสก์ทั้งหมด หรือผลผลิตเครือข่ายทั้งหมด

สำหรับส่วนย่อย CPU คุณสามารถเลือกรายการของตัวประมวลผลที่ยุ่ง หรือการใช้ประโยชน์ CPU สากล อย่างใดอย่างหนึ่ง ดังแสดงในตัวอย่างข้างบน

รายการของการประมวลผล busiest บนจอแสดงผลของการมอนิเตอร์ topas

หากต้องการดูจอแสดงผลที่แสดงการประมวลผล busiest ให้ใช้แฟล็ก **-P** ของคำสั่ง **topas**

จอแสดงผลนี้จะคล้ายกับส่วนย่อยการประมวลผลของจอแสดงผลข้อมูลสถิติของระบบโดยรวม แต่แสดงพร้อมกับรายละเอียดเพิ่มเติม คุณสามารถเรียงลำดับจอแสดงผลนี้ได้ตามคอลัมน์ใดๆ โดยย้ายเคอร์เซอร์ไปยังด้านบนสุดของคอลัมน์ที่ต้องการ ต่อไปนี้คือตัวอย่างของเอาต์พุตของจอแสดงผลสำหรับการประมวลผล busiest:

```
Topas Monitor for host:    aixhost     Interval:   2    Wed Feb  4 11:24:05 2004

                                 DATA  TEXT  PAGE              PGFAULTS
USER       PID    PPID PRI NI    RES   RES SPACE    TIME CPU%  I/O  OTH COMMAND
root         1       0  60 20    202     9   202    0:04  0.0  111 1277 init
root       774       0  17 41      4     0     4    0:00  0.0    0    2 reaper
root      1032       0  60 41      4     0     4    0:00  0.0    0    2 xmgc
root      1290       0  36 41      4     0     4    0:01  0.0    0  530 netm
root      1548       0  37 41     17     0    17    1:24  0.0    0   23 gil
root      1806       0  16 41      4     0     4    0:00  0.0    0   12 wlmsched
root      2494       0  60 20      4     0     4    0:00  0.0    0    6 rtcmd
root      2676       1  60 20     91    10    91    0:00  0.0   20 6946 cron
root      2940       1  60 20    171    22   171    0:00  0.0   15  129 errdemon
root      3186       0  60 20      4     0     4    0:00  0.0    0  125 kbiod
root      3406       1  60 20    139     2   139    1:23  0.0 1542187 syncd
root      3886       0  50 41      4     0     4    0:00  0.0    0    2 jfsz
root      4404       0  60 20      4     0     4    0:00  0.0    0    2 lvmbb
root      4648       1  60 20     17     1    17    0:00  0.0    1   24 sa_daemon
root      4980       1  60 20     97    13    97    0:00  0.0   37  375 srcmstr
root      5440       1  60 20     15     2    15    0:00  0.0    7   28 shlap
root      5762       1  60 20      4     0     4    0:00  0.0    0    2 random
root      5962    4980  60 20     73    10    73    0:00  0.0   22  242 syslogd
root      6374    4980  60 20     63     2    63    0:00  0.0    2  188 rpc.lockd
root      6458    4980  60 20    117    12   117    0:00  0.0   54  287 portmap
```

จอภาพสถิติ WLM ของการมอนิเตอร์ topas

เมื่อต้องการดูจอภาพที่แสดงสถิติ WLM ให้ใช้แฟล็ก **-W** ของ คำสั่ง **topas**

จอภาพแบ่งออกเป็นส่วนดังต่อไปนี้:

- ส่วนด้านบนคือรายการของ WLM คลาสที่ยุ่งที่สุด ดังแสดงในส่วนย่อย WLM ของจอภาพสถิติระบบโดยรวม ซึ่งคุณยังสามารถเรียงลำดับ ตามคอลัมน์ใดก็ได้
- ส่วนที่สองของจอภาพนี้คือรายการของกระบวนการที่ใช้บ่อยภายใน WLM คลาสซึ่งคุณเลือกโดยใช้ปุ่มลูกศรหรือปุ่ม *f*

ข้อมูลต่อไปนี้เป็นตัวอย่างของรายงานเต็มจอภาพ WLM:

```
Topas Monitor for host:    aixhost     Interval:   2    Wed Feb  4 11:24:29 2004
WLM-Class (Active)              CPU%      Mem%     Disk-I/O%
System                           0         0           0
Shared                           0         0           0
Default                          0         0           0
Unmanaged                        0         0           0
```

```
Unclassified                    0         0            0
```

```
================================================================================
                                 DATA  TEXT  PAGE              PGFAULTS
USER       PID    PPID PRI NI    RES   RES SPACE    TIME CPU%  I/O  OTH COMMAND
root         1       0  60 20    202     9   202    0:04  0.0    0    0 init
root       774       0  17 41      4     0     4    0:00  0.0    0    0 reaper
root      1032       0  60 41      4     0     4    0:00  0.0    0    0 xmgc
root      1290       0  36 41      4     0     4    0:01  0.0    0    0 netm
root      1548       0  37 41     17     0    17    1:24  0.0    0    0 gil
root      1806       0  16 41      4     0     4    0:00  0.0    0    0 wlmsched
root      2494       0  60 20      4     0     4    0:00  0.0    0    0 rtcmd
root      2676       1  60 20     91    10    91    0:00  0.0    0    0 cron
root      2940       1  60 20    171    22   171    0:00  0.0    0    0 errdemon
root      3186       0  60 20      4     0     4    0:00  0.0    0    0 kbiod
```

การดูจอภาพฟิสิคัลดิสก์

เมื่อต้องการดูจอภาพที่แสดงรายการของฟิสิคัลดิสก์ที่ใช้บ่อย ให้ใช้แฟล็ก **–D** พร้อมกับคำสั่ง **topas**

จำนวนสูงสุดของฟิสิคัลดิสก์ที่แสดงขึ้นคือจำนวนของฟิสิคัลดิสก์ ที่ใช้บ่อยซึ่งกำลังมีการมอนิเตอร์ตามที่ระบุด้วยแฟล็ก **–d** รายการของ ฟิสิคัลดิสก์ที่ใช้บ่อยมีการเรียงลำดับตามฟิลด์ KBPS

ข้อมูลต่อไปนี้เป็นตัวอย่างของรายงานที่สร้างขึ้นโดยคำสั่ง **topas –D**:

```
Topas Monitor for host:    aixcomm    Interval:   2    Fri Jan 13 18:00:16 XXXX
================================================================================
Disk     Busy%  KBPS     TPS   KB-R   ART   MRT   KB-W   AWT   MWT   AQW    AQD
hdisk0     3.0  56.0     3.5   0.0    0.0   5.4   56.0   5.8   33.2  0.0    0.0
cd0        0.0   0.0     0.0   0.0   0.0    0.0   0.0    0.0    0.0  0.0    0.0
```

หากต้องการข้อมูลเพิ่มเติมเกี่ยวกับคำสั่ง **topas–D** ให้ดู คำสั่ง topas ใน *Commands Reference, Volume 5*

การดูพาเนล Cross-Partition

เมื่อต้องการดูสถิติ cross–partition ใน **topas** ให้ใช้แฟล็ก **–C** ด้วยคำสั่ง **topas** หรือกดปุ่ม **C** จากพาเนลอื่น

จอภาพแบ่งออกเป็นส่วนดังต่อไปนี้:

- ส่วนด้านบนแสดงข้อมูลรวมจากชุดพาร์ติชันเพื่อแสดง พาร์ติชันโดยรวม หน่วยความจำ และกิจกรรมตัวประมวลผล ปุ่ม **G** สลับส่วนนี้ระหว่างรายการแบบย่อ รายการรายละเอียด และปิด
- ส่วนด้านล่างแสดงสถิติสำหรับแต่ละพาร์ติชัน ซึ่งมีการแบ่งย่อยอีกเป็น สองส่วนคือ: พาร์ติชันแบบแบ่งใช้และพาร์ติชันเฉพาะ ปุ่ม **S** สลับส่วนพาร์ติชันแบบแบ่งใช้ระหว่างเปิดและ ปิด ปุ่ม **D** สลับส่วนพาร์ติชันเฉพาะระหว่างเปิดและ ปิด

ข้อมูลต่อไปนี้เป็นตัวอย่างเต็มจอภาพของเอาต์พุตจากคำสั่ง **topas –C**:

```
Topas CEC Monitor            Interval:  10           Wed Mar  6 14:30:10 XXXX
Partitions     Memory (GB)              Processors
Shr:  4        Mon:  24  InUse:  14     Mon:  8  PSz:  4  Shr_PhysB:  1.7
Ded:  4        Avl:  24                 Avl:  8  APP:  4  Ded_PhysB:  4.1
```

```
Host         OS  M Mem InU Lp  Us Sy Wa Id  PhysB  Ent  %EntC Vcsw PhI
-------------------------------shared-------------------------------------------
ptools1      A53 u 1.1 0.4  4  15  3  0 82   1.30  0.50  22.0  200   5
ptools5      A53 U  12  10  1  12  3  0 85   0.20  0.25   0.3  121   3
ptools3      A53 C 5.0 2.6  1  10  1  0 89   0.15  0.25   0.3   52   2
ptools7      A53 c 2.0 0.4  1   0  1  0 99   0.05  0.10   0.3  112   2
-------------------------------dedicated----------------------------------------
ptools4      A53 S 0.6 0.3  2  12  3  0 85   0.60
ptools6      A52   1.1 0.1  1  11  7  0 82   0.50
ptools8      A52   1.1 0.1  1  11  7  0 82   0.50
ptools2      A52   1.1 0.1  1  11  7  0 82   0.50
```

พาร์ติชันสามารถเรียงลำดับตามคอลัมน์ใดๆ ยกเว้นโฮสต์, OS, และ M, โดยการเลื่อนเคอร์เซอร์ไปที่ด้านบนของคอลัมน์ที่ต้องการ

หากต้องการข้อมูลเพิ่มเติมเกี่ยวกับคำสั่ง **topas -C** ให้ดู คำสั่ง topas ใน *Commands Reference, Volume 5*

การดูข้อมูลระดับพาร์ติชันของโลคัลโลจิคัล

เมื่อต้องการดูข้อมูลระดับพาร์ติชันและเมตริกซ์ประสิทธิภาพต่อตัวประมวลผลโลจิคัล ให้ใช้แฟล็ก **-L** พร้อมกับคำสั่ง **topas** หรือกดปุ่ม **L** จากพาเนลอื่น

จอภาพแบ่งออกเป็นสองส่วนดังนี้:

- ส่วนด้านบนแสดงชุดย่อยของข้อมูลระดับพาร์ติชัน
- ส่วนด้านล่างแสดงรายการที่จัดลำดับของเมตริกซ์ตัวประมวลผลโลจิคัล

ข้อมูลต่อไปนี้เป็นตัวอย่างของเอาต์พุตจากคำสั่ง **topas -L**:

```
    Interval:   2           Logical Partition: aix       Sat Mar 13 09:44:48 XXXX
    Poolsize: 3.0                Shared SMT ON            Online Memory:   8192.0
    Entitlement: 2.5              Mode: Capped            Online Logical CPUs:  4
                                                          Online Virtual CPUs:  2
    %user  %sys  %wait  %idle physc  %entc %lbusy   app   vcsw  phint %hypv hcalls
     47.5  32.5    7.0   13.0   2.0   80.0  100.0   1.0    240    150   5.0   1500
    ================================================================================
    logcpu minpf majpf  intr   csw  icsw runq lpa  scalls usr sys  wt idl  pc lcsw
    cpu0    1135   145   134    78    60    2  95   12345  10  65  15  10 0.6  120
    cpu1     998   120   104    92    45    1  89    4561   8  67  25   0 0.4  120
    cpu2    2246   219   167   128    72    3  92   76300  20  50  20  10 0.5  120
    cpu3    2167   198   127    62    43    2  94    1238  18  45  15  22 0.5  120
```

หากต้องการข้อมูลเพิ่มเติมเกี่ยวกับคำสั่ง **topas-L** ให้ดู คำสั่ง topas ใน *Commands Reference, Volume 5*

SMIT พาเนลสำหรับ topas/topasout/topasrec

SMIT พาเนลใช้เพื่อช่วยให้การตั้งค่าคอนฟิกและ การตั้งค่าของฟังก์ชันการบันทึก **topas** และการสร้างรายงานง่ายขึ้น

เมื่อต้องการไปยัง **topas** smit พาเนล ให้พิมพ์ `smitty performance` (หรือ `smitty topas`) และ เลือก **Configure Topas options**

เมนู Configure Topas Options แสดง:

```
                    Configure Topas Options
Move cursor to desired item and press Enter

 Add Host to topas external subnet search file (Rsi.hosts)
 List hosts in topas external subnet search file (Rsi.hosts)
 List active recordings
 Start new recording
 Stop recording
 List completed recordings
 Generate Report
 Setup Performance Management
```

หากต้องการข้อมูลเพิ่มเติม ให้ดู คำสั่ง topas ใน *Commands Reference, Volume 5*

การเพิ่มโฮสต์ให้กับไฟล์การค้นหา subnet topas ภายนอก (Rsi.hosts):

ไคลเอ็นต์ PTX และคำสั่ง **topas −C**|**topasrec −C** ถูกจำกัดอยู่ในตำแหน่งที่ Remote Statistics Interface (Rsi) API ใช้เพื่อระบุโฮสต์แบบรีโมต

เมื่อใดก็ตามที่ไคลเอ็นต์สตาร์ท ไคลเอ็นต์จะกระจายเคียวรี่บนพอร์ต **xmquery** ซึ่งลงทะเบียนเซอร์วิสของ **inetd** daemon โฮสต์แบบรีโมตจะมองเห็นเคียวรี่นี้ และไฟล์ `inetd.conf` จะถูกปรับแต่งเพื่อเริ่มต้น **xmservd** หรือ **xmtopas** daemons และตอบกลับไปยังไคลเอ็นต์ที่เคียวรี่ สถาปัตยกรรมที่มีอยู่จะจำกัดการเรียก **xmquery** ให้อยู่ภายในโฮสต์ที่ตั้งอยู่บน subnet เดียวกันกับระบบที่ทำเคียวรี่

หากต้องการหลีกเลี่ยงปัญหานี้ PTX จะสนับสนุนรายชื่อโฮสต์ที่ผู้ใช้กำหนดเองเสมอ ซึ่งจะอยู่ภายนอก subnet RSi จะอ่านรายชื่อโฮสต์นี้ (ไฟล์ `RSi.hosts`) และเลือกชื่อโฮสต์หรือ IP ที่แสดงใดๆ ได้โดยตรง คุณสามารถปรับแต่งไฟล์ `RSi.hosts` ได้เอง ตามค่าดีฟอลต์ RSi จะค้นหาตำแหน่งต่อไปนี้ตามลำดับก่อนหลัง:

1. `$HOME/Rsi.hosts`
2. `/etc/perf/Rsi.hosts`
3. `/usr/lpp/perfmgr/Rsi.hosts`

รูปแบบไฟล์นี้จะแสดงหนึ่งโฮสต์ต่อหนึ่งบรรทัด ตามรูปแบบอินเตอร์เน็ตแอดเดรสหรือชื่อโฮสต์ที่ผ่านการรับรองโดยสมบูรณ์ ดังตัวอย่างต่อไปนี้:

```
ptoolsl1.austin.ibm.com
9.3.41.206
...
```

เลือกอ็อพชัน **เพิ่มโฮสต์ให้กับไฟล์การค้นหา subnet topas ภายนอก (Rsi.hosts)** เพื่อเพิ่มโฮสต์ให้กับไฟล์ **Rsi.hosts**
เลือกอ็อพชัน **แสดงโฮสต์ในไฟล์การค้นหา subnet topas ภายอก (Rsi.hosts)** เพื่อดูรายการอ็อพชันในไฟล์ `Rsi.hosts`

การเริ่มต้นการบันทึกใหม่:

ใช้ **Start new recordings** เพื่อเริ่มต้นการบันทึก CEC/ถาวรแบบโลคัล/ไม่ถาวรตามข้อมูลอินพุตที่ผู้ใช้เลือกไว้ ผู้ใช้จะเห็นเมนูแยกต่างหาก สำหรับการเริ่มต้นการบันทึก CEC/ถาวรแบบโลคัล/ไม่ถาวร

การบันทึกถาวร:

การบันทึกถาวร คือการบันทึกที่เริ่มต้นจาก SMIT ซึ่งมีอ็อพชันการระบุการตัดและการเก็บรักษาไว้ คุณสามารถระบุจำนวนวันของการบันทึกที่จะจัดเก็บสำหรับแต่ละ ไฟล์บันทึก (ตัด) และจำนวนวันของการบันทึกที่จะเก็บรักษาไว้ (การเก็บรักษา) ก่อนที่จะสามารถลบไฟล์นั้นได้ สามารถรันได้ไม่เกินหนึ่งอินสแตนซ์ ของ **การบันทึกถาวร** ของการบันทึกชนิดเดียวกัน (CEC หรือโลคัล) ในระบบ เมื่อ **การบันทึกถาวร** เริ่มต้นขึ้น คำสั่งการบันทึกจะถูกเรียกใช้พร้อมกับอ็อพชันที่ผู้ใช้ระบุ ชุดเดียวกันของอ็อพชันบรรทัดคำสั่งที่ใช้โดยการบันทึกถาวรนี้ จะถูกเพิ่มลงในรายการ **inittab** ซึ่งจะทำให้มั่นใจว่า การบันทึกเริ่มต้นขึ้นโดยอัตโนมัติเมื่อรีบูตหรือรีสตาร์ท ระบบ

พิจารณาระบบที่กำลังรัน **การบันทึกถาวร แบบโลคัล** (รูปแบบการบันทึกไบนารีหรือ nmon) อยู่แล้ว ถ้าคุณต้องการเริ่มต้น **การบันทึกถาวร** ใหม่ของการบันทึกไบนารีแบบโลคัล การบันทึกถาวร ที่มีอยู่ต้องถูกหยุดก่อนโดยใช้อ็อพชัน **หยุดการบันทึกถาวร** ที่มีอยู่ภายใต้อ็อพชัน **หยุด การบันทึก** จากนั้น **การบันทึกถาวรแบบโลคัล** ต้องเริ่มต้น จากอ็อพชัน **เริ่มต้นการบันทึกถาวรแบบโลคัล** การเริ่มต้น **การบันทึกถาวร** จะล้มเหลวถ้าการบันทึกถาวรของรูปแบบ การบันทึกเดียวกันกำลังรันอยู่แล้วในระบบ เนื่องจาก **การบันทึก ถาวร** เพิ่มรายการ **inittab** เฉพาะผู้ใช้ที่ได้รับสิทธิเท่านั้น จะได้รับอนุญาตให้เริ่มต้น **การบันทึกถาวร** ได้

ตัวอย่างเช่น ถ้าจำนวนวันที่จะจัดเก็บสำหรับแต่ละไฟล์คือ **n** ไฟล์หนึ่งไฟล์จะมีจำนวนวันสูงสุดของการบันทึกเป็น **n** วัน ถ้าการบันทึกนานเกินกว่า **n** วัน ไฟล์ใหม่จะถูกสร้างขึ้น และการบันทึกในเวลาต่อมาทั้งหมดจะถูกจัดเก็บไว้ในไฟล์ใหม่ ถ้าจำนวนวันที่จะจัดเก็บสำหรับแต่ละไฟล์คือ **0** การบันทึกจะถูก บันทึกลงในไฟล์เดียวเท่านั้น ถ้าจำนวนวันที่จะเก็บรักษาคือ **m** ระบบจะเก็บรักษาไฟล์การบันทึกที่มีการบันทึกข้อมูลภายในช่วงเวลา **m** วันล่าสุด ไฟล์การบันทึกที่สร้างขึ้นโดย อินสแตนซ์การบันทึกเดียวกันของคำสั่ง **topasrec** ซึ่งมีข้อมูล ที่บันทึกไว้ก่อนหน้า **m** วันจะถูกลบออก

ค่าดีฟอลต์สำหรับจำนวนวันที่จะจัดเก็บสำหรับแต่ละไฟล์คือ **1**

ค่าดีฟอลต์สำหรับจำนวนวันที่จะเก็บรักษาไว้คือ **7**

เมนู อ็อพชัน SMIT สำหรับเริ่มต้นการบันทึก แสดง:

```
                SMIT options for Start Recording

                      Start Recording
Move cursor to desired item and press Enter.

Start Persistent local Recording
Start Persistent CEC Recording
Start Local Recording
Start CEC Recording
```

การเริ่มต้นการบันทึกถาวรแบบโลคัล:

ผู้ใช้สามารถเลือกชนิดของการบันทึกไบนารี หรือ nmon ถาวรแบบโลคัล

เมื่อต้องการเริ่มต้นการบันทึกตามลำดับ ให้เลือกไบนารีหรือ nmon บนเมนู Type of Persistent Recording:

```
              Type of Persistent Recording

Move cursor to desired item and press Enter.

   binary
```

```
   nmon


F1=Help                 F2=Refresh              F3=Cancel
```

ถ้าคุณเลือกรายงานที่เป็นชนิดไบนารี รายงานจะแสดงเป็น:

```
   พิมพ์หรือเลือกค่าในฟิลด์ที่ต้องป้อนข้อมูล
   กด Enter หลังทำการเปลี่ยนแปลงที่ต้องการทั้งหมดเสร็จ

    Type of Recording                                    binary
    Length of Recording                                 persistent
 Recording Interval in seconds               [300]
    * Number of Days to store per file      [1]
    * Number of Days to retain                      [7]
 Output Path                                              []
 * Overwrite existing recording file     no
 * Enable WLE                                           no
 * Include Disk Basic Metrics                 [Yes/No]
 * Include Service Time Metrics              [Yes/No]
 * Include Disk Adapter Basic Metrics  [yes/No]
 * Include Disk Adapter Service Time Metrics  [Yes/No]
```

ช่วงเวลาการบันทึก (ในหน่วยวินาที) ควรเป็นผลคูณของ 60 ถ้าชนิดการบันทึกเป็นการบันทึกไบนารีแบบโลคัล ผู้ใช้มี อ็อพชันในการเปิดใช้งานการสร้างรายงาน IBM Workload Estimator (WLE) ในจอภาพ SMIT รายงาน WLE จะมีการสร้างขึ้นเฉพาะในวันอาทิตย์เวลา 00:45 a.m. และจำเป็นที่การบันทึกไบนารีแบบโลคัลต้องเปิดใช้งานอยู่เสมอ เพื่อให้ได้ข้อมูลที่สม่ำเสมอกันในรายงาน ข้อมูลในรายงานประจำสัปดาห์ ถูกต้องเฉพาะถ้ามีการเปิดใช้งานการบันทึกแบบโลคัลเสมอเท่านั้น

รายงาน WLE ที่สร้างขึ้นถูกจัดเก็บไว้ในไฟล์ **/etc/perf/<hostname>_aixwle_weekly.xml** ตัวอย่างเช่น ถ้าชื่อโฮสต์เป็น **ptoolsl1** รายงานประจำสัปดาห์ จะมีการบันทึกลงในไฟล์ **/etc/perf/ptoolsl1_aixwle_weekly.xml**

หากต้องการข้อมูลเพิ่มเติม ให้อ้างอิง:

- “การบันทึกถาวร” ในหน้า 24
- ตัวกรอง nmon ที่มีอยู่

การเริ่มต้นการบันทึก CEC ถาวร:

ใช้ **start persistent CEC recording** เพื่อเริ่มต้น การบันทึกถาวรของ CEC **Final recording** ที่เริ่มต้น จะขึ้นอยู่กับอินพุตที่ให้ที่จอภาพในเวลาต่อมา จอภาพอินพุต จะมีการโหลดพร้อมกับค่าดีฟอลต์เมื่อเริ่มต้น

ช่วงเวลาการบันทึก (ในหน่วยวินาที) ควรเป็นผลคูณของ 60

หากต้องการข้อมูลเพิ่มเติม ให้อ้างอิง “การบันทึกถาวร” ในหน้า 24

การเริ่มต้นการบันทึกแบบโลคัล:

ใช้ **start local recording** เพื่อเริ่มต้น การบันทึกแบบโลคัลตามข้อมูลอินพุตที่ให้ที่จอภาพในเวลาต่อมา ผู้ใช้สามารถเลือกระหว่างไบนารีหรือ nmon, และเลือกวัน ชั่วโมง หรือ กำหนดเองเพื่อเริ่มต้นการบันทึกตามลำดับ

```
                          Type of Recording

Move cursor to desired item and press Enter.
  binary
  nmon

F1=Help                F2=Refresh             F3=Cancel
```

หลังจากเลือกไบนารีหรือ nmon แล้ว ผู้ใช้ต้องเลือกวัน ชั่วโมง หรือ กำหนดเองในจอภาพตัวเลือกถัดไป

```
                          Length of Recording

Move cursor to desired item and press Enter.
  day
  hour
  custom

F1=Help                F2=Refresh             F3=Cancel
F8=Image               F10=Exit               Enter=Do
```

สำหรับการบันทึกแบบวันหรือชั่วโมง ช่วงเวลาการบันทึกและจำนวนตัวอย่าง ไม่สามารถแก้ไขได้ สำหรับการบันทึกแบบกำหนดเอง ช่วงเวลาการบันทึกและจำนวนตัวอย่าง สามารถแก้ไขได้ ช่วงเวลาการบันทึกควรเป็นผลคูณของ 60 การใช้การบันทึกแบบกำหนดเองคือการรวบรวมเฉพาะจำนวนตัวอย่างที่ระบุ ณ ช่วงเวลาที่ระบุ และออกจากการบันทึก ถ้า มีการระบุจำนวนตัวอย่างเป็นศูนย์ การบันทึกจะยังคงรันต่อเนื่องไป จนกว่าถูกหยุด

ค่าที่โหลดไว้แล้วซึ่งแสดงในจอภาพเป็นค่าดีฟอลต์

หากต้องการข้อมูลเพิ่มเติม ให้อ้างอิง “การบันทึก NMON” ในหน้า 27

การเริ่มต้นการบันทึก CEC:

ใช้ **เริ่มต้นการบันทึก CEC** เพื่อเริ่มต้น การบันทึกสำหรับจอภาพ CEC ในลำดับต่อมา

ผู้ใช้ต้องเลือกระยะเวลาของการบันทึก (วัน ชั่วโมง หรือกำหนดเอง) เพื่อเริ่มต้นการบันทึกตามลำดับ

```
                          Length of Recording

    Move cursor to desired item and press Enter.
     day
     hour
     custom

F1=Help                F2=Refresh             F3=Cancel
```

สำหรับการบันทึกแบบวันหรือชั่วโมง ช่วงเวลาการบันทึกและจำนวนตัวอย่าง ไม่สามารถแก้ไขได้

สำหรับการบันทึกแบบกำหนดเอง ช่วงเวลาการบันทึกและจำนวนตัวอย่าง สามารถแก้ไขได้ และช่วงเวลาการบันทึกควรเป็นผลคูณของ 60 การใช้การบันทึกแบบกำหนดเองคือการรวบรวมเฉพาะจำนวนตัวอย่างที่ระบุ ณ ช่วงเวลาที่ระบุ และออกจากการบันทึก ถ้า มีการระบุจำนวนตัวอย่างเป็นศูนย์ การบันทึกจะยังคงรันต่อเนื่องไป จนกว่าถูกหยุด

การบันทึก NMON:

NMON มาพร้อมกับตัวกรองการบันทึกที่ช่วยคุณกำหนด **การบันทึก NMON** เองได้ คุณสามารถเลือกและยกเลิกการเลือกส่วนต่อไปนี้ของ **การบันทึก NMON**:

- JFS
- RAW เคอร์นัลและ LPAR
- กลุ่มวอลุ่ม
- พื้นที่ว่างการเพจ
- MEMPAGES
- NFS
- WLM
- หน้าขนาดใหญ่
- กระบวนการอีเทอร์เน็ตแบบแบ่งใช้ (สำหรับ VIOS)
- หน้าขนาดใหญ่และอะซิงโครนัส

หมายเหตุ: ดิสก์ต่อบรรทัด ไฟล์กลุ่มดิสก์ และดิสก์ที่ต้องการเป็นอ็อพชันที่เหมาะสม เฉพาะถ้ามีการรวมส่วนการตั้งค่าคอนฟิกดิสก์ไว้ในการบันทึก เท่านั้น อ็อพชันตัวกรองกระบวนการและ threshold กระบวนการเหมาะสม เฉพาะถ้ามีการรวมรายการกระบวนการไว้ในการบันทึกเท่านั้น

ตัวกรองกระบวนการและดิสก์จะถูกโหลดโดยอัตโนมัติด้วยอ็อพชันตัวกรอง ที่ใช้สำหรับการบันทึกครั้งล่าสุดโดยผู้ใช้รายเดียวกัน คุณสามารถ ระบุว่าจะเรียกใช้คำสั่ง **External** เมื่อเริ่มต้นหรือสิ้นสุด การบันทึก NMON ในโปรแกรมเริ่มต้นหรือสิ้นสุด ตัวรวบรวมข้อมูลภายนอก ถ้าคุณต้องการเรียกใช้คำสั่งภายนอก เป็นระยะๆ เพื่อบันทึกเมตริกซ์ สามารถระบุได้ที่สแน็ปโปรแกรมของ ตัวรวบรวมข้อมูลภายนอก หากต้องการรายละเอียดเกี่ยวกับการใช้คำสั่ง ภายนอกสำหรับการบันทึก nmon ให้อ้างอิง คำสั่ง nmon ใน *Commands Reference, Volume 4*

ระเบียบการตั้งชื่อ:

ไฟล์ที่บันทึกจะถูกจัดเก็บไว้ในไฟล์ที่ระบุดังแสดงในตัวอย่าง ต่อไปนี้:

- หากชื่อไฟล์มีไดเร็กทอรีและคำเสริมหน้าชื่อไฟล์ ไฟล์เอาต์พุตสำหรับการบันทึกไฟล์เดียวจะเป็นดังนี้:

สไตล์	**ไฟล์**
Local Nmon Style:	<ชื่อไฟล์>_YYMMDD_HHMM.nmon
Local Nmon Style:	<ชื่อไฟล์>_YYMMDD_HHMM.topas
Topas Style CEC:	<ชื่อไฟล์>_YYMMDD_HHMM.topas

- หากชื่อไฟล์มีไดเร็กทอรีและคำเสริมหน้าชื่อไฟล์ ไฟล์เอาต์พุตสำหรับการบันทึกหลายไฟล์ (การตัดและการรักษาไว้) จะเป็นดังนี้:

สไตล์	ไฟล์
Local Nmon Style:	<ชื่อไฟล์>_YYMMDD.nmon
Local Nmon Style:	<ชื่อไฟล์>_YYMMDD.topas
Topas Style CEC:	<ชื่อไฟล์>_CEC_YYMMDD.topas

- หากชื่อไฟล์มีไดเร็กทอรีและไม่มีคำเสริมหน้าชื่อไฟล์ ไฟล์เอาต์พุตสำหรับการบันทึกไฟล์เดียวจะเป็นดังนี้:

สไตล์	ไฟล์
Local Nmon Style:	<ชื่อไฟล์/ชื่อโฮสต์>_YYMMDD_HHMM.nmon
Local Nmon Style:	<ชื่อไฟล์/ชื่อโฮสต์>_YYMMDD_HHMM.topas
Topas Style CEC:	<ชื่อไฟล์/ชื่อโฮสต์>_CEC_YYMMDD_HHMM.topas

- หากชื่อไฟล์มีไดเร็กทอรีและไม่มีคำเสริมหน้าชื่อไฟล์ ไฟล์เอาต์พุตสำหรับการบันทึกหลายไฟล์ (การตัดและการรักษาไว้) จะเป็นดังนี้:

สไตล์	ไฟล์
Local Nmon Style:	<ชื่อไฟล์/ชื่อโฮสต์>_YYMMDD.nmon
Local Nmon Style:	<ชื่อไฟล์/ชื่อโฮสต์>_YYMMDD.topas
Topas Style CEC:	<ชื่อไฟล์/ชื่อโฮสต์>_CEC_YYMMDD.topas

การบันทึกสองครั้งของรูปแบบการบันทึกเดียวกันและมีค่าพารามิเตอร์ **ชื่อไฟล์** เหมือนกัน (ชื่อไฟล์ดีฟอลต์หรือที่ผู้ใช้ระบุ) ไม่สามารถเริ่มต้นพร้อมกันได้ เนื่องจากกระบวนการบันทึกสองกระบวนการนี้มี แนวโน้มที่จะบันทึกลงในไฟล์บันทึกเดียวกัน

ตัวอย่าง:

1. ผู้ใช้กำลังพยายามเริ่มต้นการบันทึกวันไบนารีโลคัลพร้อมด้วยเอาต์พุตพาธ ที่ระบุเป็น **/home/test/sample_bin** ถ้าไฟล์ที่บันทึกสร้างขึ้นในเวลา 12:05 ชั่วโมง วันที่ 10 มีนาคม 2008 และชื่อโฮสต์คือ ses15 ชื่อไฟล์เอาต์พุตจะเป็น **/home/test/sample_bin/ses15_080310_1205.topas**
2. สมมุติว่าผู้ใช้กำลังพยายามเริ่มต้นการบันทึก CEC ถาวร ด้วยอ็อพชันการตัดเป็น 2 และเอาต์พุตพาธมีการระบุเป็น **/home/test/sample** สมมุติว่าไฟล์ที่บันทึกสร้างขึ้นในเวลา 12:05 ชั่วโมง วันที่ 10 มีนาคม 2008 และชื่อโฮสต์คือ ses15 ชื่อไฟล์เอาต์พุต จะเป็น /home/test/sample_bin/ses15_cec_080310.topas หลังจากจัดเก็บนาน 2 วัน (เนื่องจากการตัด =2) นับจากการบันทึกในไฟล์นี้ ไฟล์ที่บันทึกไว้ซึ่งมีชื่อว่า **/home/test/sample_bin/ses15_cec_080312.topas** จะถูกสร้างขึ้นในวันที่ 12 มีนาคมเพื่อจัดเก็บข้อมูลของวันที่ 12 มีนาคมและ 13 มีนาคม

การหยุดการบันทึก:

ใช้คำสั่ง Stop recording เพื่อหยุดการบันทึกที่กำลังรันอยู่ในปัจจุบัน ผู้ใช้สามารถเลือกการบันทึกที่กำลังรันอยู่จากรายการและหยุดการบันทึกนั้น

จากเมนู คุณต้องเลือกชนิดของการบันทึก ที่จะหยุด หลังจากเลือกชนิดของการบันทึกแล้ว การบันทึกที่กำลังรันอยู่ในปัจจุบันจะแสดงรายการบนเมนู จากนั้น คุณสามารถเลือกการบันทึก ที่จะหยุดได้

ข้อมูลต่อไปนี้คือจอภาพสำหรับการเลือกชนิดของการบันทึกที่จะ หยุด:

```
                    Stop Recording

Stop persistent recording
Stop binary recording
Stop nmon recording
Stop CEC recording
```

หมายเหตุ: การบันทึกสามารถหยุดได้เฉพาะถ้าคุณมีสิทธิได้รับอนุญาตเบื้องต้น ในการหยุดกระบวนการบันทึกเท่านั้น

รายการบันทึกที่แอ็คทีฟ:

หากต้องการแสดงการบันทึกที่รันอยู่บนระบบในไดเร็กทอรีที่ผู้ใช้ระบุ ให้ใช้ **รายการบันทึกที่แอ็คทีฟ**

หากต้องการแสดงการบันทึกที่แอ็คทีฟ:

1. ให้ป้อนพาธของการบันทึก.
2. เลือกชนิดของการบันทึกที่ต้องการแสดง.

```
                        Type of Recording
Move cursor to desired item and press Enter.
persistent
binary
nmon
cec
all

F1=Help                 F2=Refresh              F3=Cancel
F8=Image                F10=Exit                Enter=Do
/=Find                  n=Find Next
```

สิ่งนี้จะแสดง **รูปแบบ เวลาเริ่มต้น** และ **พาธของเอาต์พุต** ของการบันทึกที่แอ็คทีฟและพาธที่ระบุ พาธของเอาต์พุต ของการบันทึกที่ยังคงอยู่จะนำหน้าด้วยเครื่องหมายดอกจัน (*) สำหรับการบันทึกแบบไบนารีบนโลคัลที่ยังคงอยู่ด้วยการเปิดใช้งาน WLE พาธของเอาต์พุจจะถูกนำหน้าด้วยเครื่องหมายแสดงจำนวน (#)

รายการที่บันทึกเสร็จสิ้นแล้ว:

ใช้ **รายการที่บันทึกเสร็จสิ้นแล้ว** เพื่อแสดงรายการของการบันทึกที่เสร็จสิ้นแล้วในพาธของไดเร็กทอรีที่ระบุเฉพาะผู้ใช้ การบันทึกที่เสร็จสิ้นแล้วเหล่านี้สามารถนำมาใช้ได้โดย **สร้างเมนูรายงาน** เพื่อสร้างไฟล์รายงาน

หากต้องการ **แสดงรายการที่บันทึกเสร็จสิ้นแล้ว** ให้ทำตามขั้นตอนเหล่านี้:

1. ป้อนพาธของการยันทึก พาธนี้เป็นพาธที่ใช้เพื่อหาตำแหน่ง ไฟล์บันทึก
2. เลือกชนิดของการบันทึกที่ต้องการใช้

```
persistent
binary
nmon
cec
all
```

ซึ่งจะแสดงรายการ **ชนิดของการบันทึก เวลาเริ่มต้น** และ **เวลาที่หยุด** ของการบันทึกที่เสร็จสิ้นแล้วในพาธที่ระบุ

การสร้างรายงานจากไฟล์บันทึกที่มีอยู่:

ใช้อ็อพชัน **สร้างรายงาน** เพื่อสร้างรายงานจากไฟล์การบันทึกที่มีอยู่ในพาธไดเร็กทอรีที่ระบุเฉพาะผู้ใช้

ถ้าพาธไดเร็กทอรีที่เลือกไว้คือ Persistent Recording เงื่อนไขต่อไปนี้จะเป็นจริง:

1. ถ้าการบันทึกที่คงอยู่กำลังรันอยู่ การบันทึกที่คงอยู่ที่กำลังรันอยู่ จะถูกเลือกสำหรับการสร้างรายงาน
2. ถ้าการบันทึกที่คงอยู่ไม่ได้รันอยู่ การบันทึกที่คงอยู่ล่าสุด จะถูกเลือกสำหรับการสร้างรายงาน

การใช้อ็อพชัน **สร้างรายงาน** จะพร้อมต์ให้คุณป้อนค่าของไฟล์การบันทึก รูปแบบการรายงาน เวลาเริ่มต้น เวลาสิ้นสุด และไฟล์หรือชื่อเครื่องพิมพ์ เพื่อสร้างรายงานตามการอินพุต

ปฏิบัติตามขั้นตอนต่อไปนี้ เพื่อสร้างรายงาน:

1. เลือกชื่อไฟล์หรือเครื่องพิมพ์ที่ส่งรายงาน :

```
                Send report to File/Printer

   Move cursor to desired item and press Enter.

   1 Filename
   2 Printer
```

2. เลือกพาธที่เป็นที่ตั้งไฟล์การบันทึก:

```
    Path to locate the recording file            [] +
```

3. เลือกรูปแบบการออกรายงาน (ตามชนิดของการบันทึก):

```
   * Reporting Format                            [] +
```

ต่อไปนี้คือตัวอย่างของชนิดรายงาน *ที่คั่นด้วยเครื่องหมายจุลภาค/สเปร็ดชีต* :

```
* Type of Recording                             []
* Reporting Format                              []
    Type                                        [mean] +
  Recording File name                           []
* Output File                                   []
```

The following is an example of a *nmon* report type:

```
* Type of Recording                             []
* Reporting Format                              []
  Recording File name                           []
* Output File                                   []
```

หมายเหตุ: ฟิลด์ **ไฟล์เอาต์พุต** เป็นฟิลด์บังคับสำหรับ *คั่นด้วยเครื่องหมายจุลภาค/สเปร็ดชีต* และชนิด *nmon* และตัวเลือกสำหรับรูปแบบรายการอื่นๆ ทั้งหมด การบันทึก **topas** จะสนับสนุนชนิดสำหรับรูปแบบรายงาน *คั่นด้วยเครื่องหมายจุลภาค* และ *สเปร็ดชีต*

ต่อไปนี้คือตัวอย่างของชนิดรายงาน *สรุป/ดิสก์ สรุป/รายละเอียด/สรุปเน็ตเวิร์ก* :

```
* Type of Recording                             []
* Reporting Format                              []
  Begin Time (YYMMDDHHMM)                       []
  End Time (YYMMDDHHMM)                         []
  Interval                                      []
  Recording File name                           []
  Output File (defaults to stdout)              []
```

สำหรับตัวอย่างข้างต้น ฟิลด์สองฟิลด์แรกไม่สามารถปรับเปลี่ยนได้ และกรอกค่าจากการเลือกก่อนหน้านี้

ถ้าเลือกเครื่องพิมพ์ไว้เป็นเอาต์พุตของรายงาน ฟิลด์ **ไฟล์เอาต์พุต** จะแทนที่ด้วยฟิลด์ **ชื่อเครื่องพิมพ์** ที่ต้องการ จากรายการเครื่องพิมพ์ที่ปรับแต่งในระบบ

```
* Printer Name                                  [ ] +
```

สำหรับคำอธิบายเกี่ยวกับรูปแบบการรายงานที่พร้อมใช้งานสำหรับ CEC และการบันทึกไบนารีโลคัล โปรดดู คำสั่ง topas in *Commands Reference, Volume 5*

การตั้งค่าการจัดการประสิทธิภาพ:

เมนูนี้ใช้เพื่อตั้งค่าและตั้งค่าคอนฟิกการจัดการประสิทธิภาพ

```
  Setup Performance Management

Move cursor to desired item and press Enter.

  Enable PM Data Transmission
  Disable PM Data Transmission
  Retransmit Recorded Data
  Change/Show Customer Information
  Change/Show Data Retention Period
  Change/Show Trending Days and Shift Timing
```

- **Enable PM Data Transmission**

  ใช้ **Enable PM Data Transmission** เพื่อ เปิดใช้งานการส่งผ่านข้อมูลประสิทธิภาพไปยัง IBM จาก Electronic Service Agent (ESA) หรือ Hardware Management Console (HMC)

- **Disable PM Data Transmission**

  ใช้ **Disable PM Data Transmission** เพื่อ ปิดใช้งานการส่งผ่านข้อมูลประสิทธิภาพไปยัง IBM

- **Retransmit Recorded Data**

  ใช้ **Retransmit Recorded Data** เพื่อ ส่งผ่านข้อมูลประสิทธิภาพที่บันทึกไว้ก่อนหน้านี้ใหม่

  ```
  Retransmit Recorded Data

  Type or select values in entry fields.
  Press Enter AFTER making all desired changes.

                                                [Entry Fields]
  * Enter the Date [YYYYMMDD]                   []                      #
  ```

  - ถ้าผู้ใช้ต้องการส่งผ่านข้อมูล PM ที่ลงวันที่ 12th Feb, 2009 ใหม่ให้ป้อน 20090212 ในกล่องข้อความ
  - ป้อน 0 เพื่อส่งผ่านไฟล์ข้อมูล PM ทั้งหมดที่บันทึกไว้และมีอยู่

- หลังจากป้อนวันที่แล้ว ระบบจะแสดงขั้นตอนที่ดำเนินการด้วยตนเองต่อไปนี้ เพื่อส่งข้อมูลไปยัง IBM โดยใช้ ESA หรือ HMC:

  ```
  Steps to do a Manual transmission from Electronic Service Agent on the HMC

  1.  Login to HMC
  2.  Select 'Service Management'
  3.  Select 'Transmit Service Information'
  4.  Click the 'Send' button labeled 'To transmit the performance management information immediately,
      click Send.  (the second Send button on the page)
  5.  Check the Console Events log for results

  Steps to do a Manual transmission from Electronic Service Agent for AIX

  1. Login to ESA using web interface (https://hostname:port/esa)
  2. Go to 'Service information'
  ```

```
3. Select action 'Collect information'
4. Select the Performance Management checkbox
5. Click OK
6. Check the Activity Log for results"
```

- **Change/Show Customer Information**

  ใช้ **Change/Show Customer Information** เพื่อแสดงหรืออัพเดตข้อมูลลูกค้า ข้อมูลลูกค้าจะถูกส่งไปยัง IBM ถ้าการส่งผ่านข้อมูล PM มีการเปิดใช้งาน

- **Change/Show Data Retention Period**

  ใช้ **Change/Show Data Retention Period** เพื่อแสดงหรือเปลี่ยนรอบเวลาการเก็บรักษา ข้อมูล รอบเวลาการเก็บไว้กำหนดว่าจะเก็บข้อมูลเก่าไว้ในไดเร็กทอรีข้อมูล นานเพียงใดก่อนข้อมูลนั้นจะถูกลบออก

- **Change/Show Trending Days and Shift Timing**

  ใช้ **Change/Show Trending Days and Shift Timing** เพื่อแสดง/อัพเดต Trending Days and Shift Timings

การเซ็ตอัพ Workload Estimator:

เมนูนี้ใช้เพื่อตั้งค่าและตั้งค่าคอนฟิก Workload Estimator

```
  WLE

Type or select values in the entry fields.
Press Enter AFTER making all desired changes.

  WLE Collection
  WLE input type
```

- **คอลเล็กชัน WLE**

  ใช้ **คอลเล็กชัน WLE** เพื่อเปิดใช้งานหรือ ปิดใช้งานการสร้างรายงานที่จะใช้เป็นอินพุตสำหรับ WLE ฟิลด์ แสดงสภาพปัจจุบันเป็น **คอลเล็กชัน WLE** โดยค่าดีฟอลต์ ควรปิดใช้งาน คอลเล็กชันก่อนคุณสามารถหยุดการจัดทำเร็กคอร์ดที่เชื่อมโยงได้

- **ชนิดอินพุต WLE**

  ใช้ **ชนิดอินพุต WLE** เพื่อตัดสินใจว่า ควรจัดทำรายงาน WLE จากข้อมูลการจัดทำเร็กคอร์ดโลคัลไบนารีที่กำลังรันอยู่ในปัจจุบัน หรือจากข้อมูลการจัดทำเร็กคอร์ดโลคัล **nmon** หมายเหตุว่าอ็อพชันนี้ใช้ได้สำหรับการจัดทำเร็กคอร์ดแบบต่อเนื่องเท่านั้น

การมอนิเตอร์ผลการทำงานของระบบอย่างต่อเนื่องโดยใช้เซอร์วิส Performance Management (PM)

เซอร์วิส Performance Management (PM) จะช่วยทำให้การเก็บสะสมเป็นแบบอัตโนมัติ การเก็บถาวร และการวิเคราะห์ข้อมูลผลการทำงานของระบบ และส่งคืนรายงานเพื่อช่วยลูกค้าจัดการกับรีซอร์สของระบบและความจุ ข้อมูลที่เก็บรวบรวมคือ การใช้ประโยชน์จากระบบ ข้อมูลผลการทำงาน และข้อมูลคอนฟิกูเรชันของฮาร์ดแวร์

Performance Management (PM) Data ที่เก็บรวบรวมจะถูกส่งไปยัง IBM ผ่าน Electronic Service Agent (ESA) หรือ Hardware Management Console (HMC) IBM จะเก็บข้อมูลสำหรับลูกค้าและจัดเตรียมข้อมูลเหล่านั้นด้วยชุดของรายงานและกราฟที่แสดงการโตและผลการทำงานของ เซิร์ฟเวอร์ ลูกค้าสามารถเข้าถึงรายงานแบบอิเล็กทรอนิกได้โดยใช้ บราวเซอร์ที่เป็นมาตรฐาน

เมื่อใช้กับ IBM Systems Workload Estimator การนำเสนอนี้จะอนุญาตให้ลูกค้าเข้าใจถึงแนวโน้มทางธุรกิจ ที่เกี่ยวข้องกับเวลาของการอัพเกรดฮาร์ดแวร์ที่จำเป็นได้ดีกว่า เช่น Central Processing Unit (CPU) หรือดิสก์ IBM Systems Workload Estimator สามารถคำนวณขนาดการรวมระบบ หรือประเมินการอัพเกรดระบบที่มีโลจิคัลพาร์ติชัน โดยการใช้ PM สำหรับ IBM Power Systems™ ส่งข้อมูลสำหรับหลายระบบหรือพาร์ติชันไปยัง IBM Systems Workload Estimator

Performance Management Service จะใช้การบันทึกแบบไบนารี **topasrec** ที่ยังคงอยู่เพื่อเก็บรวบรวมข้อมูลประสิทธิภาพการทำงาน ดังนั้น การบันทึกแบบไบนารี **topasrec** ที่มีอยู่ควรเปิดใช้งานสำหรับ PM Service อยู่เสมอ เพื่อเก็บรวบรวมข้อมูลประสิทธิภาพการทำงาน

Notes:

1. ด้วยการเปิดใช้งาน PM Service คุณยอมรับว่า IBM อาจใช้และแบ่งใช้ ข้อมูลที่รวบรวมโดย PM สำหรับเซิร์ฟเวอร์ IBM Power Systems ภายใน IBM enterprise โดยไม่มีข้อจำกัด ซึ่งรวมถึงวัตถุประสงค์ ของการกำหนดปัญหา เพื่อแนะนำคุณเกี่ยวกับประสิทธิภาพและการวางแผน ความสามารถ การรักษาความสัมพันธ์ทางธุรกิจใหม่หรือที่มีอยู่เดิมกับ IBM การแจ้งเตือนคุณถึงข้อจำกัดรีซอร์สที่มีอยู่หรือที่คาดการณ์ และแนะนำเราเพื่อปรับปรุงผลิตภัณฑ์ของ IBM คุณยังยอมรับว่าข้อมูลของคุณอาจถูกถ่ายโอนไปยัง entity ในประเทศใดๆ ที่ว่าประเทศนี้คือสมาชิกของ European Union หรือไม่ก็ตาม
2. คุณอาจให้สิทธิแก่ IBM เพื่อแบ่งใช้ข้อมูลกับกลุ่มบุคคลที่สามทั้งหลาย ซึ่งประกอบด้วยผู้ให้บริการแก้ปัญหาตั้งแต่หนึ่งรายขึ้นไป และพาร์ตเนอร์ทางธุรกิจ เพื่อทำให้พวกเขาตระหนักถึงผลการทำงานของคุณ และความสามารถที่ต้องการ และเปิดให้พวกเขาจัดเตรียมระดับการให้บริการ ที่สูงกว่า การให้สิทธิจะถูกกระทำเมื่อดูกราฟแบบออนไลน์

สำหรับข้อมูลเพิ่มเติมเกี่ยวกับ Performance Management services โปรดดูไฟล์ `README.perf.tools`

การวินิจฉัยผลการทำงานเริ่มต้น

มีหลายประเภทของปัญหาเกี่ยวกับผลการทำงานที่รายงานให้เห็นเพื่อนำมาพิจารณา ขณะที่วินิจฉัยปัญหาเกี่ยวกับผลการทำงาน

ชนิดของปัญหาประสิทธิภาพที่รายงาน

เมื่อมีการรายงานปัญหาประสิทธิภาพ การกำหนดประเภทของปัญหา ประสิทธิภาพโดยการจำกัดรายการของความเป็นไปได้เป็นสิ่งที่มีประโยชน์

โปรแกรมเฉพาะหนึ่งรันช้า

โปรแกรมอาจเริ่มรันช้าลงเนื่องจากเหตุผลอย่างใด อย่างหนึ่ง

แม้ว่าการรันช้านี้อาจดูเป็นเรื่องที่ไม่สำคัญมากนัก แต่ยังคงมีคำถามที่ต้อง ตอบดังนี้:

- โปรแกรมรันช้าเสมอหรือไม่?
  ถ้าโปรแกรมเพิ่งจะเริ่มรันช้าลง สาเหตุอาจมาจากการเปลี่ยนแปลงเมื่อเร็วๆ นี้
- มีการเปลี่ยนแปลงซอร์สโค้ดหรือการติดตั้งเวอร์ชันใหม่หรือไม่?
  ถ้าใช่ให้ตรวจสอบ กับโปรแกรมเมอร์หรือผู้ขาย
- บางสิ่งในสภาพแวดล้อมมีการเปลี่ยนแปลงหรือไม่?
  ถ้าไฟล์ที่โปรแกรมใช้ รวมถึงโปรแกรมที่ดำเนินการได้ของโปรแกรมเอง ถูกลบออก ขณะนี้โปรแกรมอาจ ประสบกับความล่าช้าของเครือข่ายที่ไม่เคยมีมาก่อน หรือไฟล์อาจกำลังถูกช่วงชิงสำหรับ ตัวเข้าถึงดิสก์เดียวที่เคยอยู่บนดิสก์อื่นก่อนหน้านี้

ถ้า ผู้ดูแลระบบเปลี่ยนพารามิเตอร์การปรับระบบ โปรแกรมอาจมีข้อจำกัดที่ไม่เคย พบมาก่อนหน้านี้ ตัวอย่างเช่น ถ้าผู้ดูแลระบบเปลี่ยนวิธีคำนวณระดับความสำคัญ โปรแกรมที่เคยรันได้ค่อนข้าง เร็วในพื้นหลังอาจทำงานช้าลงในขณะนี้ ในขณะที่โปรแกรมพื้นหน้าทำงานได้เร็วขึ้น

- โปรแกรมมีการบันทึกใน **perl, awk, csh,** หรือ ภาษาที่ต้องตีความอื่นบางอย่างหรือไม่?

  โชคไม่ดีนัก ภาษาที่ต้องตีความทำให้คอมไพเลอร์ ไม่สามารถทำงานได้อย่างเต็มที่ นอกจากนี้ ในภาษา เช่น **perl** หรือ **awk** การร้องขอ การดำเนินงานที่ต้องคำนวณหรือเน้น I/O มากยังสามารถทำได้ง่ายโดยใช้อักขระเพียงไม่กี่ตัวเท่านั้น ควรสอบถามข้อมูลจากฝ่ายสนับสนุนหรือเพื่อนที่มีความรู้เกี่ยวกับโปรแกรมดังกล่าว โดยเน้นไปที่จำนวนของการทำซ้ำที่ใช้โดยแต่ละ การดำเนินงาน
- โปรแกรมรันที่ความเร็วเท่ากันเสมอหรือรันเร็วขึ้นในบางครั้ง?

  ระบบไฟล์ใช้หน่วยความจำระบบบางอย่างเพื่อจัดเก็บหน้าของไฟล์สำหรับ การอ้างอิงในอนาคต ถ้าโปรแกรมที่มีดิสก์จำกัดรันสองครั้งต่อเนื่องกันทันที โดยปกติแล้ว ครั้งที่สองจะรันได้เร็วกว่าครั้งแรก อาจพบพฤติกรรมที่คล้ายกันนี้ กับโปรแกรมที่ใช้ NFS นอกจากนี้ ยังอาจเกิดขึ้นกับโปรแกรมขนาด ใหญ่ เช่น คอมไพเลอร์ด้วย ขั้นตอนวิธีของโปรแกรมอาจไม่ถูกจำกัดด้วยดิสก์ แต่เวลาที่ใช้ในการโหลดโปรแกรมที่ดำเนินการได้ขนาดใหญ่ อาจทำให้ การดำเนินการครั้งแรกของโปรแกรมใช้เวลานานกว่าครั้งต่อมามาก
- ถ้าโปรแกรมรันช้าเสมอ หรือช้าลงโดยไม่มีการเปลี่ยนแปลงที่ชัดเจนใดๆ ใน สภาพแวดล้อมของโปรแกรม ให้ดูการพึ่งพารีซอร์สของโปรแกรมนั้น

  การระบุรีซอร์สที่จำกัดประสิทธิภาพ อธิบาย เทคนิคการค้นหาปัญหาคอขวด

ทุกสิ่งรันช้าลง ณ บางเวลาของวัน

มีหลายเหตุผลที่ทำให้ระบบอาจรันช้าลง ณ บางเวลาของวัน

ผู้คนส่วนใหญ่เคยประสบการทำงานช้าลงในช่วงชั่วโมงเร่งด่วน เนื่องจาก ผู้คนจำนวนมากในองค์กรมีนิสัยใช้ระบบที่เวลาเฉพาะหนึ่งเวลาขึ้นไป ในแต่ละวัน ปรากฏการณ์นี้ไม่ใช่เรื่องเล็กเสมอไป เนื่องจากมีโหลดหนาแน่นมาก บางครั้งการช้าลงบ่งชี้ถึงความไม่สมดุล ที่เป็นปัญหาเฉพาะเมื่อโหลดสูงเท่านั้น สาเหตุอื่นของการเกิดสถานการณ์ดังกล่าวซ้ำแล้วซ้ำเล่า ในระบบควรต้องมีการพิจารณา

- ถ้าคุณรันคำสั่ง **iostat** และ **netstat** ในช่วงเวลา ที่ครอบคลุมเวลาของการช้าลง หรือถ้าคุณได้ตรวจจับข้อมูลจากกลไกการมอนิเตอร์ ของคุณก่อนหน้านี้ มีดิสก์ใดมีการใช้งานหนักมากกว่าดิสก์อื่นอย่างมาก หรือไม่? เปอร์เซ็นต์ CPU idle ใกล้ศูนย์อย่างสม่ำเสมอหรือไม่? จำนวนแพ็กเก็ต ที่ส่งหรือที่รับสูงผิดปกติหรือไม่?
  - ถ้าดิสก์ไม่สมดุล ให้ดู “ผลการทำงานของโลจิคัลวอลุ่มและดิสก์ I/O” ในหน้า 195
  - ถ้า CPU เต็ม ให้ใช้คำสั่ง **ps** หรือ **topas** เพื่อระบุโปรแกรมที่กำลังรันในระหว่างรอบเวลานี้ สคริปต์ตัวอย่างที่กำหนดใน “การมอนิเตอร์ผลการทำงานของระบบอย่างต่อเนื่องด้วยคำสั่ง” ในหน้า 16 ช่วยให้การค้นหาผู้ใช้ CPU หนักที่สุดง่ายขึ้น
  - ถ้าการช้าลงมีผลกระทบมากจนสัมผัสได้ เช่น ภาวะชะงักงันในระหว่างเวลาพัก กลางวัน ให้ค้นหาโปรแกรมต้นเหตุ เช่น โปรแกรมภาพ **xlock** หรือ โปรแกรมเกม เป็นที่ทราบกันดีว่า โปรแกรม **xlock** บางเวอร์ชันใช้เวลา CPU จำนวนมากเพื่อแสดงรูปแบบภาพบนจอแสดงผล idle และยังอาจเป็นไปได้ว่าบางคนกำลังรันโปรแกรมที่ทราบกันว่าเผาผลาญ CPU และกำลังพยายามรันในเวลาที่น้อยที่สุด
- ยกเว้นว่าไฟล์ `/var/adm/cron/cron.allow` ของคุณเป็น null คุณอาจต้องการตรวจสอบเนื้อหาของไดเร็กทอรี `/var/adm/cron/crontab` เพื่อหาการดำเนินงานที่แพง

ถ้าคุณพบว่าสาเหตุของปัญหามาจากความขัดแย้งระหว่างกิจกรรมพื้นหน้าและ โปรแกรมเน้น CPU ที่ใช้เวลารันนานซึ่งควรจะรันในพื้นหลัง ให้พิจารณาเปลี่ยนวิธีการคำนวณระดับความสำคัญโดยใช้คำสั่ง **schedo** เพื่อให้กิจกรรมพื้นหน้ามีระดับความสำคัญสูงกว่า โปรดดู “การคำนวณค่าระดับความสำคัญของเธรด” ในหน้า 135

ทุกสิ่งรันช้าลงในเวลาที่คาดการณ์ไม่ได้

เครื่องมือที่ดีที่สุดสำหรับสถานการณ์นี้คือตัวตรวจสอบโอเวอร์โหลด เช่น **filtd** daemon ซึ่งเป็นคอมโพเนนต์หนึ่งของ PTX

filtd daemon สามารถมีการตั้งค่าให้ดำเนินการ shell scripts หรือรวบรวมข้อมูลเฉพาะ เมื่อตรวจพบเงื่อนไขเฉพาะ คุณสามารถสร้าง กลไกคล้ายกันแต่มีลักษณะเฉพาะด้านมากขึ้นได้โดยใช้ shell scripts ที่มีคำสั่ง **vmstat, iostat, netstat, sar**, และ **ps**

ถ้าปัญหาเกิดขึ้นเฉพาะที่บนระบบหนึ่งในสภาพแวดล้อมที่แจกจ่าย อาจเป็นไปได้ว่าโปรแกรมต้นเหตุหนึ่งทำงาน หรืออาจเป็นสองโปรแกรมที่เป็นต้น เหตุร่วมกัน

ทุกสิ่งที่ผู้ใช้แต่ละรายรันอยู่จะช้า

ในบางครั้งระบบจะดูเหมือนว่ามีผลกระทบ

- โซลูชันในกรณีนี้คือ การหาจำนวนของปัญหา ให้ถามผู้ใช้ว่า คำสั่งใดที่ผู้ใช้ใช้บ่อย และรันคำสั่งเหล่านั้นด้วยคำสั่ง **time** ตามตัวอย่างดังต่อไปนี้:

  ```
  # time cp .profile testjunk
  real    0m0.08s
  user    0m0.00s
  sys     0m0.01s
  ```

  จากนั้น รันคำสั่งเดียวกันนี้ภายใต้ ID ผู้ใช้ที่ไม่มีประสบการณ์เกี่ยวกับปัญหาผลการทำงาน มีความแตกต่างกันในเวลา จริง ที่รายงานหรือไม่?
- โปรแกรมไม่ควรแสดงเวลา CPU (`user+sys`) ที่แตกต่างจากการรันมากเกินไป เพื่อรัน แต่อาจแสดงความแตกต่างของเวลาจริง เนื่องจากมี I/O เพิ่มเติมหรือช้าลง ไฟล์ของผู้ใช้ทั้งหมดอยู่บนไดเร็กทอรีที่ประกอบเข้ากับ NFS หรือไม่? หรือบนดิสก์ที่มีกิจกรรมสูง สำหรับเหตุผลอื่นๆ?
- ตรวจสอบไฟล์ `.profile` ของผู้ใช้สำหรับข้อกำหนดคุณสมบัติ *$PATH* ที่ผิดปกติ ตัวอย่างเช่น ถ้าคุณค้นหาไดเร็กทอรีที่ประกอบกับ NFS ก่อนที่จะค้นหาทุกสิ่งใน `/usr/bin` จะใช้เวลานานกว่าเดิม

จำนวนของระบบที่เชื่อมต่อกับ LAN จะช้าลงอย่างพร้อมเพียงกัน

มีปัญหาบางอย่างที่เกิดขึ้นในการส่งผ่านระบบอิสระ ไปยังระบบแบบกระจาย

ปัญหาจะมีผลมาจากความต้องการในการขอรับคอนฟิกูเรชันที่รันใหม่ในทันที หรือจากการขาดการรับรู้ในเรื่องของต้นทุนของฟังก์ชัน นอกจากการปรับคอนฟิกูเรชัน LAN ในส่วนของ maximum transmission units (MTU) และ mbufs แล้ว ให้มองหาสถานการณ์ที่ระบุเฉพาะ LAN หรือที่ไม่ได้ออปติไมซ์ซึ่งอาจพัฒนาผ่านลำดับของการตัดสินใจอย่างเป็นเหตุเป็นผล

- ใช้ข้อมูลสถิติด้านเน็ตเวิร์กเพื่อมั่นใจว่าไม่มีปัญหาในเรื่องของฟิสิคัลเน็ตเวิร์ก โปรดมั่นใจว่า คำสั่ง เช่น **netstat –v, entstat, tokstat, atmstat** หรือ **fddistat** ไม่ได้แสดงข้อผิดพลาดที่เกินจริงหรือความขัดแย้ง บนอะแด็ปเตอร์
- ชนิดของบักของซอฟต์แวร์หรือเฟิร์มแวร์บางชนิดสามารถมีอยู่บน LAN ด้วยการกระจายสัญญาณหรือแพ็กเก็ตอื่นๆ เป็นระยะๆ

  เมื่อการกระจายสัญญาณเกิดขึ้น ระบบจะทำให้เน็ตเวิร์กช้าลงด้วยการอินเตอร์รัปต์อย่างต่อเนื่อง และด้วยการใช้รีซอร์ส CPU ในการรับและประมวลผล แพ็กเก็ต ปัญหาเหล่านี้จะถูกตรวจพบได้ดีกว่า และ localize ด้วยอุปกรณ์วิเคราะห์ LAN พร้อมกับบทูลด้านประสิทธิภาพปกติ

- คุณมีการเชื่อมต่อ LAN สองวงผ่านระบบหรือไม่?
  การใช้ระบบที่เป็นเราเตอร์ จะใช้จำนวนเวลา CPU ขนาดใหญ่เพื่อประมวลผลหรือทำสำเนาแพ็กเก็ต และยังเกี่ยวข้องกับการรบกวนจากการทำงานอื่นๆ ที่ถูกประมวลผลโดยระบบ ฮาร์ดแวร์เราเตอร์เฉพาะงานและบริดจ์จะเป็นโซลูชันที่สมเหตุสมผล และเสถียร
- มีวัตถุประสงค์ในการล้างข้อมูลสำหรับ NFS ที่ประกอบเข้าแต่ละตัวหรือไม่?
  ในขั้นตอนนี้ การพัฒนาคอนฟิกูเรชันที่กระจาย การประกอบ NFS จะทำให้ผู้ใช้อยู่บนระบบใหม่ ที่เข้าถึงไดเร็กทอรีโฮมบนระบบเก่า สถานการณ์นี้จะทำให้ง่ายขึ้นด้วยการส่งผ่าน แต่กำหนดต้นทุนของ การสื่อสารข้อมูล ซึ่งจะไม่ทราบว่า มีผู้ใช้อยู่บนระบบ A ซึ่งโต้ตอบกับข้อมูลที่อยู่บนระบบ B เป็นต้น
  การเข้าถึงไฟล์ผ่าน NFS จะกำหนดต้นทุนที่สามารถนำมาพิจารณาได้ในทราฟิก LAN เวลา CPU ของเซิร์ฟเวอร์และไคลเอ็นต์ และเวลาตอบสนองของผู้ใช้ขั้นปลาย แนวทางโดยทั่วไปคือ ผู้ใช้และข้อมูลควรอยู่บนระบบเดียวกัน ข้อยกเว้นคือสถานการณ์เหล่านั้นจะถูกแทนที่ ซึ่งจะจัดให้เหมาะสมกับค่าใช้จ่ายและเวลาของข้อมูลรีโมต ตัวอย่างบางส่วนต้องการข้อมูลที่อยู่ในส่วนกลางสำหรับการสำรองและการควบคุมที่เชื่อถือได้ หรือต้องให้มั่นใจว่า ผู้ใช้ทั้งหมดกำลังทำงานด้วยเวอร์ชันของโปรแกรมปัจจุบัน มากที่สุด
  ถ้าความต้องการเหล่านี้และอื่นๆ ควบคุมระดับของการแลกเปลี่ยน NFS แบบไคลเอ็นต์–เซิร์ฟเวอร์ จึงเป็นการดีที่จะใช้ระบบเฉพาะงาน กับบทบาทของเซิร์ฟเวอร์ที่มีจำนวนของระบบซึ่งเป็นส่วนของเซิร์ฟเวอร์ ส่วนของไคลเอ็นต์
- โปรแกรมได้ถูกพอร์ตอย่างถูกต้องและใช้การเรียกโพรซีเดอร์แบบรีโมต (RPCs) หรือไม่?
  วิธีที่ง่ายที่สุดของการพอร์ตโปรแกรมลงในสภาวะแวดล้อมแบบกระจายคือ การแทนที่การเรียกโปรแกรมด้วย RPC ในอัตราส่วน 1:1 น่าเสียดาย ความแตกต่างกันในเรื่องของผลการทำงานระหว่างการเรียกโปรแกรมโลคัลและ RPC มีความแตกต่างที่มากกว่าความแตกต่างระหว่างโลคัลดิสก์k I/O และ NFS I/O RPC จะเป็นสิ่งที่จำเป็น และควรถูกแบ็ตช์ไว้เมื่อเป็นไปได้

ทุกสิ่งในการบริการหรืออุปกรณ์เฉพาะช้าลงตลอดเวลา

มีเหตุผลหลายอย่างที่ทำให้ทุกสิ่งในการบริการหรืออุปกรณ์เฉพาะ ช้าลงตลอดเวลา

ถ้าทุกสิ่งที่ใช้อุปกรณ์หรือการบริการเฉพาะช้าลงตลอดเวลา ให้อ้างอิงหัวข้อที่ครอบคลุมอุปกรณ์หรือการบริการเฉพาะดังนี้:

- “ผลการทำงานของไมโครโพรเซสเซอร์” ในหน้า 112
- “ผลการทำงานของหน่วยความจำ” ในหน้า 139
- “ผลการทำงานของโลจิคัลวอลุ่มและดิสก์ I/O” ในหน้า 195
- “ผลการทำงานของระบบไฟล์” ในหน้า 257
- “การวิเคราะห์ประสิทธิภาพเครือข่าย” ในหน้า 325
- “การมอนิเตอร์และการปรับประสิทธิภาพ NFS” ในหน้า 368

ทุกสิ่งรันช้าลงเมื่อเชื่อมต่อแบบรีโมต

การพิสูจน์ตัวตนแบบโลคัลและแบบรีโมตของระบบอาจมีพฤติกรรมแตกต่างกันมาก โดยค่าดีฟอลต์ จะมีการใช้ไฟล์การพิสูจน์ตัวตนแบบโลคัลก่อน เมื่อผู้ใช้ล็อกอิน ด้วย id ผู้ใช้ของตน ซึ่งมีเวลาการตอบกลับที่เร็วกว่ากลไกการพิสูจน์ตัวตน บนเครือข่าย

ถ้าผู้ใช้ล็อกอินและพิสูจน์ตัวตนด้วยกลไกการพิสูจน์ตัวตนบนเครือข่ายบาง ประเภท ระบบจะค้นหาในกลไกนั้นก่อนเมื่อต้องการค้นหา ids ผู้ใช้ ลักษณะเช่นนี้จะส่งผลกระทบต่อคำสั่งใดๆ ที่ทำการค้นหาชื่อล็อกอินของผู้ใช้ และยังกระทบต่อคำสั่งดังต่อไปนี้:

- **ps –ef**
- **ls –l**
- **ipcs –a**

โปรแกรมการพิสูจน์ตัวตนเฉพาะมีการกำหนดไว้ในไฟล์ /usr/lib/security/methods.cfg ค่าดีฟอลต์คือ compat ซึ่งเป็นวิธีการพิสูจน์ตัวตนแบบ โลคัล เมื่อต้องการดูค่าติดตั้งการพิสูจน์ตัวตนปัจจุบันของคุณสำหรับ id ผู้ใช้เฉพาะ ให้ล็อกอินด้วย id ผู้ใช้และที่บรรทัดคำสั่ง ให้พิมพ์:

```
# echo $AUTHSTATE
```

ถ้าคุณต้องการมั่นใจว่าคุณกำลังใช้กลไกการพิสูจน์ตัวตนแบบโลคัลก่อน กลไกการพิสูจน์ตัวตนบนเครือข่าย ตัวอย่างเช่น DCE ให้พิมพ์ข้อมูลดังต่อไปนี้ที่บรรทัดคำสั่ง:

```
# export AUTHSTATE="compat,DCE"
```

การระบุรีซอร์สที่จำกัดประสิทธิภาพ

เครื่องมือที่ดีที่สุดสำหรับการดูการใช้ประโยชน์รีซอร์สโดยรวมในขณะที่รัน เวิร์กโหลดหลายผู้ใช้คือคำสั่ง **vmstat**

คำสั่ง **vmstat** รายงาน CPU และกิจกรรมดิสก์ I/O ตลอดจนข้อมูลการใช้ประโยชน์หน่วยความจำ ตัวอย่างต่อไปนี้ของคำสั่ง **vmstat** จัดทำรายงานสรุปบนบรรทัดเดียวของกิจกรรมระบบทุก 5 วินาที:

```
# vmstat 5
```

ในตัวอย่างข้างบน เนื่องจากไม่มีการระบุจำนวนของช่วงเวลา ดังนั้นการรายงานจะดำเนินต่อไปจนกว่าคุณยกเลิกคำสั่ง

รายงาน **vmstat** ต่อไปนี้ถูกสร้างขึ้นบนระบบที่กำลังรัน AIXwindows และแอ็พพลิเคชันที่สร้างขึ้นหลายรายการ (บางช่วงเวลาที่มี กิจกรรมต่ำถูกลบออกเพื่อแสดงเป็นตัวอย่าง):

```
kthr     memory             page              faults        cpu
----- ----------- ------------------------ ------------ -----------
 r  b   avm   fre  re  pi  po  fr   sr  cy  in   sy  cs us sy id wa
 0  0  8793    81   0   0   0   1    7   0 125   42  30  1  2 95  2
 0  0  8793    80   0   0   0   0    0   0 155  113  79 14  8 78  0
 0  0  8793    57   0   3   0   0    0   0 178   28  69  1 12 81  6
 0  0  9192    66   0   0  16  81  167   0 151   32  34  1  6 77 16
 0  0  9193    65   0   0   0   0    0   0 117   29  26  1  3 96  0
 0  0  9193    65   0   0   0   0    0   0 120   30  31  1  3 95  0
 0  0  9693    69   0   0  53 100  216   0 168   27  57  1  4 63 33
 0  0  9693    69   0   0   0   0    0   0 134   96  60 12  4 84  0
 0  0 10193    57   0   0   0   0    0   0 124   29  32  1  3 94  2
 0  0 11194    64   0   0  38 201 1080   0 168   29  57  2  8 62 29
 0  0 11194    63   0   0   0   0    0   0 141  111  65 12  7 81  0
 0  0  5480   755   3   1   0   0    0   0 154  107  71 13  8 78  2
 0  0  5467  5747   0   3   0   0    0   0 167   39  68  1 16 79  5
 0  1  4797  5821   0  21   0   0    0   0 191  192 125 20  5 42 33
 0  1  3778  6119   0  24   0   0    0   0 188  170  98  5  8 41 46
 0  0  3751  6139   0   0   0   0    0   0 145   24  54  1 10 89  0
```

ในการประเมินเบื้องต้นนี้ ควรให้ความสนใจเป็นพิเศษกับคอลัมน์ *pi* และ *po* ของหมวดหมู่ page และสี่คอลัมน์ในหมวดหมู่ cpu

รายการ *pi* และ *po* แสดงถึง page-ins และ page-outs ของพื้นที่ว่างการเพจ ตามลำดับ ถ้าคุณสังเกตอินสแตนซ์ใดๆ ของ I/O พื้นที่ว่างการเพจ เวิร์กโหลดอาจเข้าใกล้หรือเกินกว่าขีดจำกัดหน่วยความจำของระบบ

ถ้าผลรวมของเปอร์เซ็นต์การใช้ประโยชน์ CPU ของผู้ใช้และระบบ, *us* และ *sy*, มากกว่า 90 เปอร์เซ็นต์ในช่วงเวลา 5 วินาทีที่กำหนด แสดงว่าเวิร์กโหลดกำลัง เข้าใกล้ขีดจำกัด CPU ของระบบในระหว่างช่วงเวลานั้น

ถ้าเปอร์เซ็นต์การรอ I/O, *wa*, ใกล้กับศูนย์ และค่า *pi* และ *po* เป็นศูนย์ แสดงว่าระบบกำลังใช้เวลารอไฟล์ I/O ที่ไม่ซ้อนเหลื่อมกัน และบางส่วนของเวิร์กโหลดถูกจำกัด I/O

ถ้าคำสั่ง **vmstat** บ่งชี้เวลารอ I/O นานมาก ให้ใช้คำสั่ง **iostat** เพื่อรวบรวมข้อมูลรายละเอียด เพิ่มเติม

ตัวอย่างต่อไปนี้ของคำสั่ง **iostat** จัดทำรายงานสรุปของ กิจกรรม I/O และการใช้ประโยชน์ CPU ทุก 5 วินาที และเนื่องจาก เรา ระบุจำนวนช่วงเวลาเป็น 3 ดังนั้นการรายงานจะหยุดหลังจาก รายงานฉบับที่สาม:

```
# iostat 5 3
```

รายงาน **iostat** ต่อไปนี้ถูกสร้างขึ้นบนระบบที่กำลังรัน เวิร์กโหลดเดียวกันกับในตัวอย่าง **vmstat** ข้างบน แต่ในเวลาที่แตกต่างกัน รายงานแรกแสดงกิจกรรมสะสมตั้งแต่ บูตครั้งก่อนหน้านี้ ในขณะที่รายงานต่อมาแสดงกิจกรรมในระหว่าง ช่วงเวลา 5 วินาทีก่อนหน้านี้:

```
tty:      tin         tout   avg-cpu:  % user    % sys     % idle    %iowait
          0.0          4.3              0.2       0.6       98.8       0.4

Disks:        % tm_act     Kbps      tps    Kb_read   Kb_wrtn
hdisk0           0.0       0.2       0.0       7993      4408
hdisk1           0.0       0.0       0.0       2179      1692
hdisk2           0.4       1.5       0.3      67548     59151
cd0              0.0       0.0       0.0          0         0

tty:      tin         tout   avg-cpu:  % user    % sys     % idle    %iowait
          0.0         30.3              8.8       7.2       83.9       0.2

Disks:        % tm_act     Kbps      tps    Kb_read   Kb_wrtn
hdisk0           0.2       0.8       0.2          4         0
hdisk1           0.0       0.0       0.0          0         0
hdisk2           0.0       0.0       0.0          0         0
cd0              0.0       0.0       0.0          0         0

tty:      tin         tout   avg-cpu:  % user    % sys     % idle    %iowait
          0.0          8.4              0.2       5.8        0.0       93.8

Disks:        % tm_act     Kbps      tps    Kb_read   Kb_wrtn
hdisk0           0.0       0.0       0.0          0         0
hdisk1           0.0       0.0       0.0          0         0
hdisk2          98.4     575.6      61.9        396      2488
cd0              0.0       0.0       0.0          0         0
```

รายงานแรกแสดงว่า I/O บนระบบนี้ไม่สมดุล I/O ส่วนใหญ่ (86.9 เปอร์เซ็นต์ของกิโลไบต์ที่อ่านและ 90.7 เปอร์เซ็นต์ของกิโลไบต์ที่บันทึก) ไปยัง hdisk2 ซึ่งมีทั้งระบบปฏิบัติการและพื้นที่ว่าง การเพจ สถิติ *CPU utilization since boot* สะสมโดยปกติแล้ว ไม่ค่อยมีความหมาย ยกเว้นว่าคุณใช้ระบบอย่างสม่ำเสมอตลอด 24 ชั่วโมงต่อวัน

รายงานที่สองแสดงจำนวนเล็กน้อยของการอ่านกิจกรรมดิสก์จาก hdisk0 ซึ่งมีระบบไฟล์ที่แยกต่างหากสำหรับผู้ใช้หลักของระบบ กิจกรรม CPU เกิดขึ้นจากสองแอ็พพลิเคชันโปรแกรมและตัวคำสั่ง **iostat** เอง

ในรายงานฉบับที่สาม คุณสามารถเห็นว่าเราได้สร้างเงื่อนไข near-thrashing ขึ้นโดยการรันโปรแกรมที่จัดสรรและจัดเก็บหน่วยความจำจำนวนมาก ประมาณ 26 MB ในตัวอย่างข้างบน นอกจากนี้ ในตัวอย่างข้างบน hdisk2 มีการใช้งาน 98.4 เปอร์เซ็นต์ของเวลา ซึ่งส่งผลให้การรอ I/O เป็น 93.8 เปอร์เซ็นต์

ปัจจัยที่จำกัดสำหรับโปรแกรมเดี่ยว

ถ้าคุณคือผู้ใช้คนเดียวของระบบ คุณสามารถขอรับแนวคิดทั่วไปที่ว่า โปรแกรมคือ I/O หรือ CPU ที่พึ่งพาการใช้คำสั่ง **time** ดังนี้:

```
# time cp foo.in foo.out

real    0m0.13s
user    0m0.01s
sys     0m0.02s
```

หมายเหตุ: ตัวอย่างของคำสั่ง **time** จะใช้เวอร์ชันที่สร้างภายใน Korn shell **ksh** สำหรับคำสั่ง **time** อย่างเป็นทางการ /usr/bin/time จะรายงานด้วยความแม่นยำต่ำกว่า

ในตัวอย่างข้างต้น ความจริงที่ว่า เวลาที่ใช้ไป จริง สำหรับการประมวลผลของโปรแกรม **cp** (0.13 วินาที) จะมีค่ามากกว่าผลรวม (.03 วินาที) ของผู้ใช้และเวลาของ CPU ระบบจะบ่งชี้ว่าโปรแกรมถูกโยง I/O ไว้ เหตุการณ์นี้เกิดขึ้นในครั้งแรก เนื่องจากไฟล์ foo.in ถูกอ่านอย่างไม่ถูกต้อง

สำหรับ SMP เอาต์พุตจะใช้ในความหมายใหม่ โปรดดู “ข้อควรพิจารณาเกี่ยวกับคำสั่ง time และ timex” ในหน้า 122 สำหรับข้อมูลเพิ่มเติม

การรันคำสั่งเดียวกันในเวลาสองวินาทีหลังจากที่ไฟลเดียวกันนี้ แสดงเอาต์พุตต่อไปนี้:

```
real    0m0.06s
user    0m0.01s
sys     0m0.03s
```

เพจทั้งหมดของไฟล์ foo.in ยังคงอยู่ในหน่วยความจำ เนื่องจากไม่มีการแทรกแซงการประมวลผล เพื่อทำให้เรียกคืน และเนื่องจากไฟล์มีขนาดเล็กเกินไปเปรียบเทียบกับจำนวนของf RAM บนระบบ ไฟล์ foo.out ขนาดเล็กยังถูกบัฟเฟอร์ในหน่วยความจำ และโปรแกรมใช้อินพุตที่แสดงการพึ่งพาดิสก์เล็กน้อย

ถ้าคุณกำลังพยายามกำหนดการพึ่งพาดิสก์ของโปรแกรม คุณต้องมั่นใจว่า อินพุตอยู่ในสถานะเฝ้าดู นั่นคือ ถ้าโปรแกรมจะรันพร้อมกับไฟล์ซึ่งไม่ได้เข้าถึงเมื่อเร็วๆ นี้ คุณต้องแน่ใจว่า ไฟล์นั้นถูกใช้ในการวัดโปรแกรมที่ไม่ได้อยู่ในหน่วยความจำ หรืออีกนัยหนึ่ง ถ้าโปรแกรมรันส่วนของลำดับมาตรฐาน ซึ่งได้รับอินพุตจากเอาต์พุตของโปรแกรมที่อยู่หน้า คุณควรจัดลำดับหน่วยความจำเพื่อมั่นใจว่าการวัดนั้นถูกเฝ้าดูอยู่ ตัวอย่างเช่น คำสั่งต่อไปนี้ควรได้รับผลกระทบของหน่วยความจำที่จัดลำดับด้วยเพจของไฟล์ foo.in:

```
# cp foo.in /dev/null
```

สถานการณ์จะมีความซับซ้อนมากขึ้นหากไฟล์มีขนาดใหญ่เมื่อเปรียบเทียบกับ RAM ถ้า เอาต์พุตของหนึ่งโปรแกรมคืออินพุตของโปรแกรมถัดไป และไฟล์ทั้งหมด จะไม่ได้อยู่ใน RAM โปรแกรมที่สองจะอ่านเพจที่ส่วนหัวของไฟล์ ซึ่งบังคับเพจให้ออกในตอนจบ แม้ว่า สถานการณ์นี้เป็นสถานการณ์ที่ยากต่อการจำลอง แต่ก็เทียบเท่ากับสถานกาณณ์ที่ไม่มีการแคชดิสก์ ที่เข้าแทนที่

กรณีของไฟล์ที่มีขนาดใหญ่กว่า RAM จะเป็นกรณีพิเศษของ RAM เมื่อเปรียบเทียบกับการวิเคราะห์ดิสก์ในส่วนถัดไป

ปัญหาเกี่ยวกับดิสก์หรือหน่วยความจำ

เพียงแค่เศษส่วนขนาดใหญ่ของหน่วยความจำที่ใช้จริงจะพร้อมใช้งานสำหรับการบัฟเฟอร์ไฟล์ พื้นที่การเพจของระบบจะพร้อมใช้งานเป็นหน่วยเก็บชั่วคราวสำหรับโปรแกรมข้อมูลการทำงาน ที่ได้บังคับให้ออกจาก RAM

สมมุติว่าคุณมีโปรแกรมที่อ่านข้อมูลเพียงเล็กน้อย หรือไม่มีการอ่านข้อมูล และยังแสดงอาการของการพึ่งพา I/O สิ่งที่แย่ที่สุดคือ อัตราเรียลไทม์ของผู้ใช้ + เวลาของระบบไม่ได้รับผลกระทบด้วยการรันที่ต่อเนื่องกัน โปรแกรมจะจำกัดหน่วยความจำ และ I/O ของโปรแกรมจากหรือไปยังพื้นที่การเพจ วิธีการตรวจสอบความเป็นไปได้นี้ จะแสดงอยู่ในสคริปต์ shell **vmstatit** ต่อไปนี้:

```
vmstat -s >temp.file   # cumulative counts before the command
time $1                # command under test
vmstat -s >>temp.file  # cumulative counts after execution
grep "pagi.*ins" temp.file >>results   # extract only the data
grep "pagi.*outs" temp.file >>results  # of interest
```

สคริปต์ **vmstatit** จะสรุปรายงาน **vmstat –s** ซึ่งกำหนดให้สะสมการนับจำนวนของกิจกรรมของระบบ ตั้งแต่ที่สตาร์ทระบบ

ถ้าสคริปต์ shell ถูกรันดังต่อไปนี้:

```
# vmstatit "cp file1 file2"  2>results
```

ผลลัพธ์ที่ได้จะเป็นดังนี้:

```
real    0m0.03s
user    0m0.01s
sys     0m0.02s
     2323 paging space page ins
     2323 paging space page ins
     4850 paging space page outs
     4850 paging space page outs
```

ข้อมูลสถิติก่อนและหลังการเพจจะเป็นข้อมูลเฉพาะ ซึ่งจะยืนยันความเชื่อที่ว่า คำสั่ง **cp** ไม่ใช่การโยงการเพจ ความต่างที่ขยายเพิ่มของสคริปต์ shell **vmstatit** สามารถนำมาใช้ เพื่อแสดงสถานการณ์ที่เป็นจริง ดังต่อไปนี้:

```
vmstat -s >temp.file
time $1
vmstat -s >>temp.file
echo "Ordinary Input:"               >>results
grep "^[ 0-9]*page ins"    temp.file >>results
echo "Ordinary Output:"              >>results
grep "^[ 0-9]*page outs"   temp.file >>results
echo "True Paging Output:"           >>results
grep "pagi.*outs"          temp.file >>results
echo "True Paging Input:"            >>results
grep "pagi.*ins"           temp.file >>results
```

เนื่องจากไฟล์ I/O ในระบบปฏิบัติการถูกประมวลผลผ่าน VMM คำสั่ง **vmstat -s** จะรายงานโปรแกรม I/O เริ่มแรก เป็นเพจขาเข้าและเพจขาออก เมื่อเวอร์ชันก่อนหน้าของสคริปต์ shell **vmstatit** ถูกรันพร้อมกับคำสั่ง **cp** ของไฟล์ขนาดใหญ่ ซึ่งไม่ได้ถูกอ่านไว้ ผลลัพธ์ที่ได้จะเป็นดังนี้:

```
real    0m2.09s
user    0m0.03s
sys     0m0.74s
Ordinary Input:
    46416 page ins
    47132 page ins
Ordinary Output:
   146483 page outs
   147012 page outs
True Paging Output:
     4854 paging space page outs
     4854 paging space page outs
True Paging Input:
     2527 paging space page ins
     2527 paging space page ins
```

เอาต์พุตคำสั่ง **time** จะยืนยันการมีอยู่ของการพึ่งพา I/O การเพิ่มขึ้นในเพจขาเข้าจะแสดง I/O ที่จำเป็น เพื่อตอบสนองคำสั่ง **cp** การเพิ่มขึ้นในเพจขาออกจะบ่งชี้ว่า ไฟล์มีขนาดใหญ่พอที่จะบังคับให้เขียน dirty page (ไม่จำเป็นต้องเป็นของตนเอง) จากหน่วยความจำ ความเป็นจริงที่ว่า ไม่มีการเปลี่ยนแปลงในจำนวน paging-space-I/O ที่สะสมไว้ซึ่งยืนยันว่า คำสั่ง **cp** ไม่ได้สร้างโครงสร้างข้อมูลที่มีขนาดใหญ่พอ เพื่อโหลดหน่วยความจำของเครื่องทดสอบเพิ่มมากขึ้น

รายงานลำดับในเวอร์ชันของสคริปต์การรายงาน I/O **vmstatit** นี้ I/O เป็นไปตามเจตนา โปรแกรมตัวอย่างจะอ่านไฟล์อินพุต และจากนั้น เขียนไฟล์เอาต์พุต หรืออีกนัยหนึ่ง กิจกรรมการเพจจะเริ่มต้นด้วยการเขียนออกนอกเพจเซ็กเมนต์การทำงานที่ไม่เหมาะสม เพจจะถูกอ่านกลับ หากโปรแกรมพยายามเข้าถึงเพจนั้นเท่านั้น ความจริงที่ว่า ระบบทดสอบได้พบกับ *พื้นที่การเพจที่เพจออก* เป็นสองเท่าของ *พื้นที่การเพจที่เพจเข้า* เนื่องจากระบบที่บูตจะบ่งชี้ว่า อย่างน้อยที่สุด โปรแกรมบางโปรแกรมต้องรันอยู่บนระบบนี้ ซึ่งเก็บข้อมูลในหน่วยความจำ ที่ไม่ได้เข้าถึงอีกครั้ง ก่อนที่จะจบโปรแกรม “โปรแกรมที่จำกัดหน่วยความจำ” ในหน้า 103 จะแสดงข้อมูลเพิ่มเติม โปรดดู “ผลการทำงานของหน่วยความจำ” ในหน้า 139

หากต้องการแสดงผลกระทบของการจำกัดหน่วยความจเกี่ยวกับข้อมูลสถิติเหล่านี้ ตัวอย่างต่อไปนี้จะเฝ้าดูคำสั่งที่กำหนดไว้ในสภาวะแวดล้อมของหน่วยความที่พอเพียง (32 MB) และการหดตัวของระบบโดยใช้คำสั่ง **rmss** (โปรดดู “การประเมินผลข้อกำหนดด้านหน่วยความจำด้วยคำสั่ง rmss” ในหน้า 155) คำสั่งต่อไปนี้จะมีลำดับดังนี้

```
# cc -c ed.c
# vmstatit "cc -c ed.c" 2>results
```

อันดับแรก หน่วยความจำที่มาเป็นลำดับแรกพร้อมกับซอร์สไฟล์ 7944 บรรทัด และไฟล์เรียกทำงานของคอมไพลเลอร์ C จากนั้น วัดค่ากิจกรรม I/O ของการประมวลผลอันดับที่สอง:

```
real    0m7.76s
user    0m7.44s
sys     0m0.15s
Ordinary Input:
    57192 page ins
    57192 page ins
Ordinary Output:
   165516 page outs
   165553 page outs
```

```
True Paging Output:
    10846 paging space page outs
    10846 paging space page outs
True Paging Input:
     6409 paging space page ins
     6409 paging space page ins
```

นี่ไม่ใช่ I/O ที่ถูกจำกัดไว้ ไม่มี I/O ใดๆ ที่จำเป็นต้องอ่านซอร์สโค้ด ถ้าเราออกคำสั่งต่อไปนี้:

```
# rmss -c 8
```

เพื่อเปลี่ยนขนาดที่ผลกระทบของเครื่องไปเป็น 8 MB และดำเนินการตามลำดับของคำสั่งเดียวกัน เราจะได้รับเอาต์พุตดังต่อไปนี้:

```
real    0m9.87s
user    0m7.70s
sys     0m0.18s
Ordinary Input:
    57625 page ins
    57809 page ins
Ordinary Output:
   165811 page outs
   165882 page outs
True Paging Output:
    11010 paging space page outs
    11061 paging space page outs
True Paging Input:
     6623 paging space page ins
     6701 paging space page ins
```

อาการต่อไปนี้ของการพึ่งพา I/O จะมีอยู่:

- เวลาที่ใช้ไปจะยาวนานกว่าเวลา CPU ทั้งหมด
- จำนวนของ I/O ที่มีความสำคัญในระดับปกติบนการประมวลผล *nth* ของคำสั่ง

ความเป็นจริงที่ว่า เวลาที่ใช้ไปจะยาวนานกว่าในสถานการณ์ที่ไม่มีข้อจำกัดเกี่ยวกับหน่วยความจำ และการมีอยู่ของจำนวน I/O ของพื้นที่การเพจที่สำคัญ จะทำให้มีความชัดเจนที่ว่า คอมไพลเลอร์จะถูกทำให้หยุดชะงักโดยหน่วยความจำที่ไม่เพียงพอ

หมายเหตุ: ตัวอย่างนี้ จะแสดงให้เห็นถึงผลกระทบของข้อจำกัดเกี่ยวกับหน่วยความจำ ไม่มีความพยายามที่ถูกทำขึ้น เพื่อลดการใช้หน่วยความจำโดยการประมวลผลอื่นๆ ดังนั้น ขนาดที่แน่นอนที่คอมไพลเลอร์ควรบังคับให้เพจในสภาวะแวดล้อมนี้ไม่ควรตั้งการวัด ที่มีความหมาย

หากต้องการหลีกเลี่ยงการทำงานกับเครื่องที่มีผลการทำงานที่ลดลงจนกว่าจะถึงการรีสตาร์ทในครั้งถัดไป ให้รัน

```
# rmss -r
```

หากต้องการกลับสู่ระบบปฏิบัติการที่มีหน่วยความจำที่คำสั่ง **rmss** ถูกแยกออก แล้วจึงเรียกคืนระบบกลับไปยังความสามารถเดิม

การวิเคราะห์การจัดการเวิร์กโหลด

การจัดการเวิร์กโหลดหมายถึงการเข้าถึงระดับความสำคัญของแต่ละ คอมโพเนนต์ของเวิร์กโหลด

เมื่อคุณใช้การปรับปรุงประสิทธิภาพของโปรแกรมและโอกาสการปรับระบบ และประสิทธิภาพยังคงไม่น่าพึงพอใจเสมอ คุณมีตัวเลือกสามรายการ ดังนี้:

- ปล่อยให้สถานการณ์ยังคงเดิม
- อัพเกรดรีซอร์สที่จำกัดประสิทธิภาพ
- ใช้เทคนิคการจัดการเวิร์กโหลด

ตัวเลือกแรกอาจสร้างความหงุดหงิดและผลผลิตที่ลดลงสำหรับผู้ใช้ บางรายของคุณ ถ้าคุณเลือกที่จะอัพเกรดรีซอร์ส คุณต้องสามารถควบคุม ค่าใช้จ่ายได้ ดังนั้นวิธีแก้ไขที่น่าสนใจที่สุดคือการสืบสวน โอกาสของการจัดการเวิร์กโหลด

โดยปกติ มีงานที่คุณสามารถเลื่อนออกไปได้ ตัวอย่างเช่น รายงานที่คุณต้องใช้ ในตอนเช้า การรันรายงานนั้นเมื่อเวลา 3 a.m. หรือ 4 p.m. ของวันก่อนหน้า มีประโยชน์เหมือนกัน สิ่งที่แตกต่างคือรอบ CPU ที่ใช้ และรีซอร์สอื่นซึ่งมีโอกาส idle มากที่สุด เมื่อเวลา 3 a.m. คุณสามารถใช้คำสั่ง **at** หรือ **crontab** เพื่อร้องขอโปรแกรมที่จะรันในเวลาเฉพาะหรือในช่วงเวลาที่สม่ำเสมอ

ในลักษณะคล้ายกัน บางโปรแกรมที่ต้องรันในระหว่างวันสามารถรันที่ ระดับความสำคัญลดลงได้ แม้ว่าต้องใช้เวลารันนานขึ้น แต่มีการแข่งขันน้อยลงกับกระบวนการ ซึ่งจำเป็นต้องรันในเวลาเฉพาะ

อีกเทคนิคหนึ่งคือการย้ายงานจากเครื่องหนึ่งไปยังอีกเครื่องหนึ่ง ตัวอย่างเช่น ถ้าคุณรันการคอมไพล์บนเครื่องที่มีซอร์สโค้ด การปรับสมดุล เวิร์กโหลดประเภทนี้จำเป็นต้องวางแผนและมอนิเตอร์มากขึ้นเนื่องจาก การลดโหลดบนเครือข่ายและการเพิ่ม CPU โหลดบนเซิร์ฟเวอร์อาจส่งผลให้ สูญเสียการเชื่อมต่อ

AIX Workload Manager (WLM) เป็นส่วนประกอบของเคอร์นัลระบบปฏิบัติการ WLM ออกแบบมาเพื่อให้ผู้ดูแลระบบสามารถควบคุม ได้มากขึ้นในวิธีการที่ตัวจัดตารางเวลาและผู้จัดการหน่วยความจำเสมือน (VMM) จัดสรรรีซอร์ส CPU และหน่วยความจำฟิสิคัลให้กับกระบวนการ WLM ยังสามารถ ควบคุมการใช้ดิสก์ได้ด้วย ความสามารถนี้ช่วยป้องกันไม่ให้งานประเภทต่างกัน รบกวนซึ่งกันและกัน และทำให้สามารถกำหนดรีซอร์สตามความต้องการของ กลุ่มผู้ใช้ที่แตกต่างกันได้ หากต้องการข้อมูลเพิ่มเติม ให้ดู *ข้อควรพิจารณา เกี่ยวกับเซิร์ฟเวอร์บน RS/6000®*

การจัดการรีซอร์ส

AIX มีคอมโพเนนต์ที่ปรับได้ สำหรับการจัดการรีซอร์สซึ่งมีผลกระทบต่อประสิทธิภาพระบบมากที่สุด

หากต้องการคำแนะนำเกี่ยวกับการปรับเฉพาะ ให้ดูข้อมูลต่อไปนี้:

- “ผลการทำงานของไมโครโพรเซสเซอร์” ในหน้า 112.
- “ผลการทำงานของหน่วยความจำ” ในหน้า 139.
- “ผลการทำงานของโลจิคัลวอลุ่มและดิสก์ I/O” ในหน้า 195.
- “ประสิทธิภาพเครือข่าย” ในหน้า 282.
- “ประสิทธิภาพการทำงาน NFS” ในหน้า 362.

ประสิทธิภาพตัวจัดตารางเวลาตัวประมวลผล

มีปัญหาเกี่ยวกับประสิทธิภาพหลายอย่างที่ต้องพิจารณาเกี่ยวกับ ตัวจัดตารางเวลาตัวประมวลผล

การสนับสนุนเธรด

เธรด เป็นกระบวนการที่มีโอเวอร์เฮดต่ำ เธรด คือเอนทิตีที่จัดส่งได้ซึ่งใช้ทรัพากรในการสร้างน้อยกว่ากระบวนการ เอนทิตีที่จัดส่งได้พื้นฐานของตัวจัดตารางเวลาของ AIX เวอร์ชัน 4 คือเธรด

กระบวนการประกอบขึ้นจากเธรดหนึ่งรายการขึ้นไป ในข้อเท็จจริง เวิร์กโหลดที่ย้ายโดยตรง มาจากรีลีสก่อนหน้านี้ของระบบปฏิบัติการยังคงสร้าง และจัดการกระบวนการต่อไป กระบวนการใหม่แต่ละรายการมีการสร้างขึ้น ด้วยเธรดเดียว ซึ่งมีระดับความสำคัญของกระบวนการพาเรนต์และช่วงชิงตัวประมวล ผลกับเธรดของกระบวนการอื่น กระบวนการเป็นเจ้าของทรัพากรที่ใช้ในการดำเนินการ ในขณะที่ เธรดเป็นเจ้าของเฉพาะในสภาวะปัจจุบันของเธรดเท่านั้น

เมื่อแอ็พพลิเคชันใหม่หรือที่แก้ไขใช้ประโยชน์จากการสนับสนุนเธรดของ ระบบปฏิบัติการเพื่อสร้างเธรดเพิ่มเติม เธรดเหล่านั้นจะถูกสร้างขึ้นภายใน บริบทของกระบวนการ เธรดดังกล่าวแบ่งใช้เซกเมนต์ส่วนตัวของกระบวนการและรีซอร์ส อื่นๆ

เธรดผู้ใช้ภายในกระบวนการมี *ขอบเขตการช่วงชิง* ที่ระบุ ถ้าขอบเขตการช่วงชิงเป็น *สากล* เธรดจะช่วงชิงเวลา ตัวประมวลผลกับเธรดอื่นทั้งหมดในระบบ เธรดที่สร้างขึ้นเมื่อกระบวนการ ถูกสร้างขึ้นมีขอบเขตการช่วงชิงเป็นแบบสากล ถ้าขอบเขตการช่วงชิงเป็น *โลคัล* เธรดจะช่วงชิงกับเธรดอื่นภายในกระบวนการ เพื่อให้ได้เป็นผู้รับเวลาตัวประมวลผลของการแบ่งใช้กระบวนการ

ขั้นตอนวิธีสำหรับการกำหนดว่าควรจะรันเธรดใดถัดไปเรียกกันว่า *นโยบายการจัดตารางเวลา*

กระบวนการและเธรด

กระบวนการคือกิจกรรมภายในระบบที่เริ่มต้นขึ้นโดย คำสั่ง โปรแกรม shell หรือกระบวนการอื่น

คุณสมบัติกระบวนการมีดังนี้:

- pid
- pgid
- uid
- gid
- สภาพแวดล้อม
- cwd
- คำอธิบายไฟล์
- การดำเนินการสัญญาณ
- สถิติกระบวนการ
- nice

คุณสมบัติเหล่านี้มีการกำหนดไว้ในไฟล์ `/usr/include/sys/proc.h`

คุณสมบัติเธรดมีดังนี้:

- สแต็ค
- นโยบายการจัดตารางเวลา
- ระดับความสำคัญการจัดตารางเวลา
- สัญญาณที่ค้างอยู่

- สัญญาณที่บล็อก
- ข้อมูลเฉพาะเธรด

คุณสมบัติเธรดเหล่านี้มีการกำหนดไว้ในไฟล์ /usr/include/sys/thread.h

แต่ละกระบวนการประกอบขึ้นจากเธรดหนึ่งรายการขึ้นไป เธรดคือลำดับของการควบคุมตามลำดับ สัญญาณ การควบคุมแบบหลายเธรดช่วยให้แอ็พพลิเคชันสามารถทำการดำเนินงาน ซ้อนเหลื่อมกันได้ เช่น การอ่านจากเทอร์มินัลและการบันทึกลงในไฟล์

การควบคุมแบบหลายเธรดยังทำให้แอ็พพลิเคชันสามารถให้บริการการร้องขอ จากผู้ใช้หลายรายในเวลาเดียวกันได้ เธรดนำเสนอความสามารถเหล่านี้โดยไม่เพิ่ม โอเวอร์เฮดของหลายกระบวนการ เช่น กระบวนการที่สร้างขึ้นโดยใช้การเรียกระบบ **fork( )**

มีการแนะนำรูทีน fork ที่รวดเร็วซึ่งเรียกกันว่า **f_fork( )** ใน AIX รูทีนนี้มีประโยชน์สำหรับแอ็พพลิเคชัน ที่มีหลายเธรดซึ่งจะเรียกรูทีนย่อย **exec( )** ทันทีหลังจากการเรียกรูทีนย่อย **fork( )** รูทีนย่อย **fork( )** ทำงานช้าลง เนื่องจากต้องเรียกตัวจัดการ fork เพื่อจัดหาไลบรารีล็อคทั้งหมด ก่อนจะ forking จริงและอนุญาตให้ชายน์รันตัวจัดการชายน์เพื่อเริ่มต้น ล็อค รูทีนย่อย **f_fork( )** ข้ามตัวจัดการเหล่านี้ และเรียกการเรียกระบบ **kfork( )** โดยตรง เว็บเซิร์ฟเวอร์เป็น ตัวอย่างที่ดีของแอ็พพลิเคชันที่สามารถใช้รูทีนย่อย **f_fork( )** ได้

ระดับความสำคัญของกระบวนการและเธรด

เครื่องมือจัดการระดับความสำคัญจัดการกับระดับความสำคัญของกระบวนการ

ใน AIX เวอร์ชัน 4 ระดับความสำคัญของกระบวนการเป็นข้อมูลตั้งต้นสำหรับระดับความสำคัญของเธรด เมื่อมีการเรียกรูทีนย่อย **fork( )** จะมีการสร้างกระบวนการและเธรดที่จะรันในรูทีนย่อยนั้นขึ้น เธรดมี ระดับความสำคัญซึ่งอาจถูกกำหนดเป็นแอ็ตทริบิวต์ของกระบวนการ

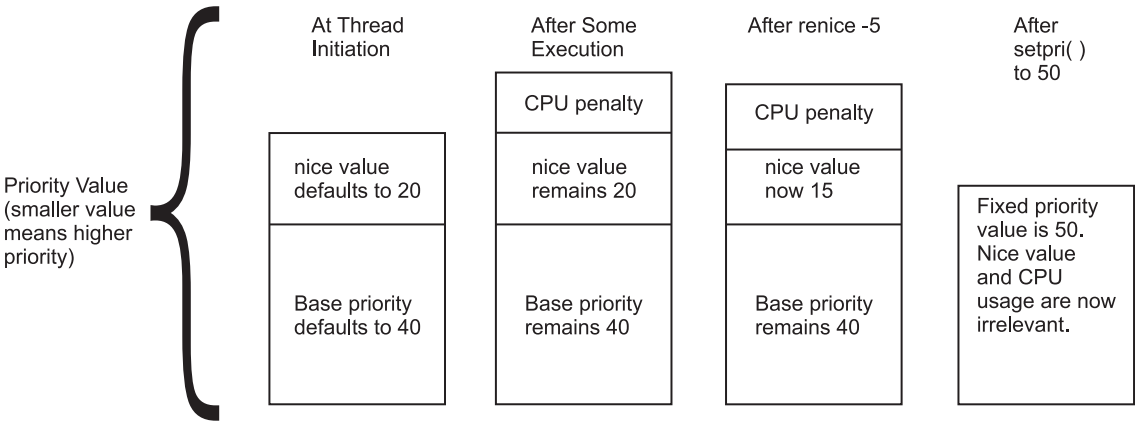
เคอร์นัลเก็บรักษา *ค่าระดับความสำคัญ* (บางครั้งเรียกว่า *ระดับความสำคัญการจัดตารางเวลา*) สำหรับแต่ละเธรด ค่าระดับความสำคัญ เป็นจำนวนเต็มบวก และแตกต่างกันไปอย่างมากตามความสำคัญของ เธรดที่เชื่อมโยง นั่นคือ ค่าระดับความสำคัญที่น้อยกว่า บ่งชี้ถึงเธรดที่มีความสำคัญมากกว่า เมื่อตัวจัดตารางเวลาค้นหาเธรดที่จะจัดส่ง ตัวจัดตารางเวลาจะเลือกเธรดที่จัดส่งได้ซึ่งมีค่าระดับความสำคัญ ต่ำที่สุด

เธรดสามารถมีระดับความสำคัญแบบคงที่หรือแบบไม่คงที่ ค่าระดับความสำคัญ ของเธรดที่มีระดับความสำคัญแบบคงที่เป็นค่าคงที่ ในขณะที่ค่าระดับความสำคัญ ของเธรดที่มีระดับความสำคัญแบบไม่คงที่ขึ้นอยู่กับระดับของระดับความสำคัญต่ำสุดสำหรับเธรดผู้ใช้ (ค่าคงที่ 40), ค่า nice ของเธรด (20 โดยค่าดีฟอลต์ ซึ่งสามารถเลือกที่จะตั้งค่าได้โดยใช้คำสั่ง **nice** หรือ **renice**) และ penalty การใช้ตัวประมวลผล

ระดับความสำคัญของเธรดสามารถระบุเป็นค่าแน่นอน ซึ่งอาจมีค่าระดับความ สำคัญน้อยกว่า 40 ได้ ถ้าระดับความสำคัญของเธรดนั้นมีการตั้งค่า (แบบคงที่) โดยใช้รูทีนย่อย **setpri( )** เธรดเหล่านี้ได้รับการป้องกันจาก ขั้นตอนวิธีการคำนวณใหม่ของตัวจัดตารางเวลา ถ้ามีการระบุค่าระดับความสำคัญของเธรด น้อยกว่า 40 เธรดเหล่านั้นจะรันและเสร็จสมบูรณ์ก่อนที่ เธรดผู้ใช้ใดๆ จะสามารถรันได้ ตัวอย่างเช่น เธรดที่มีค่าที่ระบุเป็น 10 จะรันก่อนเธรดที่มีค่าที่ระบุเป็น 15

ผู้ใช้สามารถใช้คำสั่ง **nice** เพื่อทำให้ระดับความสำคัญแบบไม่คงที่ ของเธรดมีความสำคัญน้อยลงได้ ผู้จัดการระบบสามารถใช้ค่า nice ติดลบ กับเธรด เพื่อทำให้เธรดนั้นมีระดับความสำคัญดีขึ้น

ภาพสาธิตต่อไปนี้แสดงวิธีการบางอย่างซึ่งสามารถเปลี่ยน ค่าระดับความสำคัญได้



รูปที่ 6. วิธีการกำหนดค่าระดับความสำคัญ. ภาพสาธิต แสดงวิธีการเปลี่ยนค่าระดับความสำคัญการจัดตารางเวลาของเธรด ในระหว่างการดำเนินการหรือหลังจากการใช้คำสั่ง nice ค่าระดับความสำคัญ ยิ่งน้อย ระดับความสำคัญของเธรดยิ่งสูง โดยแรกเริ่ม ค่า nice ดีฟอลต์คือ 20 และระดับความสำคัญพื้นฐานมีค่าดีฟอลต์เป็น 40 หลังจากการดำเนินการบางอย่างและ penalty ตัวประมวลผล ค่า nice ยังคงเป็น 20 และระดับความสำคัญพื้นฐานยังคงเป็น 40 หลังจากการรันคำสั่ง renice –5 และด้วยการใช้ตัวประมวลผลเท่าเดิมเหมือนแต่ก่อน ขณะนี้ค่า nice คือ 15 และระดับความสำคัญพื้นฐานยังคงเป็น 40 หลังจากออกใช้รูทีนย่อย setpri() ด้วยค่า 50 ขณะนี้ ระดับความสำคัญคงที่คือ 50 และค่า nice และ การใช้ตัวประมวลผลจะไม่เกี่ยวข้อง

ค่า nice ของเธรดมีการตั้งค่าเมื่อเธรดถูกสร้างขึ้น และเป็นค่าคงที่ ตลอดอายุการใช้งานของเธรด ยกเว้นว่าผู้ใช้เปลี่ยนค่านี้ด้วยตนเอง โดยใช้คำสั่ง **renice** หรือการเรียกระบบ **setpri( ), setpriority( ), thread_setsched( ),** หรือ **nice( )**

Penalty ตัวประมวลผลคือเลขจำนวนเต็มที่คำนวณจาก การใช้ตัวประมวลผลล่าสุดของเธรด การใช้ตัวประมวลผลล่าสุดเพิ่มขึ้นประมาณ 1 ในแต่ละครั้งที่เธรดอยู่ในความควบคุมของตัวประมวลผล เมื่อสิ้นสุดช่วงเวลา 10 ms และมีค่าสูงสุดคือ 120 Penalty ระดับความสำคัญจริงต่อช่วงเวลาเพิ่มขึ้นด้วยค่า **nice** หลังผ่านไปทุกวินาที ค่าการใช้ตัวประมวลผลล่าสุดสำหรับเธรดทั้งหมด จะมีการคำนวณใหม่

ผลลัพธ์เป็นดังต่อไปนี้:

- ระดับความสำคัญของเธรดที่มีระดับความสำคัญแบบไม่คงที่มีความสำคัญน้อยลง เมื่อการใช้ตัวประมวลผลล่าสุดเพิ่มขึ้น และในทางกลับกัน ลักษณะเช่นนี้หมายความว่า โดยเฉลี่ย ถ้าเวลาที่จัดสรรให้เธรดครั้งล่าสุดยิ่งมาก โอกาสที่เธรด จะได้รับจัดสรรเวลาครั้งถัดไป ยิ่งน้อย
- ระดับความสำคัญของเธรดที่มีระดับความสำคัญแบบไม่คงที่มีความสำคัญน้อยลง เมื่อค่า nice เพิ่มขึ้น และในทางกลับกัน

หมายเหตุ: ด้วยการใช้คิวการรันหลายตัวประมวลผลและกลไกการปรับสมดุลโหลด ค่า **nice** หรือ **renice** อาจไม่ให้ ผลต่อระดับความสำคัญของเธรดดังที่คาดไว้ เนื่องจากระดับความสำคัญที่ต้องการ น้อยลงอาจมีรันไทม์เท่ากับหรือมากกว่าระดับความสำคัญที่ ต้องการ เธรดที่ต้องการผลที่คาดไว้ของ **nice** หรือ **renice** ควรมีการวางไว้บนคิวการรันสากล

คุณสามารถใช้คำสั่ง **ps** เพื่อแสดงค่าระดับความสำคัญ ค่า nice และค่าการใช้ตัวประมวลผลระยะสั้นของกระบวนการ

โปรดดู “การควบคุม contention สำหรับไมโครโพรเซสเซอร์” ในหน้า 132 สำหรับรายละเอียด เพิ่มเติมเกี่ยวกับการใช้คำสั่ง **nice** และ **renice**

โปรดดู “การคำนวณค่าระดับความสำคัญของเธรด” ในหน้า 135 สำหรับรายละเอียด การคำนวน penalty ตัวประมวลผลและ decay ของค่าการใช้ตัวประมวลผลล่าสุด

กลไกระดับความสำคัญยังมีการใช้โดย AIX Workload Manager เพื่อบังคับใช้การจัดการ รีซอร์สตัวประมวลผล เนื่องจากเธรดประเภท Workload Manager มีการจัดการ ระดับความสำคัญโดย Workload Manager ดังนั้นเธรดดังกล่าวจึงอาจมีพฤติกรรมระดับความสำคัญแตกต่างไปจากเธรด ประเภท Workload Manager

นโยบายการจัดตารางเวลาสำหรับเธรด

นโยบายการจัดตารางเวลาประกอบด้วยค่าที่เป็นไปได้จำนวนมากสำหรับเธรด

SCHED_FIFO

หลังจากจัดตารางเวลาเธรดที่มีนโยบายนี้แล้ว เธรดจะรันจนสมบูรณ์ยกเว้นว่า ถูกบล็อก เธรดจะอนุญาตให้ควบคุมตัวประมวลผลโดยสมัครใจ หรือเธรดที่มีระดับ ความสำคัญสูงกว่าจะสามารถจัดส่งได้ เฉพาะเธรดที่มีระดับความสำคัญคงที่เท่านั้นสามารถมีนโยบายการจัดตารางเวลาแบบ SCHED_FIFO ได้

SCHED_RR

เมื่อเธรด SCHED_RR มีการควบคุมเมื่อสิ้นสุดส่วนเวลา เธรดจะย้าย ไปยังปลายของคิวของเธรดที่จัดส่งได้ซึ่งมีระดับความสำคัญเท่ากัน เฉพาะเธรดที่มีระดับความสำคัญคงที่เท่านั้นสามารถมีนโยบายการจัดตารางเวลาแบบ SCHED_RR ได้

SCHED_OTHER

นโยบายนี้มีการกำหนดโดยมาตรฐาน POSIX 1003.4a ตามที่กำหนดโดยการดำเนินการ การคำนวณค่าระดับความสำคัญของเธรดที่กำลังรันอยู่ใหม่ในการขัดจังหวะนาฬิกา แต่ละครั้งหมายความว่า เธรดอาจสูญเสียการควบคุมเนื่องจากค่าระดับความสำคัญของ เธรดเพิ่มขึ้นมากกว่าค่าของเธรดที่จัดส่งได้อื่น

SCHED_FIFO2

นโยบายนี้เหมือนกับ SCHED_FIFO ยกเว้นว่าอนุญาตให้เธรดซึ่งพักไว้ชั่วครู่หนึ่งสามารถถูกนำไปวางที่ตอนต้นของรันคิวได้เมื่อเธรดนั้น ถูกปลุกขึ้นมา รอบเวลานี้คือขีดจำกัด affinity (ปรับได้ด้วย **schedo -o affinity_lim**)

SCHED_FIFO3

เธรดที่มีการตั้งค่านโยบายการจัดตารางเวลาเป็น SCHED_FIFO3 จะถูกวาง ไว้ที่ตอนต้นของรันคิวเสมอ เพื่อป้องกันไม่ให้เธรดที่ใช้นโยบายการจัดตารางเวลา SCHED_FIFO2 ถูกวางไว้ก่อนหน้าเธรดของ SCHED_FIFO3 จึงมีการเปลี่ยนรันคิวพารามิเตอร์ เมื่อเธรด SCHED_FIFO3 ออกจากคิว เพื่อไม่ให้เธรดของ SCHED_FIFO2 ผ่านเกณฑ์และสามารถไปรวมอยู่ที่ตอนต้นของรันคิวได้

SCHED_FIFO4

เธรดในคลาสการจัดตารางเวลา SCHED_FIFO4 ที่มีลำดับความสำคัญสูงกว่าไม่มีสิทธิ เหนือกว่าเธรดลำดับความสำคัญต่ำที่กำลังรันอยู่ในปัจจุบันตราบเท่าที่ค่าลำดับ ความสำคัญแตกต่างกัน 1 ลักษณะการทำงานดีฟอลต์คือเธรดลำดับความสำคัญต่ำที่กำลังรันอยู่ใน ปัจจุบันมีสิทธิใน CPU ที่กำหนดก่อนเธรดลำดับความสำคัญสูงกว่าที่เพิ่งจะมีสิทธิรันบนตัวประมวลผลเดียวกัน

นโยบายการจัดตารางเวลามีการตั้งค่าโดยใช้การเรียกระบบ **thread_setsched( )** และมีผลเฉพาะสำหรับเธรดที่เรียกเท่านั้น อย่างไรก็ตาม สามารถตั้งค่าเธรด ให้มีนโยบายการจัดตารางเวลา SCHED_RR ได้โดยการออกใช้การเรียก **setpri( )** ที่ระบุ ID กระบวนการ ส่วนผู้เรียกของ **setpri( )** และเป้าหมายของ **setpri( )** ไม่จำเป็นต้องตรงกัน

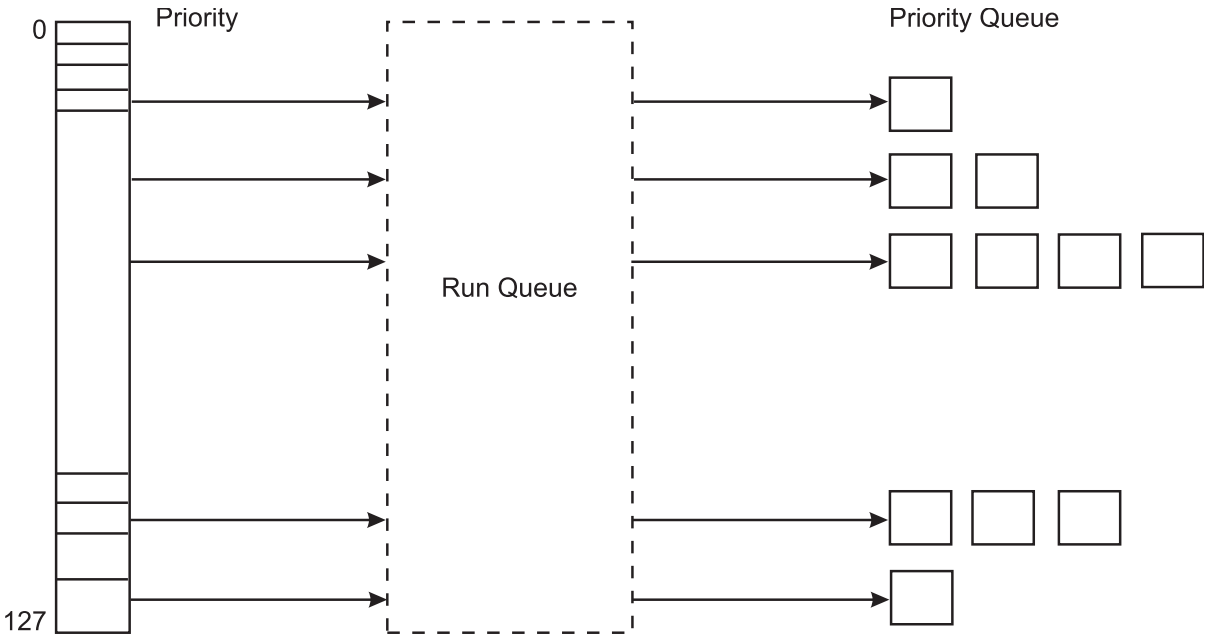
เฉพาะกระบวนการที่มีสิทธิรากเท่านั้นสามารถออกใช้การเรียกระบบ **setpri( )** ได้ เฉพาะเธรดที่มีสิทธิรากเท่านั้นสามารถเปลี่ยนนโยบายการจัดตารางเวลาเป็น อ็อพชัน SCHED_FIFO หรือ SCHED_RR ได้ ถ้านโยบายการจัดตารางเวลาเป็น SCHED_OTHER รูทีนย่อย **thread_setsched( )** จะละเว้นพารามิเตอร์ระดับความสำคัญ

เธรดเป็นสิ่งที่มีความสำคัญสำหรับแอ็พพลิเคชันที่ประกอบด้วยกระบวนการ แบบอะซิงโครนัสหลายกระบวนการในปัจจุบัน แอ็พพลิเคชันเหล่านี้อาจมีโหลดบนระบบที่เบาลง ถ้ามีการแปลงเป็นโครงสร้างแบบหลายเธรด

รันคิวของตัวจัดตารางเวลา

ตัวจัดตารางเวลาเก็บรักษารันคิวของเธรดทั้งหมด ที่พร้อมจะถูกจัดส่ง

ภาพสาธิตต่อไปนี้แสดงรันคิวโดยใช้สัญลักษณ์



รูปที่ 7. รันคิว. ภาพสาธิตนี้เพียงแต่แสดงให้เห็นว่าเธรด ที่มีค่าระดับความสำคัญต่ำถูกส่งผ่านรันคิวก่อนเธรดที่มีค่าระดับความสำคัญสูงกว่าได้อย่างไร ช่วงของค่าลำดับความสำคัญที่เป็นไปได้ คือ 0 ถึง 127 ซึ่งสัมพันธ์โดยตรงกับจำนวนรันคิวทั้งหมด 128 คิว

เธรดที่จัดส่งได้ทั้งหมดซึ่งมีลำดับความสำคัญตามที่กำหนดมีตำแหน่งอยู่ใน รันคิว

เอนทิตีที่จัดส่งได้พื้นฐานของตัวจัดตารางเวลาคือเธรด AIX เก็บรักษา 256 รันคิว รันคิวสัมพันธ์โดยตรงกับช่วงของค่าที่เป็นไปได้ (0 ถึง 255) สำหรับฟิลด์ลำดับความสำคัญของแต่ละเธรด วิธีการนี้ช่วยให้ตัวจัดตารางเวลา กำหนดเธรดที่ต้องการรันมากที่สุดได้ ง่ายขึ้น โดยไม่ต้องค้นหารันคิวขนาดใหญ่คิวเดียว ตัวจัดตารางเวลาใช้ mask ที่มีบิตอยู่ เพื่อบ่งชี้การมีอยู่ของเธรดที่พร้อมจะรัน ในรันคิวที่ตรงกัน

ค่าระดับความสำคัญของเธรดเปลี่ยนอย่างรวดเร็วและบ่อยครั้ง การเคลื่อนย้ายที่สม่ำเสมอเป็นผลมาจากการที่ตัวจัดตารางเวลาคำนวณลำดับ ความสำคัญใหม่ อย่างไรก็ตาม เธรดที่มีระดับความสำคัญคงที่ไม่ได้มีลักษณะเช่นนี้

เริ่มต้นด้วย AIX Version 6.1 แต่ละ ตัวประมวลผลมีหนึ่งรันคิวต่อโหนด ค่ารันคิวที่รายงานในเครื่องมือวัด ประสิทธิภาพเป็นผลรวมของเธรดทั้งหมดในแต่ละ รันคิว การมีรันคิวสำหรับแต่ละตัวประมวลผลช่วยประหยัดโอเวอร์เฮดในการจัดส่ง ล็อก และพัฒนา affinity ตัวประมวลผลโดยรวม เธรดมีแนวโน้มจะอยู่บน ตัวประมวลผลเดียวกันบ่อยมากขึ้น ถ้าเธรดสามารถ

ดำเนินการได้แล้ว เนื่องจากเหตุการณ์ของตัวประมวลผลอื่นที่ไม่ใช่เธรดที่ดำเนินการ ได้ซึ่งรันอยู่ ผลคือเธรดนั้นจะถูกจัดส่งในทันทีถ้ามีตัวประมวลผล ที่ไม่ได้ทำงาน ไม่มีการได้สิทธิเกิดขึ้นจนกว่าสามารถตรวจสอบสถานะของตัวประมวลผลได้ เช่น การขัดจังหวะบนตัวประมวลผลของเธรด

บนระบบแบบหลายตัวประมวลผลที่มีหลายรันคิว อาจเกิดการแปลงระดับ ความสำคัญชั่วครู่ขึ้น อาจเป็นไปได้ตลอดเวลาที่รันคิวหนึ่งอาจมี หลายเธรดซึ่งมีลำดับความสำคัญที่ต้องการมากกว่า รันคิวอื่น AIX มีกลไก สำหรับการปรับสมดุลลำดับความสำคัญเมื่อเวลาผ่านมา แต่ถ้าต้องการลำดับความ สำคัญที่แท้จริง (ตัวอย่างอย่าง สำหรับเรียลไทม์แอ็พพลิเคชัน) มีตัวแปรสภาวะแวดล้อมที่ชื่อว่า *RT_GRQ* ตัวแปรสภาวะแวดล้อม *RT_GRQ* เมื่อมีการตั้งค่าเป็น ON จะทำให้เธรดอยู่บนโกลบอลรันคิว ในกรณีนี้โกลบอลรันคิวจะถูกค้นหาเพื่อดูเธรดที่มี ลำดับความสำคัญที่ดีที่สุด ตัวแปรนี้สามารถช่วยปรับปรุงประสิทธิภาพสำหรับเธรดที่มี การขัดจังหวะ เธรดซึ่งกำลังรันที่ลำดับความสำคัญคงที่จะถูกวางไว้บนโกลบอลรันคิว ถ้าพารามิเตอร์ *fixed_pri_global* ของคำสั่ง **schedo** มีการตั้งค่าเป็น 1

จำนวนเธรดเฉลี่ยในรันคิวสามารถดูได้ในคอลัมน์แรกของ เอาต์พุตคำสั่ง **vmstat** ถ้าคุณ หารจำนวนนี้ด้วยจำนวนของตัวประมวลผล ผลลัพธ์คือจำนวนเฉลี่ย ของเธรดที่สามารถรันบนตัวประมวลผลแต่ละตัว ถ้าค่านี้ มากกว่าหนึ่ง เธรดเหล่านี้ต้องรอรอบตัวประมวลผลของตน โดยตัวเลขยิ่งมาก โอกาสจะเห็นประสิทธิภาพที่ด้อยลง ยิ่งมาก

เมื่อเธรดเลื่อนไปยังส่วนท้ายของรันคิว (ตัวอย่างเช่น เมื่อเธรด มีการควบคุมที่ตอนท้ายของส่วนเวลา) เธรดจะย้ายไปตำแหน่งหลังจาก เธรดล่าสุดในคิวซึ่งมีค่าระดับความสำคัญ เท่ากัน

ส่วนเวลาของตัวประมวลผลตัวจัดตารางเวลา

ส่วนเวลาของตัวประมวลผลคือจำนวนเวลาที่เธรด SCHED_RR สามารถ ใช้ ก่อนที่ตัวจัดตารางเวลาจะสลับไปยังเธรดอื่นที่มีระดับความสำคัญเท่ากัน

คุณสามารถใช้อ็อพชัน **timeslice** ของคำสั่ง **schedo** เพื่อเพิ่มจำนวนของ clock ticks ในส่วนเวลาทีละ 10 มิลลิวินาทีได้ (โปรดดู “การแก้ไขการแบ่งเวลาของตัวกำหนดตารางเวลาด้วยคำสั่ง schedo” ในหน้า 138)

หมายเหตุ: ส่วนเวลาไม่ใช่จำนวนที่รับประกัน ของเวลาตัวประมวลผล แต่เป็นเวลานานที่สุดซึ่งเธรดสามารถควบคุมได้ ก่อนมีโอกาสถูกแทนที่โดยเธรด อื่น เธรดอาจสูญเสียการควบคุมตัวประมวลผลได้หลายวิธี ก่อนที่เธรดจะมีการควบคุมส่วนเวลาทั้งหมด

การสับเปลี่ยนโหมด

การประมวลผลผู้ใช้ประสบกับการสับเปลี่ยนโหมด เมื่อจำเป็นต้องเข้าถึงรีซอร์สของระบบ ซึ่งต้องนำไปปฏิบัติผ่านอินเตอร์เฟสการเรียกระบบ หรือด้วยการอินเตอร์รัปต์ เช่น ข้อบกพร่องของเพจ

มีโหมดอยู่ด้วยกันสองโหมด:

- โหมดผู้ใช้
- โหมดเคอร์เนล

เวลาที่ตัวประมวลผลใช้ในโหมดผู้ใช้ (แอ็พพลิเคชันและไลบรารีที่แบ่งใช้) จะสะท้อนถึงเวลาของผู้ใช้ในเอาต์พุตของคำสั่ง เช่น คำสั่ง **vmstat, iostat** และ **sar** เวลาที่ตัวประมวลผลใช้ในโหมดเคอร์เนล จะสะท้อนถึงเวลาของระบบในเอาต์พุตของคำสั่งเหล่านี้

โหมดผู้ใช้:

โปรแกรมที่ดำเนินการในโดเมนการป้องกันผู้ใช้เป็นกระบวนการผู้ใช้

รหัสที่ดำเนินการในโดเมนการป้องกันนี้ดำเนินการในโหมดการดำเนินการของ ผู้ใช้ และมีการเข้าถึงดังต่อไปนี้:
- การเข้าถึงการอ่าน/การบันทึกข้อมูลผู้ใช้ในพื้นที่ส่วนตัวของกระบวนการ
- การเข้าถึงการอ่านข้อความผู้ใช้และพื้นที่ข้อความแบบแบ่งใช้
- การเข้าถึงพื้นที่ข้อมูลแบบแบ่งใช้โดยใช้ฟังก์ชันหน่วยความจำแบบแบ่งใช้

โปรแกรมที่ดำเนินการในโดเมนการป้องกันผู้ใช้ไม่มีสิทธิเข้าถึง เคอร์นัลหรือเซกเมนต์ข้อมูลเคอร์นัล ยกเว้นการเข้าถึงทางอ้อมผ่านทางการใช้การ เรียกระบบ โปรแกรมในโดเมนการป้องกันนี้มีผลต่อสภาพแวดล้อมการดำเนินการของ โปรแกรมเองเท่านั้น และดำเนินการในกระบวนการหรือสถานะไม่ได้รับสิทธิพิเศษ

โหมดเคอร์เนล:

โปรแกรมที่เรียกใช้งานในโดเมนการปกป้องเคอร์เนลจะสอดแทรก interrupt handlers การประมวลผลเคอร์เนล เคอร์เนลพื้นฐาน และส่วนขยายเคอร์เนล (ไดร์เวอร์อุปกรณ์ การเรียกของระบบ และระบบไฟล์)

โดเมนการปกป้องนี้จะหมายถึงโค้ดจะเรียกใช้งานในโหมดการประมวลผลเคอร์เนล และมีสิทธิเข้าถึงต่อไปนี้:
- สิทธิการอ่าน/เขียนลงในพื้นที่แอดเดรสเคอร์เนลแบบโกลบอล
- สิทธิการอ่าน/เขียนข้อมูลเคอร์เนลในส่วนของการประมวลผล ขณะที่เรียกใช้งานภายในการประมวลผล

เคอร์เนลเซอร์วิสต้องถูกใช้เพื่อเข้าถึงข้อมูลผู้ใช้ภายใน พื้นที่แอดเดรสการประมวลผล

การเรียกใช้งานโปรแกรมในโดเมนการปกป้องนี้สามารถมีผลต่อสภาวะแวดล้อมการประมวลผลของโปรแกรมทั้งหมด เนื่องจากโปรแกรมจะมีคุณสมบัติต่อไปนี้:
- โปรแกรมสามารถเข้าถึงข้อมูลระบบแบบโกลบอล
- โปรแกรมสามารถใช้เคอร์เนลเซอร์วิส
- โปรแกรมได้รับการยกเว้นจากการยับยั้งการรักษาความปลอดภัย
- โปรแกรมจะเรียกใช้งานตัวประมวลผลในสถานะที่ได้รับสิทธิพิเศษ

การสับเปลี่ยนโหมด:

การใช้ของการเรียกระบบด้วยการประมวลผลในโหมดผู้ใช้อนุญาตให้ฟังก์ชันเคอร์เนล ถูกเรียกจากโหมดของผู้ใช้ การเข้าถึงฟังก์ชันที่เรียกใช้การเรียกของระบบโดยตรง หรือโดยอ้อมจะถูกจัดเตรียมไว้โดยไลบรารีโปรแกรมมิง ซึ่งจัดเตรียมการเข้าถึงฟังก์ชันของระบบปฏิบัติการ

การสับเปลี่ยนโหมดควรมีความแตกต่างจาก context switch ที่มองเห็นได้ในเอาต์พุตของคำสั่ง **vmstat** (คอลัมน์ **cs**) และ **sar** (*cswch/s*) context switch จะเกิดขึ้นเมื่อการรันเธรดในปัจจุบันแตกต่างจากการรันเธรดก่อนหน้านี้ บนตัวประมวลผลนั้น

ตัวกำหนดตารางเวลาจะดำเนินการกับ context switch เมื่อเหตุการณ์ต่อไปนี้เกิดขึ้น:
- เธรดต้องรอรีซอร์ส (ตามความสมัครใจ) เช่น ดิสก์ I/O เน็ตเวิร์ก I/O sleep หรือล็อก
- เธรดที่มีระดับความสำคัญที่สูงกว่าจะตื่นขึ้น (โดยอัตโนมัติ)

- เธรดจะถูกใช้ตามการแบ่งเวลา (โดยปกติ 10 มิลลิวินาที)

เวลาของ Context switch การเรียกของระบบ การอินเตอร์รัปต์อุปกรณ์ NFS I/O และกิจกรรมอื่นใดในเคอร์เนลจะถูกนำมาพิจารณาเป็นเวลาระบบ

ประสิทธิภาพผู้จัดการหน่วยความจำเสมือน

พื้นที่ว่างที่อยู่เสมือนมีการแบ่งพาร์ติชันเป็นเซกเมนต์ เซกเมนต์คือ ส่วนที่ติดกันขนาด 256 MB ของพื้นที่ว่างที่อยู่หน่วยความจำเสมือนซึ่ง สามารถแม็พอ็อบเจ็กต์ข้อมูลได้

กระบวนการระบุที่อยู่ของข้อมูลมีการจัดการที่ระดับเซกเมนต์ (หรืออ็อบเจ็กต์) เพื่อให้สามารถแบ่งใช้เซกเมนต์ระหว่างกระบวนการหรือรักษาเป็นส่วนตัวได้ ตัวอย่างเช่น กระบวนการสามารถแบ่งใช้โค้ดเซกเมนต์ที่มีเซกเมนต์ข้อมูลแยก ต่างหากและส่วนตัวได้

การจัดการหน่วยความจำจริง

VMM มีบทบาทสำคัญในการจัดการหน่วยความจำจริง

เซกเมนต์หน่วยความจำเสมือนมีการแบ่งพาร์ติชันเป็นหน่วยขนาดคงที่ ที่เรียกว่า *หน้า* AIX 7.1 ที่กำลังรันบนตัวประมวลผล POWER5+ สนับสนุนหน้าสี่ขนาด คือ: 4 KB, 64 KB, 16 MB, และ 16 GB หากต้องการข้อมูลเพิ่มเติม ให้ดู การสนับสนุนหลายขนาดหน้า แต่ละหน้าในเซกเมนต์อาจอยู่ในหน่วยความจำจริง (RAM) หรือถูกจัดเก็บไว้บนดิสก์จนกว่าจะต้องการใช้ในลักษณะคล้ายกัน หน่วยความจำจริงมีการแบ่งออกเป็น เฟรมหน้า บทบาทของ VMM คือการจัดการจัดสรรเฟรมหน้าหน่วยความจำจริง และแก้ไขการอ้างอิงโดยโปรแกรมไปยังหน้าหน่วยความจำเสมือน ที่ไม่ได้อยู่ในหน่วยความจำจริงในปัจจุบันหรือยังไม่มีอยู่ (ตัวอย่างเช่น เมื่อ กระบวนการอ้างอิงครั้งแรกไปยังหน้าของเซกเมนต์ข้อมูลของกระบวนการ)

เนื่องจากจำนวนของหน่วยความจำเสมือนที่ใช้อยู่ ณ ขณะหนึ่งอาจมากกว่า หน่วยความจำจริง VMM จึงต้องจัดเก็บส่วนเกินไว้บนดิสก์ จากจุดยืน ด้านประสิทธิภาพ VMM มีวัตถุประสงค์ซึ่งตรงข้ามกันอยู่สองข้อคือ:

- ลดเวลาตัวประมวลผลโดยรวมและต้นทุนดิสก์แบนด์วิธของการใช้หน่วยความจำ เสมือนให้เหลือน้อยที่สุด
- ลดต้นทุนเวลาตอบกลับของ page faults ให้เหลือน้อยที่สุด

เพื่อให้เป็นไปตามวัตถุประสงค์เหล่านี้ VMM จึงจัดทำ *รายการที่ว่าง* ของ เฟรมหน้าที่มีอยู่เพื่อใช้สำหรับ page fault VMM ใช้ขั้นตอนวิธีการเปลี่ยนหน้า เพื่อกำหนดว่า หน้าหน่วยความจำเสมือนที่อยู่ในหน่วยความจำในปัจจุบันใดจะต้องมี การกำหนดเฟรมหน้าของหน้าเหล่านั้นอีกครั้งในรายการที่ว่าง ขั้นตอนวิธี การเปลี่ยนหน้าใช้กลไลหลายอย่างดังนี้:

- เซกเมนต์หน่วยความจำเสมือนมีการจัดประเภทเป็นเซกเมนต์ถาวร หรือเซกเมนต์ที่กำลังทำงาน อย่างใดอย่างหนึ่ง
- เซกเมนต์หน่วยความจำเสมือนมีการจัดประเภทตามข้อมูลที่มีอยู่ เป็นหน่วยความจำ สำหรับการคำนวณหรือไฟล์
- หน้าหน่วยความจำเสมือนที่เป็นเจ้าของการเข้าถึงซึ่งทำให้เกิด page fault จะถูกติดตาม
- Page faults มีการจัดประเภทเป็น page faults ใหม่หรือ repage faults
- มีการเก็บสถิติเกี่ยวกับอัตราของ repage faults ในแต่ละเซกเมนต์หน่วย ความจำเสมือน
- Thresholds ที่ผู้ใช้ปรับได้มีผลต่อการตัดสินใจของขั้นตอนวิธีการเปลี่ยนหน้า

รายการที่ว่าง:

VMM จะรักษารายการของกรอบเพจแบบโลจิคัลที่ว่างอยู่ ซึ่งจะใช้เพื่อทำให้เหมาะสมกับข้อบกพร่องของเพจ

ในสภาวะแวดล้อมส่วนใหญ่ VMM ต้องเพิ่มให้กับรายการที่ว่างเป็นครั้งคราว โดยการกำหนดกรอบเพจบางอย่างที่เป็นเจ้าของโดยการประมวลผลที่รันอยู่ เพจหน่วยความจำเสมือนที่มีกรอบเพจ ที่ต้องการกำหนดใหม่จะถูกเลือกโดยอัลกอริธึมการแทนที่เพจของ VMM VMM threshold จะเป็นตัวกำหนดจำนวนของกรอบที่ได้กำหนดไว้ใหม่

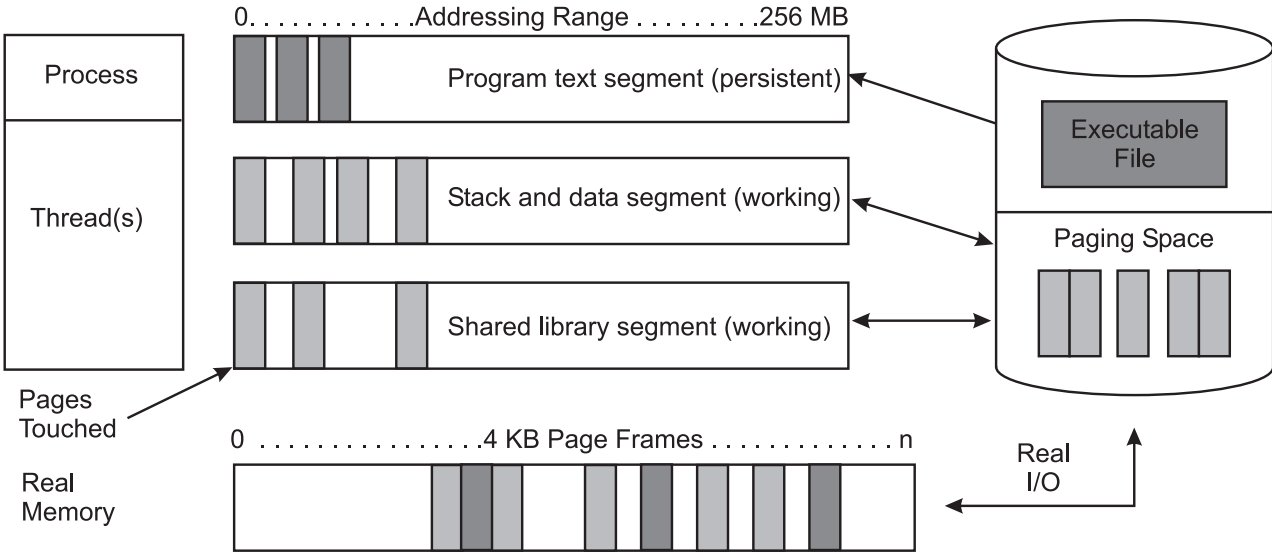
เซกเมนต์ถาวรและที่กำลังทำงาน:

เซกเมนต์ถาวรเป็นข้อมูลถาวรในขณะที่เซกเมนต์ที่กำลังทำงาน เป็นข้อมูลชั่วคราว

หน้าของเซกเมนต์ถาวรมีที่ตั้งหน่วยเก็บถาวร บนดิสก์ ไฟล์ที่มีข้อมูลหรือโปรแกรมที่ดำเนินการได้จะถูกแม็พเข้ากับ เซกเมนต์ถาวร เนื่องจากแต่ละหน้าของเซกเมนต์ถาวรมี ที่ตั้งหน่วยเก็บดิสก์ถาวร VMM จะบันทึกหน้ากลับไปยังที่ตั้งนั้น เมื่อหน้ามีการเปลี่ยนแปลงและไม่สามารถเก็บไว้ในหน่วยความจำจริงได้ อีกต่อไป ถ้าหน้าไม่มีการเปลี่ยนแปลงเมื่อเลือกสำหรับการวางในรายการที่ว่าง จะไม่ต้องการ I/O ถ้าหน้ามีการอ้างอิงอีกครั้งในภายหลัง จะมีการอ่านสำเนาใหม่จากที่ตั้งหน่วยเก็บดิสก์ ถาวร

เซกเมนต์ที่กำลังทำงานเป็นข้อมูลชั่วคราว และมีอยู่เฉพาะในระหว่างที่ถูกใช้งาน โดยกระบวนการเท่านั้น และไม่มีที่ตั้งหน่วยเก็บดิสก์ถาวร สแต็คกระบวนการ และพื้นที่ข้อมูลจะถูกแม็พเข้ากับเซกเมนต์ที่กำลังทำงาน เช่น เซกเมนต์ข้อความ เคอร์นัล เซกเมนต์ข้อความส่วนขยายเคอร์นัล ตลอดจนเซกเมนต์ข้อความไลบรารี แบบแบ่งใช้และข้อมูล หน้าของเซกเมนต์ที่กำลังทำงานยังต้องมีที่ตั้งหน่วยเก็บดิสก์ ซึ่งจะใช้เมื่อไม่สามารถเก็บเซกเมนต์นั้นไว้ในหน่วยความจำจริงได้ พื้นที่ว่างการเพจดิสก์ใช้สำหรับวัตถุประสงค์นี้

ภาพสาธิตต่อไปนี้แสดงความสัมพันธ์ระหว่างเซกเมนต์บางชนิดและ ที่ตั้งของหน้าเซกเมนต์นั้นบนดิสก์ นอกจากนี้ ยังแสดงที่ตั้งจริง (กำหนดเอง) ของหน้าเมื่อหน้าอยู่ใน หน่วยความจำจริง



รูปที่ 8. เซกเมนต์หน่วยเก็บถาวรและที่กำลังทำงาน. ภาพสาธิตนี้แสดงความสัมพันธ์ระหว่างเซกเมนต์บางชนิดและ ที่ตั้งของหน้าเซกเมนต์นั้นบนดิสก์ นอกจากนี้ ยังแสดงที่ตั้งจริง (กำหนดเอง) ของหน้า เมื่อหน้าอยู่ในหน่วยความจำจริง เซกเมนต์ที่กำลังทำงาน เป็นข้อมูลชั่วคราว ซึ่งหมายความว่ามีอยู่เฉพาะในระหว่างที่ถูกใช้งานโดย กระบวนการเท่านั้น และไม่มีที่ตั้งหน่วยเก็บดิสก์ถาวร สแต็คกระบวนการและพื้นที่ข้อมูลจะถูกแม็พเข้ากับเซกเมนต์ที่กำลังทำงาน เช่น เซกเมนต์ข้อความ เคอร์นัล เซกเมนต์ข้อความส่วนขยายเคอร์นัล และเซกเมนต์ข้อความไลบรารี แบบแบ่งใช้และข้อมูล หน้าของเซกเมนต์ที่กำลังทำงานยังต้องมีที่ตั้งหน่วยเก็บดิสก์ ซึ่งจะใช้เมื่อไม่สามารถเก็บเซกเมนต์นั้นไว้ในหน่วยความจำจริงได้ พื้นที่ว่างการเพจดิสก์ใช้สำหรับวัตถุประสงค์นี้

เซกเมนต์ชนิดถาวรยังมีการจัดประเภทเพิ่มเติมอีก *ไคลเอ็นต์เซกเมนต์* ใช้เพื่อ แม็พรีโมตไฟล์ (ตัวอย่างเช่น ไฟล์ที่กำลังมีการเข้าถึงผ่านทาง NFS) รวมถึงโปรแกรมที่ดำเนินการได้แบบรีโมต หน้าจากไคลเอ็นต์ เซกเมนต์มีการบันทึกและเรียกคืนบนเครือข่ายจากที่ตั้งไฟล์ถาวรของเซกเมนต์ นั้น ไม่ใช่จากพื้นที่ว่างการเพจดิสก์แบบโลคัล *เซกเมนต์ที่เจอร์นัลและ เลื่อนออกไป* เป็นเซกเมนต์ถาวรที่ต้องอัพเดต โดยสมบูรณ์ ถ้าหน้าจากเซกเมนต์ที่เจอร์นัลหรือเลื่อนออกไปถูกเลือกเพื่อลบออกจาก หน่วยความจำจริง (paged out) หน้านั้นต้องถูกบันทึกลงในพื้นที่ว่างการเพจดิสก์ ยกเว้นว่าหน้าอยู่ในสถานะที่อนุญาตให้ committed ได้ (บันทึกลงในที่ตั้งไฟล์ถาวรของเซกเมนต์นั้น)

การคำนวณเปรียบเทียบกับหน่วยความจำไฟล์:

หน่วยความจำที่ได้จากการคำนวณ ซึ่งรู้จักกันในชื่อของเพจที่ได้จากการคำนวณ ประกอบด้วยเพจที่เป็นของเซ็กเมนต์หน่วยเก็บใช้งาน หรือเซ็กเมนต์โปรแกรมข้อความ (ไฟล์เรียกทำงาน)

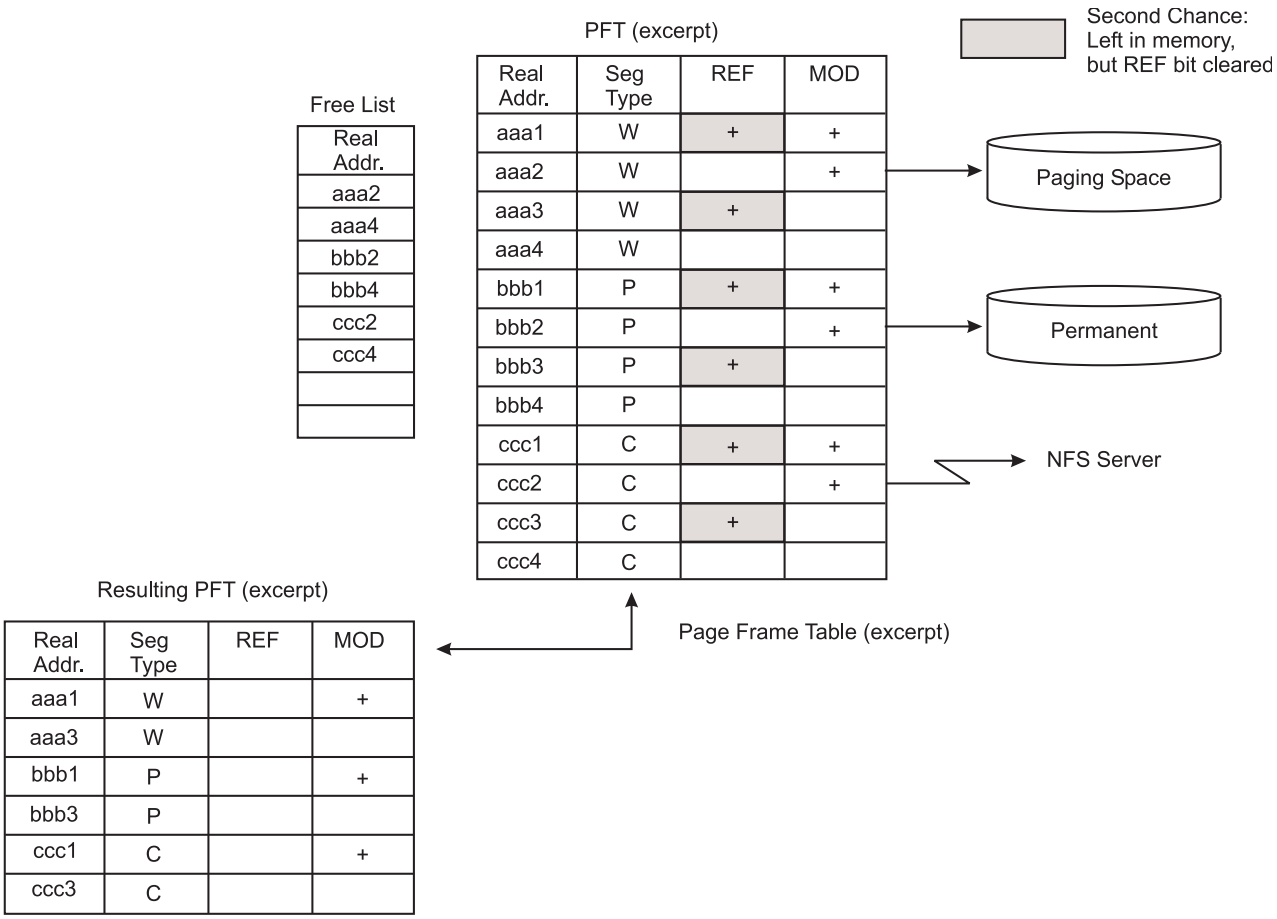
หน่วยความจำไฟล์ (หรือเพจไฟล์) ประกอบด้วยเพจที่เหลืออยู่ เพจเหล่านี้ คือเพจจากไฟล์ข้อมูลแบบถาวรในหน่วยเก็บที่ยังอยู่มีอยู่

การเปลี่ยนหน้า:

เมื่อจำนวนของเฟรมหน่วยความจำจริงที่มีอยู่ในรายการที่ว่าง เริ่มต่ำลง จะมีการเรียกใช้ตัวขโมยหน้า ตัวขโมยหน้าเคลื่อน ผ่าน Page Frame Table (PFT) เพื่อมองหาหน้าที่จะขโมย

PFT มีแฟล็กที่ส่งสัญญาณว่าหน้าใดมีการอ้างอิงแล้วและหน้าใด มีการแก้ไขไปแล้ว ถ้าตัวขโมยหน้าพบหน้าที่มีการอ้างอิงแล้ว ตัวขโมยหน้าจะไม่ขโมยหน้านั้น แต่จะรีเซ็ตแฟล็กอ้างอิงสำหรับ หน้านั้นแทน ในครั้งถัดไปที่ clock hand (ตัวขโมยหน้า) ผ่านหน้านั้นและบิตอ้างอิงยังคงปิดอยู่ หน้านั้นจะถูกขโมย หน้าซึ่งยังไม่ได้ถูกอ้างอิงในการผ่านครั้งแรก จะถูกขโมยในทันที

แฟล็กแก้ไขบ่งชี้ว่าข้อมูลบนหน้านั้นมีการเปลี่ยนแปลงแล้ว ตั้งแต่นำหน้านั้นเข้าในหน่วยความจำ เมื่อหน้าจะถูกขโมย ถ้า มีการตั้งค่าแฟล็กแก้ไข จะมีการเรียก pageout ก่อนที่จะขโมย หน้า หน้าที่เป็นส่วนประกอบหนึ่งของเซกเมนต์ซึ่งทำงานอยู่จะถูกบันทึกลงในพื้นที่ว่าง การเพจ ส่วนเซกเมนต์ถาวรจะถูกบันทึกลงในดิสก์



รูปที่ 9. ตัวอย่างการเปลี่ยนหน้า. ภาพสาธิตประกอบด้วย ข้อความบางส่วนจากสามตาราง ตารางแรกคือตารางเฟรมหน้า ที่มีสี่คอลัมน์ซึ่งประกอบด้วยที่อยู่จริง ชนิดเซกเมนต์ แฟล็กอ้างอิง และแฟล็กแก้ไข ตารางที่สองเรียกว่าตาราง รายการที่ว่างและมีที่อยู่ของหน้าทั้งหมดที่ว่าง ตารางสุดท้าย คือตารางเฟรมหน้าผลลัพธ์หลังจากลบที่อยู่ที่ว่าง ทั้งหมดออกไปแล้ว

นอกเหนือจากการเปลี่ยนหน้าแล้ว ขั้นตอนวิธีจะเก็บรักษาประวัติของทั้ง page faults ใหม่ (อ้างอิงเป็นครั้งแรก) และ repage faults (หน้าอ้างอิงที่ paged out แล้ว) โดยใช้บัฟเฟอร์ประวัติ ที่มี IDs ของ page faults ล่าสุด หลังจากนั้น จะพยายามปรับสมดุลระหว่าง page outs ไฟล์ (ข้อมูลถาวร) กับ page outs การคำนวณ (หน่วยเก็บหรือข้อความโปรแกรมที่ทำงาน)

เมื่อกระบวนการเสร็จสิ้น หน่วยเก็บที่ทำงานจะถูกรีลีสในทันที และเฟรมหน่วยความจำที่เชื่อมโยงจะถูกใส่กลับไปในรายการที่ว่าง อย่างไรก็ตาม ไฟล์ใดๆ ที่กระบวนการเปิดไว้ยังคงสามารถอยู่ในหน่วยความจำต่อไปได้

การเปลี่ยนหน้ามีการทำโดยตรงภายในขอบเขตของเธรด ถ้ากำลังรันบนตัวประมวลผลเดียว บนระบบแบบหลายตัวประมวลผล การเปลี่ยนหน้ามีการทำ ผ่านทางกระบวนการ **lrud** เคอร์นัล ซึ่งถูกจัดส่งไปยัง CPU เมื่อใช้งานถึง *minfree* threshold กระบวนการเคอร์เนล **lrud** เป็นแบบมัลติเธรดที่มี หนึ่งเธรดต่อพูลหน่วยความจำ หน่วยความจำจริง ถูกแบ่งออกเป็นพูลหน่วยความจำที่มีขนาดเท่ากันตามจำนวนของ CPUs และจำนวนของ RAM จำนวนของพูลหน่วยความจำบนระบบสามารถทราบได้โดยการรันคำสั่ง **vmstat -v**

คุณสามารถใช้คำสั่ง **vmo -r -o mempools=** <*จำนวน ของพูลหน่วยความจำ*> เพื่อเปลี่ยนจำนวนของพูลหน่วยความจำ ที่จะตั้งค่าคอนฟิกเมื่อบูตระบบได้ ค่าของพารามิเตอร์ *minfree* และ *maxfree* ในเอาต์พุตคำสั่ง **vmo** คือผลรวมของพารามิเตอร์ *minfree* และ *maxfree* สำหรับแต่ละพูลหน่วยความจำ

Repaging:

Page fault ถือว่าเป็น page fault ใหม่หรือ repage fault อย่างใดอย่างหนึ่ง Page fault ใหม่เกิดขึ้นเมื่อไม่มีการอ้างอิงเร็กคอร์ดของหน้า เมื่อเร็วๆ นี้ Repage fault เกิดขึ้นเมื่อหน้าที่ทราบกันว่ามีการ อ้างอิงไปแล้วเมื่อเร็วๆ นี้ถูกอ้างอิงอีกครั้งหนึ่ง และไม่พบในหน่วยความจำเนื่องจาก หน้าถูกเปลี่ยนแล้ว (และบางทีถูกบันทึกลงในดิสก์) ตั้งแต่ เข้าถึงหน้านั้นครั้งล่าสุด

นโยบายการเปลี่ยนหน้าที่สมบูรณ์อาจตัด repage faults ได้ทั้งหมด (สมมุติว่าหน่วยความจำจริงสมบูรณ์) โดยการขโมยเฟรมจากหน้าที่จะไม่อ้างอิง อีกครั้งหนึ่งเสมอ ดังนั้น จำนวนของ repage faults จึง เป็นตัววัดแบบผกผันกับประสิทธิผลของขั้นตอนวิธีการเปลี่ยนหน้า เพื่อรักษาหน้าที่ใช้บ่อยไว้ในหน่วยความจำ ด้วยเหตุนี้การลดความต้องการ I/O โดยรวม อาจช่วยพัฒนาประสิทธิภาพระบบได้

เพื่อจัดประเภท page fault เป็นแบบใหม่หรือ repage VMM จะใช้บัฟเฟอร์ ประวัติ repage ที่มีข้อมูล IDs หน้าของ *N* page faults ล่าสุด โดยที่ *N* คือจำนวนเฟรม ที่หน่วยความจำสามารถเก็บไว้ได้ ตัวอย่างเช่น หน่วยความจำ 512 MB ต้องการบัฟเฟอร์ประวัติ repage 128 KB เมื่อ page-in ถ้าพบ ID ของหน้าในบัฟเฟอร์ประวัติ repage ID นั้นจะถูกนับเป็น repage นอกจากนี้ VMM ยังประเมินอัตรา repaging หน่วยความจำสำหรับ คำนวณและอัตรา repaging หน่วยความจำไฟล์แยกกัน โดยการเก็บรักษา จำนวนของ repage faults สำหรับหน่วยความจำแต่ละชนิด อัตรา repaging มีการคูณด้วย 0.9 ในแต่ละครั้งที่ขั้นตอนวิธีการเปลี่ยนหน้ารัน เพื่อให้สะท้อนถึงกิจกรรม repaging ล่าสุดมากกว่ากิจกรรม repaging ในอดีต

VMM thresholds:

Thresholds ตัวเลขหลายรายการกำหนดวัตถุประสงค์ของ VMM เมื่อมีการฝ่าฝืน thresholds อย่างใดอย่างหนึ่ง VMM จะใช้การดำเนินการที่ เหมาะสมเพื่อให้สถานะของหน่วยความจำกลับมาอยู่ภายในขอบเขต ส่วนนี้อธิบาย thresholds ที่ผู้ดูแลระบบสามารถเปลี่ยนโดยใช้คำสั่ง **vmo**

จำนวนของ page frames บนรายการที่ว่างมีการควบคุมโดยพารามิเตอร์ ต่อไปนี้:

minfree
จำนวนต่ำสุดที่ยอมรับได้ของ page frames หน่วยความจำจริงในรายการที่ว่าง เมื่อขนาดของรายการที่ว่างลดต่ำกว่าจำนวนนี้ VMM จะเริ่มการขโมย หน้า มีการขโมยหน้าต่อไปจนกว่าขนาดของรายการที่ว่างเพิ่มถึง *maxfree*

maxfree
ขนาดสูงสุดซึ่งรายการที่ว่างจะเพิ่มขึ้นโดยการขโมยหน้า VMM ขนาด ของรายการที่ว่างอาจเกินกว่าจำนวนนี้อันเป็นผลมาจากการยุติกระบวนการ และการปล่อยหน้าเซกเมนต์การทำงานให้เป็นอิสระ หรือการลบไฟล์ที่มี หน้าในหน่วยความจำ

VMM พยายามรักษาขนาดของรายการที่ว่างให้มากกว่าหรือเท่ากับ *minfree* เมื่อ page faults หรือความต้องการระบบทำให้ขนาดรายการที่ว่างลดต่ำกว่า *minfree* ขั้นตอนวิธีการเปลี่ยน หน้าจะรัน ต้องรักษาขนาดของพื้นที่ว่างให้มากกว่าบางระดับ (ค่าดีฟอลต์ของ *minfree*) ด้วยเหตุผลหลายอย่าง ตัวอย่างเช่น ขั้นตอนวิธี sequential-prefetch ของระบบปฏิบัติการต้องการ หลาย frames ในแต่ละครั้งสำหรับแต่ละกระบวนการที่กำลังทำการอ่านตามลำดับ นอกจากนี้ VMM ยังต้องหลีกเลี่ยง deadlock ภายในตัวระบบปฏิบัติการเอง ซึ่งอาจเกิดขึ้นถ้ามีพื้นที่ว่างไม่เพียงพอที่จะ read in หน้าที่ต้องการสำหรับ การปล่อย page frame

Thresholds ต่อไปนี้มีการระบุเป็นเปอร์เซ็นต์ เปอร์เซ็นต์แสดงถึงเศษส่วน ของหน่วยความจำจริงทั้งหมดของเครื่องที่ใช้โดยหน้า ไฟล์ (หน้าของเซกเมนต์ที่ไม่ได้ใช้คำนวณ)

minperm

ถ้าเปอร์เซ็นต์ของหน่วยความจำจริงที่ใช้โดยหน้าไฟล์ลดต่ำกว่าระดับนี้ ขั้นตอนวิธีการเปลี่ยนหน้าจะขโมยทั้งหน้าไฟล์และหน้าการคำนวณ โดยไม่คำนึงถึงอัตรา repage

maxperm

ถ้าเปอร์เซ็นต์ของหน่วยความจำจริงที่ใช้โดยหน้าไฟล์เพิ่มเกินกว่าระดับนี้ ขั้นตอนวิธีการเปลี่ยนหน้าจะขโมยเฉพาะหน้าไฟล์

maxclient

ถ้าเปอร์เซ็นต์ของหน่วยความจำจริงที่ใช้โดยหน้าไฟล์เพิ่มเกินกว่าระดับนี้ ขั้นตอนวิธีการเปลี่ยนหน้าจะขโมยเฉพาะหน้าไคลเอ็นต์

เมื่อเปอร์เซ็นต์ของหน่วยความจำจริงที่ใช้โดยหน้าไฟล์อยู่ระหว่างค่าพารามิเตอร์ *minperm* และ *maxperm* โดยปกติ VMM จะขโมยเฉพาะหน้าไฟล์ แต่ถ้าอัตรา repaging ของหน้าไฟล์สูงกว่าอัตรา repaging ของหน้าการคำนวณ หน้าการคำนวณจะถูกขโมยด้วย

วัตถุประสงค์หลักของขั้นตอนวิธีการเปลี่ยนหน้าคือเพื่อให้มั่นใจว่า หน้าการคำนวณถูกจัดการอย่างยุติกรรม ตัวอย่างเช่น การอ่านตามลำดับของไฟล์ข้อมูลยาวเข้าใน หน่วยความจำไม่ควรทำให้ต้องสูญเสียหน้าข้อความโปรแกรมที่มีแนวโน้มจะใช้ อีกครั้งในเร็วๆ นี้ การใช้ thresholds และอัตรา repaging ของขั้นตอนวิธี การเปลี่ยนหน้าช่วยให้มั่นใจว่าหน้าทั้งสองชนิดได้รับการจัดการอย่างยุติธรรม โดยเอนเอียงไปทางหน้าการคำนวณเล็กน้อย

ฟังก์ชันการควบคุมโหลดหน่วยความจำ VMM

กระบวนการต้องการหน้าหน่วยความจำจริงเพื่อดำเนินการ เมื่อกระบวนการ อ้างอิงหน้าหน่วยความจำเสมือนที่อยู่บนดิสก์ เนื่องจากถูก paged-out หรือไม่เคยอ่าน หน้าที่อ้างอิงต้องถูก paged-in และ โดยเฉลี่ย หนึ่งหน้าขึ้นไปต้องถูก paged out (ถ้าหน้าที่แทนที่มีการ แก้ไข) การสร้างการจราจร I/O และการถ่วงเวลาความคืบหน้าของกระบวนการ

ระบบปฏิบัติการพยายามขโมยหน่วยความจำจริงจากหน้าซึ่งไม่น่าจะมีการอ้างอิง ในอนาคตอันใกล้โดยใช้ขั้นตอนวิธีการเปลี่ยน หน้า ขั้นตอนวิธีการเปลี่ยนหน้าที่สำเร็จช่วยให้ระบบปฏิบัติการสามารถเก็บกระบวนการ ที่ใช้งานอยู่ไว้ในหน่วยความจำในจำนวนเพียงพอจะทำให้ CPU ยุ่ง แต่ในการแย่งชิง หน่วยความจำบางระดับ การไม่มีหน้าอาจเป็นตัวเลือกที่ดีสำหรับ paging out ไปยังดิสก์ เนื่องจากหน้าจะถูกนำมาใช้ใหม่ในอนาคตอันใกล้โดยชุดของ กระบวนการที่ใช้งานอยู่ สถานการณ์นี้ขึ้นอยู่กับข้อมูลดังต่อไปนี้:

- จำนวนหน่วยความจำทั้งหมดในระบบ
- จำนวนของกระบวนการ
- ความต้องการหน่วยความจำที่ผันแปรตามเวลาของแต่ละกระบวนการ
- ขั้นตอนวิธีการเปลี่ยนหน้า

เมื่อกรณีนี้เกิดขึ้น paging-in และ paging-out จะเกิดขึ้นอย่างต่อเนื่อง สภาพนี้ เรียกว่า *thrashing* Thrashing ส่งผลให้มี I/O ต่อเนื่องไปยัง paging disk และทำให้แต่ละกระบวนการพบ page fault เกือบในทันทีที่ ถูกจัดส่ง ส่งผลให้ไม่มีกระบวนการใดคืบหน้า ขึ้นมาก

ข้อเสียที่สุดของ thrashing คือ แม้ว่า thrashing อาจถูกทริกเกอร์โดยความต้องการสูงสุดคร่าวๆ แบบสุ่มในเวิร์กโหลด (เช่น ผู้ใช้ทุกรายของระบบ กดปุ่ม Enter ในวินาทีเดียวกัน) ระบบอาจ thrashing ต่อไปไม่สิ้นสุด

ระบบปฏิบัติการมีขั้นตอนวิธีการควบคุมโหลดหน่วยความจำที่ตรวจพบเวลา ซึ่งระบบเริ่มต้นที่จะ thrash แล้วพักกระบวนการที่ใช้งานอยู่ และถ่วงเวลา การเริ่มต้นกระบวนการใหม่ไปช่วงเวลาหนึ่ง มีพารามิเตอร์ห้าตัวตั้งค่า อัตราและขอบเขตสำหรับขั้นตอนวิธี ค่าดีฟอลต์ของพารามิเตอร์เหล่านี้ มีการเลือกเป็น "fail safe" ตลอดเวิร์กโหลด ใน AIX เวอร์ชัน 4 การควบคุม โหลดหน่วยความจำมีการปิดใช้งานโดยค่าดีฟอลต์บนระบบที่มีเฟรมหน่วยความจำ รวมมากกว่าหรือเท่ากับ 128 MB

อัลกอริธึมการควบคุมโหลดหน่วยความจำ:

กลไกการควบคุมโหลดหน่วยความจำจะประเมินผลว่า มีหน่วยความจำที่เพียงพอสำหรับชุดของการประมวลผลที่แอ็คทีฟหนึ่งครั้งต่อวินาที เมื่อตรวจพบเงื่อนไขของการ overcommit หน่วยความจำ การประมวลผลบางกระบวนการจะถูกหยุดทำงานชั่วคราว ลดจำนวนของการประมวลผลที่แอ็คทีฟลง และลดระดับของการ overcommit หน่วยความจำ

เมื่อการประมวลผลหยุดทำงานชั่วคราว เธรดทั้งหมดจะหยุดทำงานชั่วคราว เมื่อเธรดเข้าใกล้สถานะที่สามารถหยุดทำงานชั่วคราวได้ เพจของการประมวลผลที่หยุดทำงานชั่วคราวจะเสถียร และเพจออกโดยอัลกอริธึมการแทนที่เพจ รีลีสกรอบของเพจที่มีเพียงพอ เพื่ออนุญาตให้การประมวลผลที่แอ็คทีฟซึ่งเหลืออยู่ดำเนินการต่อ ระหว่างช่วงเวลาที่การประมวลผลที่มีอยู่ ถูกหยุดทำงานชั่วคราว การประมวลผลที่ถูกสร้างขึ้นใหม่จะยังคงถูกหยุดทำงานชั่วคราว การป้องกันการทำงานใหม่จากการเข้าสู่ระบบ การประมวลผลที่หยุดทำงานชั่วคราว จะถูกเรียกใช้งานอีกครั้งจนกว่าช่วงเวลาการส่งผ่านลำดับถัด ซึ่งไม่มีการ thrash เงื่อนไขที่จำเป็นยังคงมีอยู่ หากช่วงเวลาที่ปลอดภัยนี้ผ่านไป เธรดของการประมวลผลที่หยุดทำงานชั่วคราวจะถูกเรียกใช้งานอีกครั้ง

พารามิเตอร์การควบคุมโหลดหน่วยความจำ **schedo** ระบุข้อมูลต่อไปนี้:

- หน่วยความจำระบบ overcommit threshold (*v_repage_hi*)
- จำนวนวินาทีที่ต้องการเพื่อทำให้ช่วงนั้นปลอดภัย (*v_sec_wait*)
- หน่วยความจำการประมวลผลแต่ละตัว overcommit threshold โดยการประมวลผล ที่ผ่านการรับรองให้หยุดการเป็นผู้สมัครชั่วคราว (*v_repage_proc*)
- จำนวนต่ำสุดของการประมวลผลที่แอ็คทีฟ เมื่อการประมวลผลถูกหยุดทำงานชั่วคราว (*v_min_process*)
- จำนวนต่ำสุดของวินาทีที่ผ่านไปของกิจกรรมสำหรับการประมวลผลหลังจากเรียกทำงานอีกครั้ง (*v_exempt_secs*)

สำหรับตัวอย่างเกี่ยวกับค่าติดตั้งและการปรับพารามิเตอร์เหล่านี้ โปรดดู “การปรับการควบคุมโหลดหน่วยความจำ VMM ด้วยคำสั่ง schedo” ในหน้า 162

หนึ่งครั้งต่อวินาที ตัวกำหนดตารางเวลา (การประมวลผล 0) จะตรวจสอบค่าข้างต้นทั้งหมด ที่วัดค่าได้ ซึ่งจะสะสมผ่านช่วงเวลาหนึ่งวินาทีก่อนหน้านี้ และพิจารณาว่า การประมวลผลจะถูกหยุดทำงานหรือเรียกใช้งาน ถ้าการประมวลผลที่ต้องการหยุดทำงานชั่วคราว การประมวลผลที่เลือกทุกกระบวนการสำหรับการหยุดทำงานชั่วคราวโดยพารามิเตอร์การทดสอบ **–p** และ **–e** ที่ถูกทำเครื่องหมายหยุดทำงานชั่วคราว เมื่อการประมวลผลได้รับ CPU ในโหมดผู้ใช้ถัดไป ซึ่งจะถูกหยุดทำงานชั่วคราว (ยกเว้นว่า การทำเช่นนี้จะลดจำนวนของการประมวลผลที่แอ็คทีฟ ให้ต่ำกว่าค่า **–m**) เงื่อนไขของโหมดผู้ใช้ที่ถูกใช้ เพื่อให้การประมวลผลที่ไม่เหมาะสมหยุดทำงานชั่วคราวในระหว่างการทำกิจกรรมของระบบที่สำคัญ แทน ถ้าในช่วงเวลาหนึ่งวินาทีต่อมา เงื่อนไขการ thrash จะยังคงเป็นไปตามกฎเกณฑ์ ผู้สมัครของการประมวลผลเพิ่มเติมที่ตรงตามเงื่อนไข ที่ถูกตั้งค่าโดย **–p** และ **–e** จะถูกทำเครื่องหมายหยุดทำงานชั่วคราว เมื่อตัวกำหนดตารางเวลา พิจารณาว่า ปฏิบัติตามเกณฑ์ของช่วงเวลาที่ปลอดภัย และการประมวลผลถูกเรียกใช้งานอีกครั้ง จำนวนของการประมวลผลที่หยุดทำงานบางกระบวนการ จะอยู่บนคิวที่รันอยู่ (ทำให้แอ็คทีฟ) ทุกๆ วินาที

การประมวลผลที่หยุดทำงานชั่วคราวจะถูกเรียกใช้งานอีกครั้งโดย:

1. ระดับความสำคัญ

2. การเรียงลำดับที่การประมวลผลหยุดทำงานชั่วคราว

การประมวลผลที่หยุดทำงานชั่วคราวจะไม่เรียกใช้งานใหม่ทั้งหมดในหนึ่งครั้ง ค่าของจำนวนของการประมวลผล ที่เรียกใช้งานใหม่จะถูกเลือกด้วยสูตรที่จดจำจำนวนของการประมวณที่แอ็คทีฟ และเรียกใช้งานใหม่อีกครั้งสำหรับหนึ่งในห้าจำนวนของการประมวลผลที่แอ็คทีฟ หรือการเพิ่มการโยงที่ต่ำลง ซึ่งจะมีขนาดใหญ่ขึ้น ซึ่งควรระมัดระวังกลยุทธ์ที่ส่งผลทำให้มีการเพิ่มระดับของมัลติโปรแกรมมิง ประมาณ 20 เปอร์เซ็นต์ต่อวินาที เจตนาของกลยุทธ์นี้คือ การทำให้อัตราของการเรียกใช้งานช้าลงในระหว่างวินาทีแรก หลังจากที่ช่วงเวลาที่ปลอดภัยหมดอายุลง ขณะที่เพิ่มอัตราการแนะนำใหม่ ในวินาทีต่อมา ถ้าเงื่อนไขการ overcommit หน่วยความจำเกิดขึ้นในระหว่างคอร์สของการเรียกใช้งานการประมวลผล เหตุการณ์ต่อไปนี้จะเกิดขึ้น:

- การเรียกใช้งานอีกครั้งหยุดทำงาน
- การประมวลผลที่ต้องการเรียกใช้งานอีกครั้งจะถูกทำเครื่องหมายหยุดทำงานชั่วคราวอีกครั้ง
- การประมวลผลเพิ่มเติมจะถูกหยุดทำงานชั่วคราวตามกฎข้างต้น

การจัดสรรและการเรียกคืนสล็อตพื้นที่การเพจ

ระบบปฏิบัติการจะสนับสนุนเมธอดการจัดสรรทั้งหมดสามเมธอด สำหรับหน่วยเก็บใช้งาน

เมธอดการจัดสรรทั้งสามเมธอดสำหรับหน่วยเก็บใช้งานยังอ้างถึง *สล็อตพื้นที่การเพจ* ดังต่อไปนี้:

- การจัดสรรถัดมา
- การจัดสรรก่อนหน้า
- การจัดสรรที่รอ

หมายเหตุ: สล็อตพื้นที่การเพจจะถูกปล่อยโดยการยกเลิกการประมวลผล (ซึ่งไม่ใช่เธรด) หรือโดยการเรียกของระบบ **disclaim( )** สล็อตจะไม่ถูกปล่อยโดยการเรียกของระบบ **free( )**

อัลกอริธึมการจัดสรรล่าสุด

โปรแกรมจำนวนมากใช้ประโยชน์จากการจัดสรรล่าสุด ด้วยการจัดสรรช่วงของแอดเดรสหน่วยความจำเสมือนสำหรับโครงสร้างที่มีขนาดสูงสุด จากนั้น ใช้โครงสร้างมากเท่าที่จะมากได้สำหรับสถานการณ์ที่ต้องการ เพจของช่วงของแอดเดรสหน่วยความจำเสมือน ซึ่งไม่เคยเข้าถึง ไม่เคยต้องการกรอบหน่วยความจำที่ใช้จริง หรือสล็อตพื้นที่การเพจ

เทคนิคนี้จะเกี่ยวข้องกับระดับของความเสี่ยง บางระดับ ถ้าโปรแกรมทั้งหมดที่รันอยู่ในเครื่องที่เกิดสถานการณ์ ซึ่งมีขนาดสูงสุดอย่างพร้อมเพียงกัน พื้นที่การเพจอาจไม่เหลืออยู่ บางโปรแกรมอาจไม่สามารถดำเนินการได้จนเสร็จสิ้น

อัลกอริธึมก่อนหน้า

เมธอด การจัดสรรสล็อตพื้นที่การเพจของระบบปฏิบัติการอันดับที่สองจะใช้ในการติดตั้ง ซึ่งสถานการณ์นี้เหมือนกับหรือมีค่าใช้จ่ายของความล้มเหลว ที่ต้องแก้ไขให้เสร็จสิ้นที่สูงกว่า สำหรับการจัดสรรก่อนหน้าที่จะเรียกที่เหมาะสม อัลกอริธึมนี้เป็นสาเหตุทำให้จำนวนของสล็อตพื้นที่การเพจที่เหมาะสมถูกจัดสรร ในเวลาที่ช่วงของแอดเดรสหน่วยความจำเสมือนถูกจัดสรร ตัวอย่างเช่น รูทีนย่อย **malloc( )** ถ้ามีสล็อตพื้นที่การเพจที่ไม่เพียงพอในการสนับสนุนรูทีนย่อย **malloc( )** โค้ดระบุความผิดพลาดจะถูกตั้งค่า อัลกอริธึมการจัดสรรก่อนหน้าจะถูกเรียกใช้งาน ดังต่อไปนี้:

```
# export PSALLOC=early
```

ตัวอย่างนี้ เป็นสาเหตุของโปรแกรมในอนาคตทั้งหมดที่ต้องประมวลผลในสถาวะแวดล้อมของการใช้การจัดสรรก่อนหน้า shell ที่ประมวลผลอยู่ในปัจจุบันไม่ได้รับผลกระทบ

การจัดสรรก่อนหน้านี้ของ จุดสนใจในเรื่องของการวิเคราะห์ผลการทำงาน เนื่องจากกขนาดของพื้นที่การเพจ ที่เกี่ยวข้อง ถ้าการจัดสรรก่อนหน้านี้ถูกเปิดใช้งานสำหรับโปรแกรมเหล่านี้ ข้อกำหนดพื้นที่การเพจสามารถเพิ่มขึ้นหลายๆ ครั้งได้ ขณะที่ข้อแนะนำปกติ สำหรับขนาดของพื้นที่การเพจอย่างน้อยต้องมีเป็นสองเท่าของหน่วยความจำที่ใช้จริงของระบบ ขอแนะนำสำหรับระบบที่ใช้ **PSALLOC=early** จะมีขนาดอย่างน้อยเป็นสี่เท่าของขนาดหน่วยความจำจริง ตามความเป็นจริง นี่เป็นเพียงจุดเริ่มต้น การวิเคราะห์ข้อกำหนดหน่วยเก็บเสมือนของเวิร์กโหลดของคุณ และจัดสรรพื้นที่การเพจเพื่อทำให้เกิดความเหมาะสม ดังตัวอย่างในหนึ่งครั้ง เซิร์ฟเวอร์ AIXwindows ต้องการ 250 MB ของพื้นที่การเพจเมื่อรันด้วยการจัดสรรก่อนหน้า

ขณะที่ใช้ **PSALLOC=early** ผู้ใช้ควรตั้งค่า handler สำหรับสัญญาณ SIGSEGV โดยการจัดสรรล่วงหน้าและตั้งค่าหน่วยความจำเป็นสแต็กโดยใช้ฟังก์ชัน **sigaltstack** แม้ว่าจะระบุ **PSALLOC=early** ไว้ เมื่อมีพื้นที่การเพจที่ไม่เพียงพอ และโปรแกรมพยายามขยายสแต็ก โปรแกรมอาจได้รับสัญญาณ SIGSEGV

อัลกอริธึมการจัดสรรที่รอ

เมธอด paging-space-slot-allocation ของระบบปฏิบัติการที่สาม คือลักษณะการทำงานดีฟอลต์ นโยบาย Deferred Page Space Allocation (DPSA) จะหน่วงเวลาการจัดสรรพื้นที่การเพจจนกว่าจำเป็นต้อง page out หน้า ซึ่งส่งผลให้ไม่มีการจัดสรรพื้นที่การเพจซึ่งสูญเปล่า เมธอดนี้ช่วยประหยัดพื้นที่การเพจซึ่งคือพื้นที่ดิสก์ได้เป็นจำนวนมาก

สำหรับระบบบางระบบ พื้นที่การเพจอาจไม่มีความต้องการ แม้ว่า เพจทั้งหมดที่เข้าถึงจะได้รับการติดต่อ สถานการณ์นี้เป็นสถานการณ์ทั่วไปบนระบบ พร้อมกับจำนวนของ RAM ที่มีขนาดใหญ่ อย่างไรก็ตาม อาจส่งผลทำให้เกิด overcommitment ของพื้นที่การเพจในกรณีที่หน่วยความจำเสมือนจะถูกใช้มากกว่า RAM ที่เข้าถึง

หากต้องการปิดใช้งาน DPSA และสงวนนโยบายการจัดสรรพื้นที่เพจถัดมา ให้รันคำสั่ง :

```
# vmo -o defps=0
```

หากต้องการเรียกใช้งาน DPSA ให้รันคำสั่ง:

```
# vmo -o defps=1
```

โดยทั่วไป ผลการทำงานของระบบสามารถปรับปรุงได้โดย DPSA เนื่องจากการใช้งานของพื้นที่การเพจที่จัดสรร หลังจากข้อบกพร่องของเพจถูกหลีกเลี่ยง อุปกรณ์พื้นที่การเพจต้องการพื้นที่ดิสก์เพียงเล็กน้อย หากใช้ DPSA

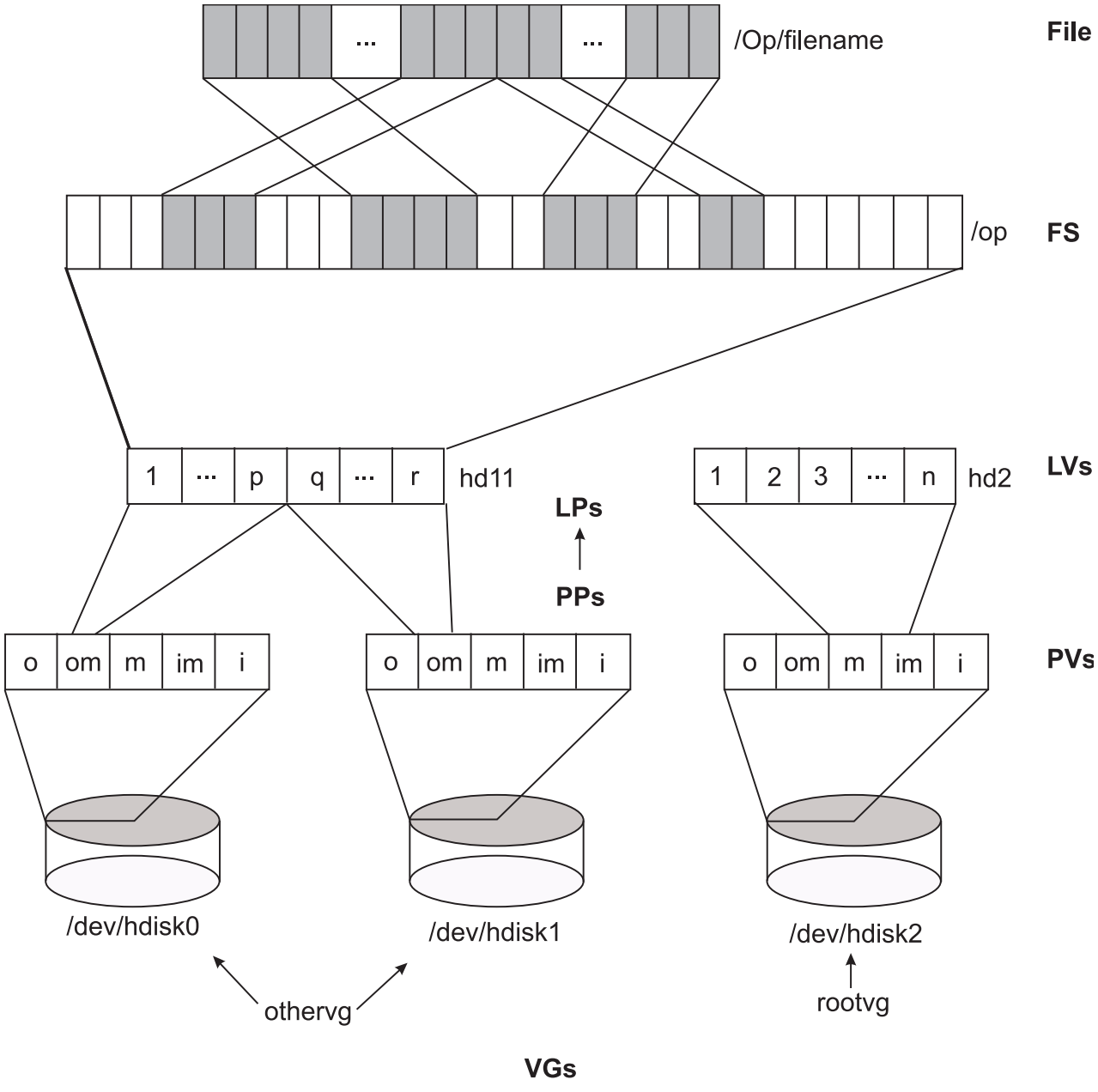
สำหรับข้อมูลเพิ่มเติม โปรดดู “การจัดสรรพื้นที่ว่างหน้า” ในหน้า 170 และ “การวางตำแหน่งของพื้นที่ว่างการเพจและขนาด” ในหน้า 107

ประสิทธิภาพการจัดการหน่วยเก็บดิสก์ถาวร

ระบบปฏิบัติการใช้ลำดับชั้นของโครงสร้างเพื่อจัดการ หน่วยเก็บดิสก์ถาวร

ดิสก์ไดร์ฟแต่ละรายการที่เรียกว่าฟิสิคัลวอลุ่ม (PV) มีชื่อ เช่น /dev/hdisk0 ถ้าฟิสิคัลวอลุ่ม ถูกใช้งานอยู่ ฟิสิคัลวอลุ่มจะเป็นสมาชิกของกลุ่มวอลุ่ม (VG) ฟิสิคัลวอลุ่มทั้งหมด ในกลุ่มวอลุ่มถูกแบ่งออกเป็นฟิสิคัลพาร์ติชัน (PPs) ขนาดเท่ากัน (โดยค่าดีฟอลต์ 4 MB ในกลุ่มวอลุ่มที่มีฟิสิคัลวอลุ่มขนาดเล็กกว่า 4 GB; 8 MB หรือมากกว่าในกรณีที่มีดิสก์ที่ใหญ่ขึ้น)

เพื่อการจัดสรรพื้นที่ว่าง แต่ละฟิสิคัลวอลุ่มถูกแบ่งออกเป็น ห้าพื้นที่ โปรดดู “ตำแหน่งบนฟิสิคัลวอลุ่ม” ในหน้า 221 สำหรับข้อมูลเพิ่มเติม จำนวนของฟิสิคัลพาร์ติชันในแต่ละพื้นที่แตกต่างกันไปขึ้นอยู่กับความจุทั้งหมดของดิสก์ไดร์ฟ



รูปที่ 10. การจัดระเบียบของข้อมูลดิสก์ถาวร (ไม่มิเรอร์). ภาพสาธิตแสดงลำดับชั้นของฟิสิคัลวอลุ่มที่มีการแบ่งพาร์ติชัน เป็นโลจิคัลวอลุ่มหนึ่งรายการขึ้นไป พาร์ติชันหรือโลจิคัลวอลุ่ม เหล่านี้มีระบบไฟล์ที่มีโครงสร้างไดเร็กทอรี ซึ่งประกอบด้วยไฟล์แต่ละไฟล์ ไฟล์ถูกบันทึกลงในบล็อกที่มีอยู่ใน แทร็คบนสื่อหน่วยเก็บ และโดยปกติ บล็อกเหล่านี้ไม่ติดกัน การแบ่งเฟรกเมนต์ดิสก์เกิดขึ้นเมื่อข้อมูลถูกลบออกและมีการบันทึกไฟล์ข้อมูลใหม่ ลงในบล็อกที่ว่าง ซึ่งกระจัดกระจายอยู่ทั่วไปรอบหลายแทร็ค บนสื่อบันทึก

ภายในแต่ละกลุ่มวอลุ่ม มีการกำหนดโลจิคัลวอลุ่ม (LVs) หนึ่งรายการ ขึ้นไป แต่ละโลจิคัลวอลุ่มประกอบด้วยโลจิคัลพาร์ติชันหนึ่งรายการขึ้นไป แต่ละโลจิคัลพาร์ติชันตรงกับฟิสิคัลพาร์ติชันอย่างน้อยหนึ่งรายการ ถ้ามีการระบุการทำมิเรอร์สำหรับโลจิคัลวอลุ่ม จะมีการจัดสรรฟิสิคัลพาร์ติชันเพิ่มเติม เพื่อจัดเก็บสำเนาเพิ่มเติมของแต่ละโลจิคัล พาร์ติชัน แม้ว่าโลจิคัลพาร์ติชันมีการกำหนดหมายเลขต่อเนื่องกัน แต่ฟิสิคัลพาร์ติชันที่เกี่ยวข้องไม่จำเป็นต้องต่อเนื่องหรือ ติดกัน

โลจิคัลวอลุ่มสามารถใช้ทำงานของระบบได้หลายอย่าง เช่น การเพจ แต่แต่ละโลจิคัลวอลุ่มที่มีข้อมูลระบบปกติหรือข้อมูลผู้ใช้หรือ โปรแกรมมี journaled file system (JFS หรือ Enhanced JFS) เดียว แต่ละ JFS ประกอบด้วยพูลของบล็อกขนาดหน้า

(4096 ไบต์) เมื่อข้อมูลถูกบันทึกลงในไฟล์ จะมีการจัดสรรบล็อกเพิ่มเติมหนึ่งบล็อกขึ้นไป ให้กับไฟล์นั้น บล็อกเหล่านี้อาจหรืออาจไม่ติด กันและกับบล็อกอื่นซึ่งจัดสรรก่อนหน้านี้ให้กับ ไฟล์

เพื่อการสาธิต รูปภาพก่อนหน้านี้แสดงสถานการณ์ที่ไม่ดี (แต่ไม่ใช่สถานการณ์ เลวร้ายที่สุดที่อาจเป็นไปได้) ที่อาจเกิดขึ้นในระบบไฟล์ซึ่งใช้อยู่เป็นเวลานานโดยไม่มีการจัดระเบียบใหม่ ไฟล์ /op/filename มีการบันทึก แบบฟิสิคัลบนบล็อกจำนวนมากที่อยู่ห่าง กัน การอ่านไฟล์ตามลำดับอาจส่งผลให้เกิดการดำเนินงานค้นหา ที่ใช้เวลานานหลายครั้ง

ในขณะที่ ตามหลักการแล้ว ไฟล์ของระบบปฏิบัติการเป็นสตริงของไบต์ที่เป็นลำดับ และติดกัน แต่ความเป็นจริงอาจแตกต่างไปมาก การแบ่งเฟรกเมนต์อาจเกิดขึ้นตั้งแต่หลายส่วนขยายไปจนถึงโลจิคัลวอลุ่ม ตลอดจนกิจกรรมการจัดสรร/รีลีส/การจัดสรรใหม่ภายในระบบ ไฟล์ ระบบไฟล์มีการแบ่งเฟรกเมนต์เมื่อพื้นที่ว่างที่มีอยู่ของระบบนั้นประกอบด้วย ชิ้นส่วนเล็กๆ ของพื้นที่ว่างจำนวนมาก ทำให้ไม่สามารถบันทึกไฟล์ใหม่ใน บล็อกที่ติดกันได้

การเข้าถึงไฟล์ในระบบไฟล์ที่มีการแบ่งเฟรกเมนต์สูงอาจส่งผลให้เกิดการค้นหา จำนวนมากและเวลาการตอบกลับ I/O ที่นานขึ้น (เวลาแฝงการค้นหา มีผลกระทบต่อเวลาการตอบกับ I/O) ตัวอย่างเช่น ถ้าเข้าถึงไฟล์ตามลำดับ การวางไฟล์ที่ประกอบด้วยชิ้นส่วนจำนวนมากที่กระจัดกระจายเป็นวงกว้าง ต้องใช้การค้นหามากกว่าการวางไฟล์ที่ประกอบด้วยชิ้นส่วนขนาดใหญ่ที่ติดกันจำนวน สองสามชิ้น ถ้าเข้าถึงไฟล์แบบสุ่ม การวางที่กระจัดกระจาย เป็นวงกว้างต้องใช้การค้นหานานกว่าการวางซึ่งบล็อกของไฟล์ อยู่ใกล้กัน

ผลกระทบจากการวางของไฟล์ต่อประสิทธิภาพ I/O จะหายไป เมื่อไฟล์ถูกบัฟเฟอร์ในหน่วยความจำ เมื่อเปิดไฟล์ในระบบปฏิบัติการ ไฟล์จะถูกแม็พเข้าถึงเซกเมนต์ข้อมูลถาวรในหน่วยความจำ เสมือน เซกเมนต์แสดงถึงบัฟเฟอร์เสมือนของไฟล์ บล็อกของไฟล์แม็พเข้ากับ หน้าเซกเมนต์โดยตรง VMM จัดการหน้าเซกเมนต์ การอ่านบล็อกของไฟล์เข้าในหน้าเซกเมนต์เมื่อต้องการ (เมื่อมีการเข้าถึง หน้าเซกเมนต์) มีหลายสถานการณ์ที่ทำให้ VMM บันทึกหน้ากลับไปยังบล็อกที่ตรงกันในไฟล์บนดิสก์ แต่โดยทั่วไปแล้ว VMM จะเก็บหน้าไว้ในหน่วยความจำถ้ามีการเข้าถึงหน้า เมื่อเร็วๆ นี้ ดังนั้น หน้าที่เข้าถึงบ่อยครั้งจึงมีแนวโน้มที่จะอยู่ในหน่วยความจำนานกว่า และการเข้าถึงโลจิคัลไฟล์ไปยังบล็อกที่ตรงกันสามารถดำเนินการได้โดยไม่ต้องเข้าถึงฟิสิคัลดิสก์

ในบางเวลา ผู้ใช้หรือผู้ดูแลระบบสามารถเลือกที่จะจัดระเบียบการวางไฟล์ ภายในโลจิคัลวอลุ่ม และการวางโลจิคัลวอลุ่มภายใน ฟิสิคัลวอลุ่มใหม่ เพื่อลดการแบ่งเฟรกเมนต์และกระจาย I/O โหลดทั้งหมด อย่างเท่าเทียมกันมากขึ้น “ผลการทำงานของโลจิคัลวอลุ่มและดิสก์ I/O” ในหน้า 195 มีรายละเอียดเพิ่มเติม เกี่ยวกับการตรวจพบและการแก้ไขปัญหาการวางดิสก์และการแบ่งเฟรกเมนต์

การสนับสนุนสำหรับหน่วยความจำ pinned

AIX ช่วยให้สามารถเก็บรักษา หน้าหน่วยความจำไว้ในหน่วยความจำจริงได้ตลอดเวลา กลไกนี้เรียกว่า pinning หน่วยความจำ

Pinning พื้นที่หน่วยความจำห้ามไม่ให้เพจเจอร์ขโมยหน้าจากหน้า ที่กลับไปยังพื้นที่หน่วยความจำที่ pinned พื้นที่หน่วยความจำที่กำหนดไว้ในพื้นที่ว่างระบบ หรือพื้นที่ว่างผู้ใช้ อาจมีการ pinned ได้ หลังจาก pinned พื้นที่หน่วยความจำแล้ว การเข้าถึงพื้นที่นั้น จะไม่ทำให้เกิด page fault จนกว่าพื้นที่จะถูก unpinned ในเวลา ต่อมา ในขณะที่ส่วนของเคอร์นัลยังคง pinned อยู่ พื้นที่จำนวนมากสามารถเพจได้ และมีการ pinned เฉพาะในขณะที่เข้าถึงเท่านั้น

ข้อดีของการมีส่วนของหน่วยความจำที่ pinned คือ เมื่อเข้าถึงหน้าที่ pinned คุณสามารถดึงข้อมูลหน้าได้โดยไม่ต้องดำเนินการตามขั้นตอนวิธีการเปลี่ยน หน้า ผลข้างเคียงเชิงลบของการมีหน้าหน่วยความจำที่ pinned มากเกินไปคือ อาจเพิ่มกิจกรรมการเพจสำหรับหน้าที่ unpinned ซึ่งจะทำให้ ประสิทธิภาพด้อยลง

สามารถใช้ **vmo maxpin%** ที่ปรับได้เพื่อปรับจำนวนของหน่วยความจำ ที่สามารถ pinned ได้ **maxpin%** ที่ปรับได้ระบุเปอร์เซ็นต์สูงสุด ของหน่วยความจำจริงที่สามารถ pinned ได้

หมายเหตุ: เนื่องจากเคอร์นัลต้องสามารถ pin ข้อมูลเคอร์นัลจำนวนหนึ่งได้ ดังนั้นการลด ค่าของ **maxpin%** ที่ปรับได้อาจนำไปสู่ปัญหาเกี่ยวกับฟังก์ชัน และไม่แนะนำให้ทำเช่นนั้น

แอ็พพลิเคชันผู้ใช้อาจจะ pin หน่วยความจำผ่านทางกลไกที่แตกต่างกันหลายอย่าง แอ็พพลิเคชันสามารถใช้รูทีนย่อย **plock ( )**, **mlock( )**, และ **mlockall( )** เพื่อ pin หน่วยความจำของแอ็พพลิเคชัน

แอ็พพลิเคชันสามารถ pin พื้นที่หน่วยความจำแบบแบ่งใช้เองโดยการระบุอ็อพชัน **SHM_LOCK** ใน รูทีนย่อย **shmctl( )** แอ็พพลิเคชันยังสามารถ pin พื้นที่หน่วยความจำแบบแบ่งใช้โดยการระบุแฟล็ก **SHM_PIN** ใน **shmget( )** ได้ด้วย

มัลติโพรเซสซิง

ในเวลาที่กำหนดใดๆ ข้อจำกัดด้านเทคโนโลยียังคงเกี่ยวข้องกับความเร็ว ที่ชิปตัวประมวลผลเดี่ยวสามารถปฏิบัติงานได้ ถ้าเวิร์กโหลดของระบบ ไม่สามารถจัดการตามความต้องการได้ด้วยตัวประมวลผลเดี่ยว การตอบกลับตัวประมวลผลจำนวนมากกับปัญหานี้

ความสำเร็จของการตอบกลับนี้ไม่ได้ขึ้นอยู่กับความชำนาญของผู้ออกแบบระบบเพียงอย่างเดียว แต่ยังขึ้นอยู่กับเวิร์กโหลดที่สามารถแก้ไขในมัลติโพรเซสซิงได้ ในส่วนของภารกิจที่บุคคลต้องเป็นผู้กระทำ การเพิ่มบุคคลอาจเป็นความคิดที่ดี หากภารกิจคือการตอบปัญหาไปยังหมายเลขโทรฟรี แต่อาจมีความน่าสงสัย หากภารกิจนั้นคือการขับรถ

ถ้าผลการทำงานที่ปรับปรุงแล้วคือวัตถุประสงค์ของการโอนย้ายระบบที่นำเสนอจากระบบยูนิโพรเซสเซอร์ ไปเป็นระบบมัลติโพรเซสเซอร์ เงื่อนไขต่อไปนี้ต้องเป็นจริง :

- เวิร์กโหลดคือตัวประมวลผลที่จำกัดและเต็มไปด้วยระบบยูนิโพรเซสเซอร์
- เวิร์กโหลดมีองค์ประกอบที่ใช้ตัวประมวลผล เช่น การทำรายการ หรือการคำนวณที่ซับซ้อน ซึ่งสามารถดำเนินการอย่างพร้อมเพียงกันและเป็นอิสระ
- ยูนิโพรเซสเซอร์ที่มีอยู่ไม่สามารถอัพเกรดหรือแทนที่ด้วยยูนิโพรเซสเซอร์อื่นๆ ของกำลังที่เพียงพอ

แม้ว่าแอ็พพลิเคชันเธรดเดี่ยวที่ไม่เปลี่ยนแปลงจะทำงานได้อย่างถูกต้องในสภาวะแวดล้อมแบบมัลติโพรเซสเซอร์ ผลการทำงานจะเปลี่ยนแปลงไปด้วยวิธีที่คาดไม่ถึง การโอนย้ายระบบไปยังมัลติโพรเซสเซอร์สามารถปรับปรุงทรูพุตของระบบ และสามารถปรับปรุงเวลาในการทำงานของความซับซ้อน แอ็พพลิเคชันแบบมัลติเธรด แต่จะปรับปรุงเวลาตอบกลับของแต่ละระบบ คำสั่งเธรดแบบเธรดเดี่ยว

การขอรับผลการทำงานที่เป็นไปได้จากระบบมัลติโพรเซสเซอร์ ที่ต้องการความเข้าใจระบบปฏิบัติการและการประมวลผลฮาร์ดแวร์แบบไดนามิก ที่ไม่ซ้ำกับสภาวะแวดล้อมแบบมัลติโพรเซสเซอร์

หลักการและสถาปัตยกรรม Symmetrical Multiprocessor

ตามการเปลี่ยนแปลงที่เพิ่มความซับซ้อนของระบบ การใช้หลายตัวประมวลผลทำให้เกิดข้อควรพิจารณาเกี่ยวกับการออกแบบที่ต้อง คำนึงถึงเพื่อให้เกิดการดำเนินงานและประสิทธิภาพที่พึงพอใจ

ความซับซ้อนมากขึ้นส่งผลให้มีการแลกเปลี่ยนระหว่างฮาร์ดแวร์/ซอฟต์แวร์ในขอบเขตที่กว้างขึ้น และต้องการการประสานงานออกแบบฮาร์ดแวร์/ซอฟต์แวร์ที่ ใกล้ชิดกันมากกว่าในระบบตัวประมวลผลเดียว ชุดที่แตกต่างกันของการตอบสนองและการแลกเปลี่ยนในการ ออกแบบเพิ่มความหลากหลายของสถาปัตยกรรมระบบหลายตัวประมวลผล

ชนิดของ multiprocessing

ระบบ multiprocessing (MP) มีอยู่หลายชนิด

MP ที่ไม่แบ่งใช้สิ่งใดเลย:

ตัวประมวลผลไม่แบ่งใช้สิ่งใดเลย (แต่ละตัวมีหน่วยความจำ แคช และดิสก์ของตัวเอง) แต่มีการเชื่อมต่อระหว่างกัน ระบบหลายตัวประมวลผลชนิดนี้ยังเรียกอีก อย่างว่า *pure cluster*

ตัวประมวลผลแต่ละตัวเป็นเครื่องแบบสแตนด์อะโลนโดยสมบูรณ์ และรันสำเนาของระบบปฏิบัติการ เมื่อเชื่อมต่อกับ LAN ตัวประมวลผลมีการ coupled หลวมๆ เมื่อ เชื่อมต่อโดยใช้สวิตช์ ตัวประมวลผลมีการ coupled แน่น การสื่อสารระหว่าง ตัวประมวลผลทำผ่านทางการส่งผ่านข้อความ

ข้อดีของระบบดังกล่าวคือการปรับสเกลได้ดีมากและการมีอยู่สูง ข้อเสียของระบบดังกล่าวคือโมเดลการเขียนโปรแกรมที่ไม่คุ้นเคย (การส่งผ่าน ข้อความ)

ดิสก์ MP แบบแบ่งใช้:

ข้อดีของดิสก์แบบแบ่งใช้คือมีการเก็บรักษาส่วนประกอบของโมเดล การเขียนโปรแกรมที่คล้ายกัน (ข้อมูลดิสก์สามารถเข้าถึงได้และต่อเนื่องกัน แต่ หน่วยความจำไม่เป็นเช่นนั้น) และสามารถทำให้มีอยู่สูงได้ง่ายกว่าระบบหน่วยความจำแบบแบ่งใช้เป็นอย่างมาก ข้อเสีย คือการปรับสเกลได้ที่จำกัดเนื่องจากปัญหาคอขวดในการเข้าถึงข้อมูลแบบแบ่งใช้ ในแบบฟิสิคัลและโลจิคัล

ตัวประมวลผลมีหน่วยความจำและแคชของตนเอง ตัวประมวลผลรันในแบบขนาน และแบ่งใช้ดิสก์ ตัวประมวลผลแต่ละตัวรันสำเนาของระบบปฏิบัติการ และตัวประมวลผล มีการ coupled หลวมๆ (เชื่อมต่อผ่านทาง LAN) การสื่อสารระหว่าง ตัวประมวลผลทำผ่านทางการส่งผ่านข้อความ

คลัสเตอร์หน่วยความจำแบบแบ่งใช้:

ตัวประมวลผลทั้งหมดในคลัสเตอร์หน่วยความจำแบบแบ่งใช้มีรีซอร์ส ของตนเอง (หน่วยความจำหลัก ดิสก์, I/O) และแต่ละตัวประมวลผลรันสำเนาของ ระบบปฏิบัติการ

ตัวประมวลผลมีการ coupled แน่น (เชื่อมต่อผ่านทางสวิตช์) การสื่อสารระหว่างตัวประมวลผลทำผ่านทางหน่วยความจำแบบแบ่งใช้

MP หน่วยความจำแบบแบ่งใช้:

ตัวประมวลผลทั้งหมดมีการ coupled แน่นอยู่ภายในกล่องเดียวกันกับ บัสความเร็วสูงหรือสวิตช์ ตัวประมวลผลแบ่งใช้หน่วยความจำสากล ดิสก์ และอุปกรณ์ I/O เดียวกัน มีระบบปฏิบัติการเพียงสำเนาเดียวเท่านั้นรันบนตัวประมวลผล ทั้งหมด และระบบปฏิบัติการต้องมีการกำหนดเพื่อใช้ประโยชน์ สถาปัตยกรรมนี้ (ระบบปฏิบัติการแบบมัลติเธรด)

SMPs มีข้อดีหลายอย่างดังนี้:

- เป็นวิธีที่ประหยัดสำหรับการเพิ่มผลผลิต

- นำเสนอภาพของระบบเดียวเนื่องจากระบบปฏิบัติการมีการแบ่งใช้ ระหว่างตัวประมวลผลทั้งหมด (จัดการได้ง่าย)
- ใช้หลายตัวประมวลผลในปัญหาเดียว (การเขียนโปรแกรมแบบขนาน)
- การปรับสมดุลโหลดดำเนินการโดยระบบปฏิบัติการ
- สามารถใช้โมเดลการเขียนโปรแกรม uniprocessor (UP) ใน SMP ได้
- ปรับสเกลได้สำหรับข้อมูลแบบแบ่งใช้
- ข้อมูลทั้งหมดสามารถเข้าถึงได้โดยตัวประมวลผลทุกตัว และมีการรักษาความต่อ เนื่องโดยตรรกะ snooping ของฮาร์ดแวร์
- ไม่จำเป็นต้องใช้ไลบรารีการส่งผ่านข้อความเพื่อสื่อสารระหว่าง ตัวประมวลผล เนื่องจากการสื่อสารดำเนินการผ่านทางหน่วยความจำแบบแบ่งใช้สากล
- ความต้องการกำลังเพิ่มเติมสามารถแก้ไขได้โดยการเพิ่มตัวประมวลผล เพิ่มเติมในระบบ อย่างไรก็ตาม คุณต้องคาดหวังในสิ่งที่เป็นไปได้จริงเกี่ยวกับประสิทธิภาพ ที่เพิ่มขึ้นเมื่อเพิ่มตัวประมวลผลเพิ่มเติมในระบบ SMP
- ในปัจจุบัน มีแอ็พพลิเคชันและเครื่องมือมากขึ้นมาก แอ็พพลิเคชัน UP ส่วนใหญ่ สามารถรันบนหรือเชื่อมต่อกับสถาปัตยกรรม SMP

ระบบ SMP มีข้อจำกัดบางอย่างดังนี้:

- มีข้อจำกัดในการปรับสเกลได้เนื่องจากการเชื่อมกันของแคช กลไกการล็อก อ็อบเจ็กต์แบบแบ่งใช้ และอื่นๆ
- ต้องการทักษะใหม่เพื่อใช้ประโยชน์หลายตัวประมวลผล เช่น การเขียนโปรแกรม เธรดและการเขียนโปรแกรมไดรเวอร์อุปกรณ์

การทำแอ็พพลิเคชันเป็นแบบขนาน

แอ็พพลิเคชันสามารถมีการทำเป็นแบบขนานบน SMP ด้วยวิธีอย่างใดอย่าง หนึ่งจากสองวิธี

- วิธีดั้งเดิมคือการแบ่งแอ็พพลิเคชันออกเป็นหลายกระบวนการ กระบวนการเหล่านี้สื่อสารโดยใช้การสื่อสารระหว่างกระบวนการ (IPC) เช่น ไปป์, semaphores หรือหน่วยความจำแบบแบ่งใช้ กระบวนการต้องสามารถบล็อกการรอเหตุการณ์ต่างๆ เช่น ข้อความจากกระบวนการอื่น และต้องประสานสิทธิเข้าถึงอ็อบเจ็กต์แบบแบ่งใช้ ด้วยบางสิ่ง เช่น ล็อก
- อีกวิธีหนึ่งคือการใช้อินเตอร์เฟสระบบปฏิบัติการที่ใช้ได้หลายระบบของเธรด UNIX (POSIX) เธรดมีปัญหา การประสานงานคล้ายกันเนื่องจากมีกระบวนการและกลไกการจัดการปัญหา คล้ายกัน ดังนั้น กระบวนการหนึ่งจึงอาจมีเธรดที่กำลังรันอยู่พร้อมกันบนตัวประมวล ที่แตกต่างกันได้เป็นจำนวนมาก การประสานงานเธรดและการจัดลำดับ การเข้าถึงข้อมูลแบบแบ่งใช้ถือเป็นความรับผิดชอบของผู้พัฒนา

ควรพิจารณาข้อดีของทั้งเธรดและกระบวนการเมื่อคุณกำลังเลือก วิธีการที่จะใช้สำหรับการทำแอ็พพลิเคชันเป็นแบบขนาน เธรดอาจเร็วกว่า กระบวนการและแบ่งใช้หน่วยความจำได้ง่ายกว่า ในทางกลับกัน การใช้กระบวนการ จะกระจายไปยังหลายเครื่องหรือคลัสเตอร์ได้ง่ายกว่า ถ้าแอ็พพลิเคชัน จำเป็นต้องสร้างหรือลบอินสแตนซ์ใหม่ เธรดจะเร็วกว่า (โอเวอร์เฮดมากกว่าในการ forking กระบวนการ) สำหรับฟังก์ชันอื่น โอเวอร์เฮดของเธรดประมาณเท่ากับโอเวอร์เฮด ของกระบวนการ

การทำข้อมูลให้เป็นอนุกรม

องค์ประกอบของหน่วยเก็บใดๆ ที่สามารถอ่านหรือเขียนด้วยเธรดตั้งแต่หนึ่งตัวขึ้นไป อาจเปลี่ยนแปลงได้ ขณะที่โปรแกรมกำลังรันอยู่

นี่คือความจริงของสภาวะแวดล้อมแบบมัลติโปรแกรมมิงเช่นเดียวกับสภาวะแวดล้อมแบบมัลติโพรเซสซิง แต่มัลติโพรเซสเซอร์จะเพิ่มขอบเขตและความสำคัญของการพิจารณา ในสองวิธีคือ:

- ส่วนสนับสนุนมัลติโพรเซสเซอร์และเธรดจะทำให้มีจุดดึงดูดและง่ายต่อการเขียนแอ็พพลิเคชัน ที่แบ่งใช้ข้อมูลระหว่างเธรด

- เคอร์เนลสามารถแก้ปัญหาเกี่ยวกับการทำให้เป็นอนุกรมโดยปิดใช้งาน อินเตอร์รัปต์

หมายเหตุ: หากต้องการหลีกเลี่ยงปัญหาที่ร้ายแรง โปรแกรมที่แบ่งใช้ข้อมูลต้องจัดเรียงการเข้าถึง ข้อมูลนั้นแบบอนุกรม แทนที่จะเป็นแบบขนาน ก่อนที่โปรแกรมจะอัพเดตหน่วยข้อมูลที่แบ่งใช้ ตรวจสอบให้แน่ใจว่า ไม่มีโปรแกรมอื่น (ซึ่งรวมถึงสำเนาอื่นๆ ของโปรแกรมรันอยู่บนเธรดอื่น) จะเปลี่ยนแปลงรายการ การอ่านสามารถทำได้เป็นปกติ แบบขนาน

กลไกหลักที่ถูกใช้เพื่อเก็บโปรแกรมออกจากการแทรกแซงกับโปรแกรมอื่น คือ *การล็อก* การล็อกคือสิ่งที่แสดงถึงสิทธิการใช้งาน เพื่อเข้าถึงหน่วยข้อมูลตั้งแต่หนึ่งหน่วยขึ้นไป คำร้องขอการล็อกและปลดล็อกจะเป็นคำร้องขอที่มีขนาดเล็กมาก นั่นคือ คำร้องขอจะถูกนำไปใช้ด้วยวิธีที่ไม่ใช่การอินเตอร์รัปต์หรือการเข้าถึงมัลติโพรเซสเซอร์ ที่ส่งผลถึงผลลัพธ์โปรแกรมทั้งหมดที่เข้าถึงหน่วยข้อมูลที่แบ่งใช้ ต้องขอรับการล็อกที่สอดคล้องกับหน่วยข้อมูลก่อนที่จะจัดการกับข้อมูลนั้น ถ้าการล็อกไม่ได้ถูกพักไว้โดยโปรแกรมอื่น (หรือเธรดอื่นกำลังรันโปรแกรมเดียวกันนี้อยู่) โปรแกรมที่ร้องขอต้องเลื่อนการเข้าถึงออกไป จนกว่าการล็อกจะกลับมาพร้อมใช้งาน

ในด้านการใช้เวลาที่รอสำหรับการล็อก การทำให้เป็นอนุกรมจะถูกเพิ่มให้กับ จำนวนเวลาที่เธรดไม่สามารถจัดส่งได้ ขณะที่เธรดไม่สามารถจัดส่งได้ เธรดอื่นจะเป็นสาเหตุทำให้ไม่สามารถจัดส่งบรรทัดแคชของเธรดที่ต้องการแทนที่ได้ ซึ่งส่งผลทำให้ต้นทุนของเวลาแฝงของหน่วยความจำเพิ่มขึ้น เมื่อเธรดได้รับล็อกและถูกจัดส่ง

เคอร์เนลของระบบปฏิบัติการมีหน่วยข้อมูลที่แบ่งใช้จำนวนมาก ซึ่งต้องดำเนินการทำให้เป็นอนุกรมภายใน หน่วงเวลาการทำให้เป็นอนุกรมสามารถเกิดขึ้นได้ในแอ็พพลิเคชันโปรแกรม ซึ่งไม่ได้แบ่งใช้ข้อมูลกับโปรแกรมอื่น เนื่องจากเคอร์เนลเซอร์วิสจะถูกใช้โดยโปรแกรมที่ทำให้เป็นอนุกรมเพื่อแบ่งใช้ข้อมูลเคอร์เนล

ล็อก

ใช้ล็อกเพื่อจัดสรรและล้างข้อมูลหน่วยความจำของระบบปฏิบัติการภายในให้ว่าง

สำหรับข้อมูลเพิ่มเติม โปรดดู การทำความเข้าใจเกี่ยวกับการล็อก

ชนิดของล็อก:

มีการใช้ระเบียบวิธีการล็อก Open Software Foundation/1 (OSF/1) 1.1 เป็นโมเดลสำหรับฟังก์ชันล็อก AIX หลายตัวประมวลผล

อย่างไรก็ตาม เนื่องจากระบบมีการให้สิทธิก่อนและเพจได้ จึงมีการเพิ่มลักษณะ บางอย่างลงในโมเดลการล็อก OSF/1 1.1 ล็อกแบบง่ายและล็อกซับซ้อน มีการให้สิทธิก่อน นอกจากนี้ เธรดอาจพักอยู่เมื่อพยายามจัดหาล็อกแบบง่ายที่ยุ่ง ถ้าเจ้าของของล็อกไม่ได้กำลังรันอยู่ในปัจจุบัน นอกจากนั้น ล็อกแบบง่าย กลายเป็นล็อกที่พักไว้เมื่อตัวประมวลผล spinning บนล็อกแบบง่าย ในช่วงเวลาหนึ่ง (ช่วงเวลานี้เป็นตัวแปรทั้ง ระบบ)

ล็อกแบบง่าย:

ล็อกแบบง่ายในระบบปฏิบัติการเวอร์ชัน 4 คือ spin lock ที่จะพัก ภายใต้เงื่อนไขบางอย่าง เพื่อป้องกันไม่ให้เธรดหมุนอย่างไม่สิ้นสุด

ล็อกแบบง่ายอาจถูกแย่งสิทธิได้ ซึ่งหมายความว่าเคอร์นัลเธรดอาจถูกแย่งสิทธิ โดยเคอร์นัลเธรดอื่นที่มีระดับความสำคัญสูงกว่าในขณะที่มีล็อกแบบง่าย บน ระบบหลายตัวประมวลผล ล็อกแบบง่ายซึ่งป้องกันส่วนสำคัญไม่ให้เธรดขัดจังหวะ ต้องมีการใช้พร้อมกับการควบคุมการขัดจังหวะเพื่อจัดลำดับการดำเนินการ ทั้งภายในกระบวนการที่กำลังดำเนินการและระหว่างตัวประมวลผลที่แตกต่างกัน

บนระบบตัวประมวลผลเดียว การควบคุมการขัดจังหวะเป็นสิ่งที่เพียงพอแล้ว ไม่จำเป็นต้องใช้ล็อก ล็อกแบบง่ายมีไว้เพื่อป้องกัน thread-thread และ การขัดจังหวะเธรดให้กับส่วนที่สำคัญ ล็อกแบบง่ายจะหมุนจนกว่าล็อก มีอยู่ถ้าอยู่ในตัวจัดการขัดจังหวะ ล็อกแบบง่ายมีสองสถานะคือ: ล็อก หรือปลดล็อก

การล็อกที่ซับซ้อน:

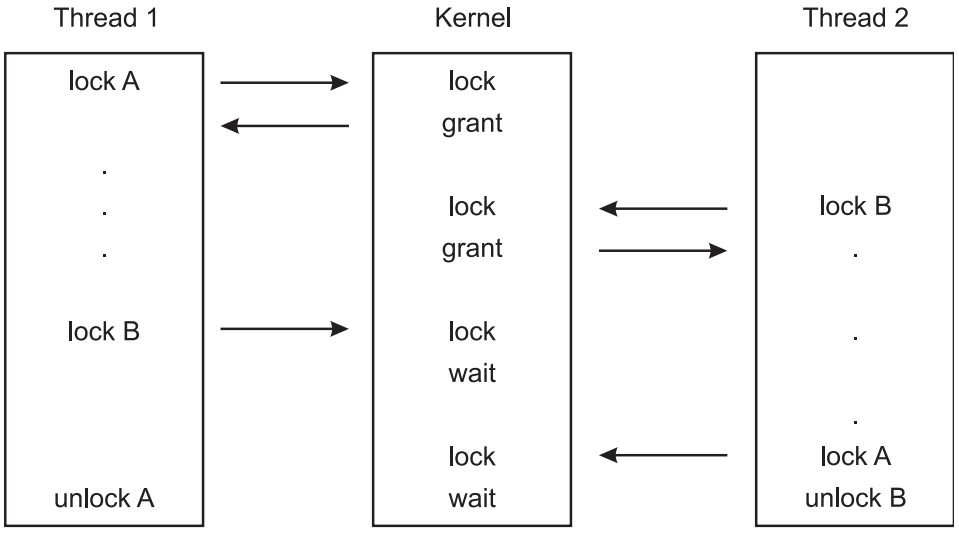
การล็อกที่ซับซ้อนใน AIX จะเป็นการล็อกแบบอ่าน-เขียน ซึ่งจะปกป้องส่วนที่สำคัญของเธรด-เธรด และเป็นการล็อกที่สามารถครอบครองได้ก่อน

การล็อกที่ซับซ้อนคือการล็อกแบบ spin ซึ่งจะ sleep อยู่ภายใต้เงื่อนไขบางอย่าง ตามค่าดีฟอลต์แล้ว การล็อกเหล่านี้จะไม่เป็นการเรียกซ้ำ แต่สามารถเรียกซ้ำได้ผ่านเคอร์เนลเซอร์วิส **lock_set_recursive( )** และจะมีสถานะอยู่สามสถานะคือ: การเขียนอย่างเดียว (exclusive-write) การอ่านที่แบ่งใช้ (shared-read) หรือปลดล็อก

การล็อก:

โปรแกรมเมอร์ที่ทำงานในสภาวะแวดล้อมแบบมัลติโพรเซสเซอร์ต้องตัดสินใจว่า จะมีจำนวนการล็อกที่แยกจากันซึ่งต้องสร้างขึ้นสำหรับข้อมูลที่แบ่งใช้ ถ้า มีการล็อกเดี่ยวที่ต้องการทำให้เป็นอนุกรมสำหรับชุดของหน่วยข้อมูลที่แบ่งใช้ทั้งหมด contention การล็อกที่นำมาเปรียบเทียบ การมีอยู่ของล็อกที่ใช้ ซึ่งวางข้อจำกัดด้านบนบนทรูพุตของระบบ

ถ้าหน่วยข้อมูลที่มีความแตกต่างมีการล็อกของตนเอง ความเป็นไปได้ของเธรดสองตัว ที่แย่งการล็อกจะมีค่าต่ำเมื่อนำมาเปรียบเทียบ การเรียกการล็อกและการปลดล็อกเพิ่มเติมในแต่ละครั้ง จะมีต้นทุนของเวลาของตัวประมวลผล อย่างไรก็ตาม การมีอยู่ของการล็อกจำนวนมาก จะทำให้เกิด deadlock deadlock คือสถานการณ์ที่แสดงในภาพประกอบต่อไปนี้ ซึ่ง Thread 1 จะเป็นเจ้าของ Lock A และกำลังรอ Lock B นั่นหมายความว่า Thread 2 จะเป็นเจ้าของ Lock B และกำลังรอ Lock A ไม่มีโปรแกรมที่เคยเข้าใกล้การเรียก **unlock( )** ซึ่งจะหยุด deadlock การป้องกันตามปกติสำหรับ deadlock คือ การสร้างโปรโตคอล ซึ่งโปรแกรมที่ใช้ ชุดของล็อกที่กำหนดไว้ต้องได้รับมาจากโปรแกรม ในลำดับเดียวกัน



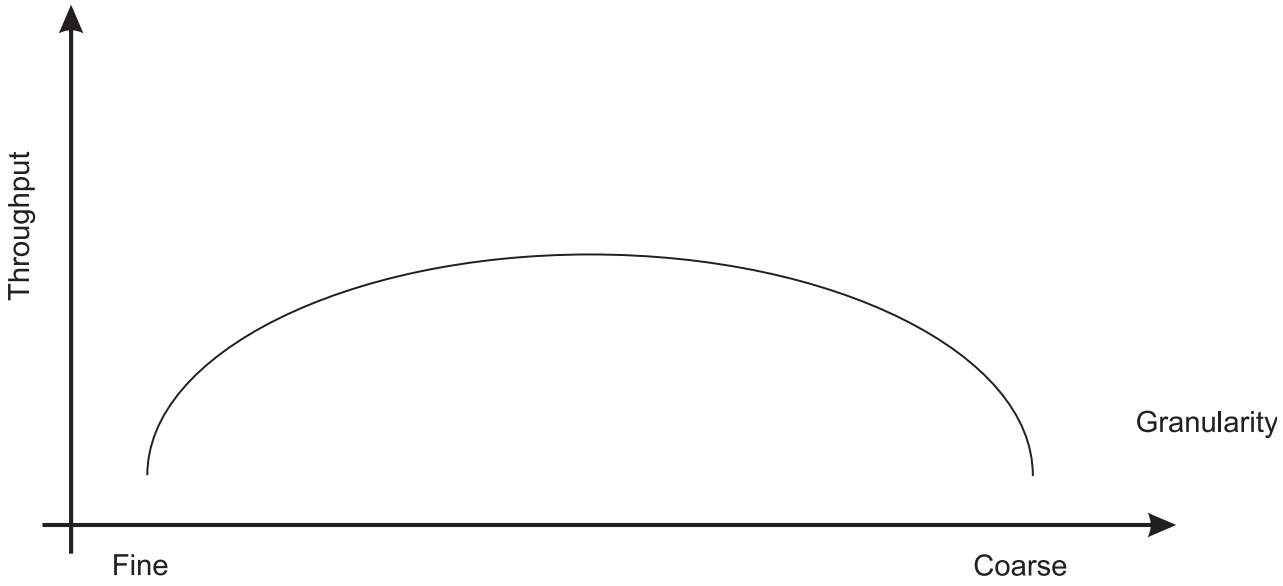
รูปที่ 11. Deadlock. แสดงอยู่ในภาพประกอบต่อไปนี้ คือ deadlock ที่มีคอลัมน์ที่ชื่อ Thread 1 เป็นเจ้าของ Lock A และรอ Lock B นั่นหมายความว่า คอลัมน์ที่ชื่อ Thread 2 เป็นเจ้าของ Lock B และกำลังรอ Lock A ไม่มีโปรแกรมเธรดที่เคยเข้าถึงการเรียกการปลดล็อก ที่จะหยุด deadlock

ตามที่ทฤษฎีของการเคียวรี่ รีซอร์สที่ไม่ได้ทำงาน จะมีค่าเฉลี่ยในการรอเพื่อขอรับรีซอร์สนั้น ความสัมพันธ์ที่ไม่ใช่แบบเชิงเส้น ถ้าล็อกไม่เป็นสองเท่า เวลารอโดยเฉลี่ยสำหรับการล็อกจะมีมากกว่าสองเท่า

วิธีที่ได้ผลดีที่สุดคือ การลดเวลารอการล็อก เพื่อลดขนาดของการล็อกที่ป้องกันไว้ นี่คือคำแนะนำบางข้อ:

- ลดความถี่ของล็อกที่ร้องขอ
- ล็อกเพียงแค่โค้ดที่เข้าถึงข้อมูลที่แบ่งใช้ ไม่ได้ล็อกโค้ดทั้งหมด ที่อยู่ในคอมโพเนนต์ (ซึ่งจะลดเวลาที่พักการล็อก)
- ล็อกเฉพาะหน่วยข้อมูลหรือโครงสร้าง และไม่ใช่รูทีนทั้งหมด
- เชื่อมโยงล็อกด้วยหน่วยข้อมูลเฉพาะ หรือโครงสร้างที่ไม่ใช่รูทีน
- สำหรับโครงสร้างที่มีขนาดใหญ่ ให้เลือกการล็อกหนึ่งแบบสำหรับแต่ละองค์ประกอบ ของโครงสร้างแทนการล็อกสำหรับโครงสร้างทั้งหมด
- ไม่ต้องทำ I/O ซิงโครไนซ์หรือการบล็อกกิจกรรมใดๆ ขณะที่พักการล็อก
- ถ้าคุณมีการเข้าถึงข้อมูลเดียวกันที่มากกว่าหนึ่งในคอมโพเนนต์ของคุณ ให้ลองย้ายคอมโพเนนต์นั้นเข้าด้วยกัน เพื่อให้สามารถครอบคลุมได้โดยล็อกและปลดล็อก เพียงครั้งเดียว
- หลีกเลี่ยงการทำให้ตื่นตัวเป็นสองเท่า ถ้าคุณแก้ไขข้อมูลบางส่วนภายใต้การล็อก และได้แจ้งให้บุคคลอื่นทราบว่า คุณได้ทำแก้ไขข้อมูลนั้น ให้ปลดล็อก ก่อนที่คุณจะติดประกอบการตื่นตัว
- ถ้าคุณต้องพักการล็อกสองกระบวนการพร้อมกัน ให้ร้องขอ busiest ลำดับสุดท้าย

ในกรณีอื่น กระบวนการขนาดเล็กจะเพิ่มความถี่ของการล็อกคำร้องขอ และปลดล็อก ซึ่งจะเพิ่มคำสั่งเพิ่มเติม คุณต้องหาที่ตั้งที่สมดุลระหว่าง too-fine และ too-coarse ของกระบวนการขนาดเล็ก กระบวนการขนาดเล็กที่เหมาะสมจะพบได้โดยการทำการทดสอบ ข้อผิดพลาด และเป็นหนึ่งในความท้าทายในระบบ MP กราฟต่อไปนี้จะแสดงความสัมพันธ์ระหว่าง ทรูพุตและกระบวนการล็อก



รูปที่ 12. ความสัมพันธ์ระหว่างทรูพุตและกระบวนการขนาดเล็ก. นี่คือภาพประกอบแบบง่ายที่เป็นแผนภูมสองแกน แนวตั้ง หรือ แกน y จะแสดงถึงทรูพุต แนวนอน หรือแกน x จะแสดงถึงกระบวนการขนาดเล็ก ที่มาจาก fine ไปยัง coarse และย้ายออกจากมาตรวัด แผนภูมิระฆังจะแสดงความสัมพันธ์ของกระบวนการขนาดเล็กบนทรูพุต เนื่องจากกระบวนการขนาดเล็กมาจาก fine ไปยัง coarse ทรูพุตจะเพิ่มขึ้นไปยังระดับสูงสุด จากนั้นจะเริ่มลดลง ซึ่งแสดงให้เห็นว่า การยอมรับในกระบวนการขนาดเล็กจำเป็นต้องเข้าถึงทรูพุตสูงสุด

การล็อกการใช้:

การร้องขอให้ล็อก การรอสำหรับล็อก และการปลดล็อกที่เพิ่มการใช้ การประมวลผล

- โปรแกรมที่สนับสนุนการประมวลผลจำนวนมากจะทำการล็อกแบบเดียวกัน และปลดล็อกการประมวลผล แม้ว่าจะกำลังทำงานอยู่ในยูนิโพรเซสเซอร์ หรือเฉพาะผู้ใช้ที่อยู่ในระบบแบบมัลติโพรเซสเซอร์ของล็อกที่เป็นคำถาม
- เมื่อเธรดหนึ่งร้องขอการล็อกที่จัดการโดยเธรดอื่น เธรดที่ร้องขออาจทำงานอยู่ หรือ sleep และหากเป็นไปได้ เธรดอื่นจะถูกจัดส่งแทน ซึ่งจะใช้เวลาของตัวประมวลผล
- การมีอยู่ของของล็อกที่ใช้ซึ่งโยงกับส่วนบนของทรูพุต ของระบบ ตัวอย่างเช่น ถ้าโปรแกรมที่กำหนดต้องจ่าย 20 เปอร์เซ็นต์ของเวลาในการประมวลผล ซึ่งจัดการโดยการล็อกแบบไม่เกิดร่วมกัน ที่อินสแตนซ์ 5 ตัวของโปรแกรมนั้น สามารถรันได้อย่างพร้อมเพียงกัน โดยไม่พิจารณาจำนวนของตัวประมวลผลในระบบ ในความเป็นจริง แม้อินสแตนซ์ทั้ง 5 ตัวจะไม่เคยซิงโครไนซ์มาก่อน เพื่อหลีกเลี่ยงการรออินสแตนซ์อื่น (โปรดดู “การปรับสเกลผลผลิตของหลายตัวประมวลผลได้” ในหน้า 72)

การรอล็อก:

เมื่อเธรดต้องการล็อกที่เป็นของเธรดอื่นแล้ว เธรดจะถูกบล็อก และต้องรอจนกว่าล็อกจะว่าง

การรอมีอยู่สองวิธีดังนี้:

- Spin ล็อกเหมาะสมสำหรับล็อกที่ใช้เพียงชั่วครู่เท่านั้น วิธีนี้อนุญาตให้เธรดที่กำลังรอเก็บตัวประมวลผลไว้ ตรวจสอบล็อกบิตใน tight loop (spin) ซ้ำจนกว่าล็อกจะว่าง Spinning ส่งผลให้ เวลา CPU เพิ่มขึ้น (เวลาระบบสำหรับล็อกเคอร์นัลหรือส่วนขยายเคอร์นัล)
- Sleeping ล็อกเหมาะสมสำหรับล็อกที่ใช้ในรอบเวลานานขึ้น เธรดจะพักจนกว่าล็อกว่างและถูกส่งกลับไปในรันคิว เมื่อล็อกว่าง Sleeping ส่งผลให้เวลา idle มากขึ้น

การรอลดประสิทธิภาพระบบเสมอ ถ้าใช้ spin ล็อก ตัวประมวลผลจะยุ่ง แต่ไม่ได้ทำงานที่เป็นประโยชน์ (ไม่ได้ทำให้เกิดผลผลิต) ถ้าใช้ sleeping ล็อก โอเวอร์เฮดของการสลับบริบทและการจัดส่ง ตลอดจนการเพิ่ม cache misses ในเวลาต่อมาจะเกิดขึ้น

ผู้พัฒนาระบบปฏิบัติการสามารถเลือกระหว่างล็อกสองชนิด: mutually exclusive simple locks ที่อนุญาตให้กระบวนการ spin และพักในขณะที่รอ ให้ล็อกว่าง และ complex read–write locks ที่สามารถ spin และบล็อกกระบวนการในขณะที่รอให้ล็อกว่าง

ระเบียบที่ควบคุมกฎเกี่ยวกับการใช้ล็อก ทั้งฮาร์ดแวร์และซอฟต์แวร์ ไม่มีการบังคับใช้หรือกลไกการตรวจสอบ แม้ว่าการใช้ล็อกทำให้ AIX เวอร์ชัน 4 "MP Safe," ผู้พัฒนามีหน้าที่ในการกำหนดและดำเนินการกลยุทธ์การล็อกที่เหมาะสม เพื่อป้องกันข้อมูลสากลของตนเอง

ความเกี่ยวเนื่องกับแคช

ในการออกแบบมัลติโพรเซสเซอร์ วิศวกรจะให้ความสนใจเป็นพิเศษ เพื่อมั่นใจว่ามีความเกี่ยวเนื่องกับแคช ซึ่งพวกเขาจะประสบความสำเร็จ และความเกี่ยวเนื่องกับแคช จะมีต้นทุนของผลการทำงาน

เราต้องการความเข้าใจถึงปัญหาที่กำลังจู่โจม:

ถ้าตัวประมวลผลแต่ละตัวมีแคชที่สะท้อนถึงสถานะของส่วนของหน่วยความจำที่หลากหลาย อาจเป็นไปได้ที่ แคชตั้งแต่สองแคชขึ้นไปอาจมีสำเนาของบรรทัด ที่เป็นบรรทัดเดียวกัน ซึ่งยังอาจเป็นไปได้ที่ว่า บรรทัดที่กำหนดไว้อาจมีหน่วยข้อมูล ที่สามารถล็อกได้หนึ่งรายการ ถ้าเธรดสองเธรดทำการเปลี่ยนแปลงที่เป็นอนุกรมอย่างเหมาะสมกับหน่วยข้อมูลเหล่านั้น ผลลัพธ์คือ แคชทั้งสองจะสิ้นสุดด้วยความแตกต่าง เวอร์ชันของบรรทัดในหน่วยความจำไม่ถูกต้อง หรืออีกนัยหนึ่ง สถานะของระบบ จะไม่มีความเกี่ยวเนื่องกันอีกต่อไป เนื่องจากระบบมีเวอร์ชันของสิ่งที่สนับสนุนนแตกต่างกัน ซึ่งเป็นเนื้อหาของพื้นที่ของหน่วยความจำเฉพาะ

โซลูชันเกี่ยวกับปัญหาของความเกี่ยวเนื่องกับแคชจะสอดแทรกการตรวจสอบความไม่ถูกต้องทั้งหมด แต่หนึ่งในบรรทัดที่ทำซ้ำเมื่อบรรทัดนั้นถูกแก้ไข แม้ว่าฮาร์ดแวร์จะใช้ตรรกะของการสอบถาม เพื่อตรวจสอบความไม่ถูกต้อง โดยไม่มีการแทรกแซงซอฟต์แวร์ใดๆ ตัวประมวลผลใดๆ ที่มีบรรทัดแคชที่มีการตรวจสอบความไม่ถูกต้องจะมีแคชที่หายไป พร้อมกับหน่วงเวลาครั้งถัดไปที่บรรทัดนั้นถูกกำหนดแอดเดรส

การสอบถาม คือตรรกะที่ใช้เพื่อแก้ปัญหาของความสอดคล้องกันของแคช ตรรกะการสอบถามในตัวประมวลผลจะกระจายข้อความผ่านบัสในแต่ละครั้งที่ คำในแคชได้ถูกแก้ไข ตรรกะการสอบถามยังสอบถามบนบัส ที่มองหาข้อความจากตัวประมวลผลอื่นๆ

เมื่อตัวประมวลผลตรวจพบว่า ตัวประมวลผลอื่นๆ มีการเปลี่ยนแปลงค่าที่แอดเดรส ที่มีอยู่ในแคชของตนเอง ตรรกะการสอบถามจะตรวจสอบความไม่ถูกต้องที่รายการ ในแคช สิ่งนี้เรียกว่า *ข้ามการตรวจสอบความไม่ถูกต้อง* การข้ามการตรวจสอบความไม่ถูกต้อง จะเตือนตัวประมวลผลว่า ค่าที่อยู่ในแคชไม่ถูกต้อง และต้องมองหาค่าที่ถูกต้องในตำแหน่งอื่นๆ (หน่วยความจำหรือแคชอื่นๆ) เนื่องจากการข้ามการตรวจสอบความไม่ถูกต้อง จะเพิ่มแคชที่หายไป และโปรโตคอลการสอบถามจะเพิ่มให้กับทราฟิกของบัส การแก้ปัญหาความสอดคล้องกันของแคชจะลดผลการทำงาน และความสามารถในการวัด SMP ทั้งหมด

Affinity ตัวประมวลผลและการยึด

Affinity ตัวประมวลผล คือความเป็นไปได้ของการจัดส่งเธรด ไปยังตัวประมวลผลที่เคยดำเนินการเธรดนั้นก่อนหน้านี้ ระดับความเข้มของ affinity ตัวประมวลผลควรแตกต่างกันไปโดยแปรผันในทิศทางเดียวกับขนาดชุด ทำงานแคชของเธรด และแปรผันในทิศทางกลับกันกับระยะเวลาตั้งแต่มีการจัดส่ง ครั้งล่าสุด ผู้จัดส่ง AIX เวอร์ชัน 4 บังคับใช้ affinity กับตัวประมวลผล ดังนั้น affinity จึงมีการดำเนินการ โดยระบบปฏิบัติการเอง

ถ้าเธรดถูกขัดจังหวะและจัดส่งใหม่ไปยังตัวประมวลผลเดิมในภายหลัง แคชของตัวประมวลผลอาจจะยังคงมีบรรทัดที่เป็นของเธรดอยู่ ถ้าเธรดถูกจัดส่งไปยังตัวประมวลผลอื่น อาจจะเจอกับชุดของ cache misses จนกว่ามีการดึงข้อมูลชุดทำงานแคชของเธรดนั้นจาก RAM หรือแคชของตัวประมวลผลอื่น ในทางกลับกัน ถ้าเธรดที่จัดส่งได้ ต้องรอจนกว่าตัวประมวลผลที่เคยรันเธรดนั้นก่อนหน้านี้จะพร้อมใช้งาน เธรดอาจล่าช้ามากขึ้น

ระดับ affinity ตัวประมวลผลสูงสุดที่เป็นไปได้คือการยึดเธรดไว้กับ ตัวประมวลผลเฉพาะ การยึดหมายความว่าเธรดจะถูกจัดส่งไปยังตัวประมวลผลนั้น เท่านั้น โดยไม่คำนึงถึงการมีอยู่ของตัวประมวลผลอื่น คำสั่ง **bindprocessor** และ รูทีนย่อย **bindprocessor( )** ยึดเธรด (หรือหลายเธรด) ของ กระบวนการที่ระบุเข้ากับตัวประมวลผลเฉพาะ (โปรดดู "คำสั่ง bindprocessor" ในหน้า 84) การยึดทางอ้อมได้รับมาจากการเรียกระบบ **fork( )** และ **exec( )**

การยึดมีประโยชน์สำหรับโปรแกรมที่เน้น CPU ซึ่งมีการขัดจังหวะ น้อยมาก บางครั้ง การยึดอาจให้ผลทางลบสำหรับโปรแกรมทั่วไป เนื่องจาก อาจทำให้การจัดส่งใหม่ของเธรดหลังจาก I/O ล่าช้าออกไปจนกว่าตัวประมวลผล ซึ่งเธรดถูกยึดไว้จะพร้อมใช้งาน ถ้าเธรดถูกบล็อกไว้ในช่วงเวลาของการดำเนินงาน I/O มีความเป็นไปได้น้อยที่บริบทการประมวล ผลของเธรดนั้นจะยังคงอยู่ในแคชของตัวประมวลผลซึ่งยึดเข้ากับเธรดนั้น เธรดอาจ ทำงานได้ดีขึ้นถ้าถูกจัดส่งไปยังตัวประมวลที่พร้อมใช้งาน ตัวถัดไป

หน่วยความจำและ contention บัส

สำหรับยูนิโพรเซสเซอร์ contention สำหรับรีซอร์สภายใน เช่น ที่เก็บหน่วยความจำ และ I/O หรือบัสหน่วยความจำ จะเป็นคอมโพเนนต์ส่วนน้อยที่ใช้เวลา สำหรับมัลติโพรเซสเซอร์ ผลกระทบเหล่านี้อาจกลายเป็นสิ่งที่สำคัญ โดยเฉพาะอย่างยิ่ง หากอัลกอริธึมความสัมพันธ์ของแคชเพิ่มให้กับจำนวนของการเข้าถึง RAM

ประเด็นประสิทธิภาพ SMP

มีหลายสิ่งที่ต้องพิจารณาเพื่อให้สามารถใช้ SMP ได้อย่างมีประสิทธิภาพ

การเกิดพร้อมกันของเวิร์กโหลด

ประเด็นประสิทธิภาพหลักเฉพาะสำหรับระบบ SMP คือ การเกิดพร้อมกันของเวิร์กโหลด ซึ่งสามารถอธิบายได้ว่า "ขณะนี้เรามี *n* ตัวประมวลผล เราจะใช้งานทุกตัวพร้อมกันได้อย่างไร"?

ถ้าตัวประมวลผลเดียวในระบบหลายตัวประมวลผล four-way ทำงานที่มีประโยชน์ได้ตลอดเวลา ไม่มีอะไรดีไปกว่าระบบตัวประมวลผลเดียว ซึ่งอาจจะแย่ลงเนื่องจากรหัสพิเศษที่ใช้เพื่อหลีกเลี่ยงการรบกวนระหว่าง ตัวประมวลผล

การเกิดพร้อมกันของเวิร์กโหลดเป็นส่วนเสริมของการจัดลำดับ หากซอฟต์แวร์ระบบ หรือแอ็พพลิเคชันเวิร์กโหลด (หรือการโต้ตอบระหว่างสองรายการ) ต้องการการจัดลำดับ การเกิดพร้อมกันของเวิร์กโหลดจะได้รับผลกระทบ

การเกิดพร้อมกันของเวิร์กโหลดยังอาจลดลง เนื่องจาก affinity ตัวประมวลผล ที่เพิ่มขึ้นซึ่งเป็นที่ต้องการมากกว่า ประสิทธิผลแคชที่พัฒนาขึ้นอันเป็นผลมาจาก affinity ตัวประมวลผล อาจส่งผลให้โปรแกรมสมบูรณ์เร็วขึ้น การเกิดพร้อมกันของเวิร์กโหลดลดลง (ยกเว้นว่ามีเธรดที่จัดส่งได้มากขึ้น) แต่เวลาการตอบกลับ ดีขึ้น

คอมโพเนนต์ของการเกิดพร้อมกันของเวิร์กโหลด *การเกิดพร้อมกันของกระบวนการ* คือระดับ ซึ่งกระบวนการมัลติเธรดมีหลายเธรดที่จัดส่งได้ ตลอดเวลา

ผลผลิต

ผลผลิตของระบบ SMP ส่วนใหญ่แล้วขึ้นอยู่กับปัจจัย หลายอย่าง

- ระดับที่สูงอย่างต่อเนื่องของเวิร์กโหลดพร้อมกัน การมีเธรดที่จัดส่งได้มากกว่า ตัวประมวลผลในบางเวลาไม่สามารถชดเชยตัวประมวลผล idle ใน เวลาอื่นได้
- จำนวนของการช่วงชิงล็อก
- ระดับของ affinity ตัวประมวลผล

เวลาตอบกลับ

เวลาตอบกลับของโปรแกรมเฉพาะในระบบ SMP ขึ้นอยู่กับปัจจัย หลายอย่าง

- ระดับการเกิดขึ้นพร้อมกันของกระบวนการของโปรแกรม ถ้าโปรแกรมมีเธรดที่จัดส่งได้ ตั้งแต่สองตัวขึ้นไปอย่างสม่ำเสมอ เวลาตอบกลับของโปรแกรมจะดีขึ้น ในสภาพแวดล้อม SMP ถ้าโปรแกรมมีเธรดเดียว เวลาตอบกลับของโปรแกรมจะดีที่สุด เมื่อเปรียบเทียบกับเวลาตอบกลับในตัวประมวลผลเดียวที่มีความเร็วเท่ากัน
- จำนวนการช่วงชิงล็อกของอินสแตนซ์อื่นของโปรแกรม หรือ การช่วงชิงกับโปรแกรมอื่นที่ใช้ล็อกเดียวกัน
- ระดับของ affinity ตัวประมวลผลของโปรแกรม ถ้าแต่ละการจัดส่งของโปรแกรมคือ การจัดส่งไปยังตัวประมวลผลอื่นซึ่งไม่มีรูปแบบแคชของโปรแกรม โปรแกรมอาจรันช้าลงกว่าในตัวประมวลผลเดียวที่มีความเร็วเท่ากัน

SMP เวิร์กโหลด

ผลกระทบต่อประสิทธิภาพจากตัวประมวลผลที่เพิ่มขึ้นส่วนใหญ่แล้ว มาจากลักษณะบางอย่างของเวิร์กโหลดเฉพาะที่กำลังจัดการ ส่วนนี้ อธิบายลักษณะที่สำคัญเหล่านี้และผลกระทบ

คำศัพท์ต่อไปนี้ใช้เพื่ออธิบายขอบเขตการแก้ไขโปรแกรมที่มีอยู่ หรือ การออกแบบโปรแกรมใหม่เพื่อดำเนินงานในสภาพแวดล้อม SMP:

ความปลอดภัยของ SMP

การหลีกเลี่ยงการดำเนินการใดๆ ในโปรแกรม เช่น การเข้าถึงแบบไม่จัดลำดับ เพื่อแบ่งใช้ข้อมูล ซึ่งอาจทำให้เกิดปัญหาการทำงานขึ้นในสภาพแวดล้อม SMP คำศัพท์นี้ เมื่อใช้เพียงคำเดียว โดยปกติแล้ว หมายถึงโปรแกรมที่ต้องทำการเปลี่ยนแปลง เพียงเล็กน้อยเท่านั้นเพื่อให้สามารถทำงานได้อย่างถูกต้องในสภาพแวดล้อม SMP

ประสิทธิผลของ SMP

การหลีกเลี่ยงการดำเนินการใดๆ ในโปรแกรม ซึ่งอาจทำให้เกิดปัญหาการทำงานหรือประสิทธิภาพขึ้นในสภาพแวดล้อม SMP โปรแกรมที่มีการอธิบายเป็นประสิทธิผลของ SMP เป็นความปลอดภัยของ SMP เช่นกัน โปรแกรมแบบประสิทธิผลของ SMP โดยทั่วไป ต้องมีการเปลี่ยน แปลงเพิ่มเติมเพื่อลดปัญหาคอขวดที่กำลังจะก่อตัวขึ้น

การใช้ประโยชน์ของ SMP

การเพิ่มคุณลักษณะในโปรแกรมที่สร้างขึ้นมาสำหรับการใช้สภาพแวดล้อม SMP โดยเฉพาะ เช่น multithreading โปรแกรมที่มีการอธิบายเป็น การใช้ประโยชน์ของ SMP โดยทั่วไปแล้ว ถือว่าเป็นความปลอดภัยของ SMP และประสิทธิผลของ SMP เช่นกัน

Workload multiprocessing

ระบบปฏิบัติการแบบหลายโปรแกรมที่กำลังรันเวิร์กโหลดหนักบนคอมพิวเตอร์ ที่รวดเร็วสร้างความประทับใจให้แก่ผู้ใช้งานโดยการทำงานหลายอย่าง พร้อมกัน

ในข้อเท็จจริง เวิร์กโหลดที่มีความต้องการสูงหลายรายการไม่มีเธรดที่จัดส่งได้ จำนวนมาก ณ เวลาที่กำหนดใดๆ แม้ในขณะที่รันบนระบบตัวประมวลผลเดียว ซึ่งการจัดลำดับมีปัญหาน้อย ยกเว้นว่ามีเธรดที่จัดส่งได้ในจำนวนมากเท่ากับ ตัวประมวลผลเป็นอย่างน้อยเสมอ ตัวประมวลผลตั้งแต่หนึ่งตัวขึ้นไปจะ idle ในบางเวลา

จำนวนของเธรดที่จัดส่งได้คือจำนวนทั้งหมดของเธรดใน ระบบ

- ลบจำนวนเธรดที่กำลังรอ I/O
- ลบจำนวนเธรดที่กำลังรอรีซอร์สแบบแบ่งใช้
- ลบจำนวนเธรดที่กำลังรอผลลัพธ์ของเธรด อื่น
- ลบจำนวนเธรดที่กำลังพักอยู่ที่การร้องขอของตนเอง

อาจกล่าวได้ว่าเวิร์กโหลดมีความสามารถหลายกระบวนการ เมื่อเวิร์กโหลดสามารถ นำเสนอเธรดที่จัดส่งได้ในจำนวนที่มาก เท่ากับจำนวนตัวประมวลผลในระบบได้ตลอดเวลา หมายเหตุว่าความสามารถนี้ไม่ได้หมายความเพียงแค่จำนวนเฉลี่ยของเ ธรดที่จัดส่งได้ เท่ากับจำนวนตัวประมวลผลเท่านั้น ถ้าจำนวนของเธรดที่จัดส่งได้เป็นศูนย์ ครึ่งเวลา และจำนวนตัวประมาณ มากเป็นสองเท่าและใช้เวลาที่เหลืออีกครึ่งหนึ่ง จำนวนเฉลี่ยของเธรดที่จัดส่งได้จะเท่ากับจำนวนตัวประมวลผล แต่ตัว ประมวลผล ที่กำหนดใดๆ ในระบบจะทำงานเพียงครึ่งเวลาเท่านั้น

การเพิ่ม multiprocessability ของเวิร์กโหลดเกี่ยวข้องกับปัจจัยหนึ่งหรือทั้งสอง อย่างต่อไปนี้:

- การระบุและการแก้ไขปัญหาคอขวดที่ทำให้เธรดต้องรอ
- การเพิ่มจำนวนทั้งหมดของเธรดในระบบ

วิธีแก้ไขเหล่านี้มีความเชื่อมโยงกัน ถ้ามีปัญหาคอขวดที่ระบบหลักหนึ่ง การเพิ่มจำนวนเธรดของเวิร์กโหลดที่มีอยู่ซึ่งเดินทาง ผ่านจุด ปัญหาคอขวดจะเป็นการเพิ่มจำนวนการรอเธรด เท่านั้น ถ้าไม่มีปัญหาคอขวดในปัจจุบัน การเพิ่มจำนวนเธรด อาจ สร้างเธรดขึ้นหนึ่งรายการ

การปรับสเกลผลผลิตของหลายตัวประมวลผลได้

เวิร์กโหลดที่แท้จริงไม่ได้กำหนดสเกลอย่างสมบูรณ์บนระบบ SMP

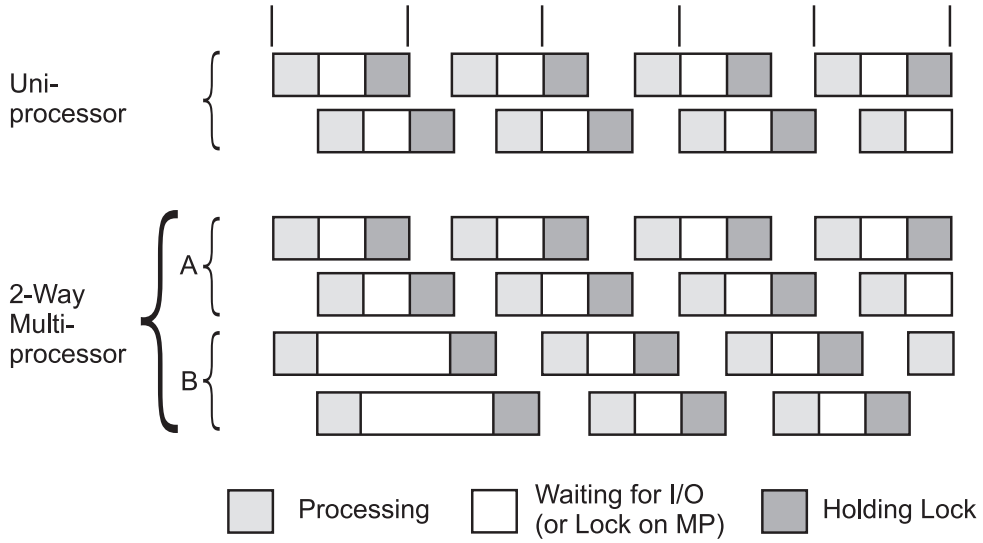
ปัจจัยบางอย่างที่เป็นอุปสรรคต่อการกำหนดสเกลอย่างสมบูรณ์มีดังนี้:

- การช่วงชิงบัส/สวิตช์เพิ่มขึ้นในขณะที่จำนวนของตัวประมวลผล เพิ่มมากขึ้น
- การช่วงชิงหน่วยความจำเพิ่มขึ้น (หน่วยความจำทั้งหมดมีการแบ่งใช้โดยทุก ตัวประมวลผล)
- ต้นทุนที่เพิ่มขึ้นของ cache misses เนื่องจากหน่วยความจำห่างไกลมากขึ้น
- ความไม่ถูกต้องและการอ่านข้ามแคชจากแคชอื่นเพื่อรักษา ความเชื่อมโยงของแคช
- Cache misses ที่เพิ่มขึ้นเนื่องจากอัตราการจัดส่งที่สูงขึ้น (มีกระบวนการ /เธรดที่ต้องจัดส่งมากขึ้นบนระบบ)
- ต้นทุนที่เพิ่มสูงขึ้นของคำสั่งการซิงโครไนซ์
- Cache misses ที่เพิ่มขึ้นเนื่องจากโครงสร้างของระบบปฏิบัติการและ ข้อมูลแอ็พพลิเคชันที่ใหญ่ขึ้น
- ความยาวพาธที่เพิ่มขึ้นของระบบปฏิบัติการและแอ็พพลิเคชันสำหรับการล็อก-ปลดล็อก
- ความยาวพาธที่เพิ่มขึ้นของระบบปฏิบัติการและแอ็พพลิเคชันที่กำลังรอ การล็อก

ปัจจัยทั้งหมดเหล่านี้เป็นส่วนประกอบของสิ่งที่เรียกว่า *การปรับสเกลได้* ของ เวิร์กโหลด การปรับสเกลได้คือระดับซึ่งผลผลิตเ วิร์กโหลด ได้รับประโยชน์จากการมีอยู่ของตัวประมวลผลเพิ่มเติม โดยปกติแล้ว มีการระบุเป็นผลหารของผลผลิตของเวิร์ก โหลดบนหลายตัวประมวลผล หารด้วยผลผลิตบนตัวประมวลผลแบบเดียวกันเพียงตัวเดียว ตัวอย่างเช่น ถ้าตัวประมวลผลตัว เดียวสามารถดำเนินการได้ 20 การร้องขอต่อวินาทีบนเวิร์กโหลดที่กำหนด และระบบที่มีตัวประมวลผลสี่ตัวสามารถดำเนิน การได้ 58 การร้องขอต่อวินาที ปัจจัยการปรับสเกล จะเป็น 2.9 ซึ่งแสดงว่าเวิร์กโหลดนั้นสามารถปรับสเกลได้สูง เวิร์กโหลดที่ มีเฉพาะ โปรแกรมด้านการคำนวณที่ใช้เวลาการรันยาวนาน ซึ่งมี I/O ที่ปฏิเสธได้หรือกิจกรรมเคอร์นัลอื่นและไม่มีข้อมูลแบบ

แบ่งใช้อาจมี ปัจจัยการปรับสเกลสูงถึง 3.2 ถึง 3.9 บนระบบ 4-way อย่างไรก็ตาม เวิร์กโหลดส่วนใหญ่ในโลกที่แท้จริงไม่เป็นเช่นนั้น เนื่องจากการปรับสเกลได้เป็นสิ่งที่ประเมินได้ ยากมาก ดังนั้นสมมุติฐานของการประเมินได้จึงควรสร้างจากข้อมูลการประเมิน ของเวิร์กโหลดที่พิสูจน์ตัวตน

รูปภาพต่อไปนี้แสดงปัญหาของการปรับสเกล เวิร์กโหลด ประกอบด้วยชุดของคำสั่งแบบสมมุติฐาน แต่ละคำสั่งเกี่ยวกับการประมวลผล ปกติจำนวนหนึ่งในสาม การรอ I/O หนึ่งในสาม และการประมวลผลที่มีการล็อก จำนวนหนึ่งในสาม บนระบบที่มีตัวประมวลผลเดียว สามารถประมวลผลคำสั่งจริง ได้ทีละหนึ่งคำสั่งเท่านั้น โดยไม่คำนึงว่ามีการล็อกอยู่หรือไม่ ใน ช่วงเวลาที่แสดง (ห้าคูณเวลาดำเนินการแบบสแตนด์อะโลนของ คำสั่ง) ระบบตัวประมวลผลเดียวสามารถจัดการได้ 7.67 ของคำสั่ง



รูปที่ 13. การปรับสเกลหลายตัวประมวลผล. รูปภาพนี้แสดงปัญหา ของการปรับสเกล เวิร์กโหลดประกอบด้วยชุดของคำสั่ง แบบสมมุติฐาน แต่ละคำสั่งเกี่ยวกับการประมวลผลปกติจำนวนหนึ่งในสาม การรอ I/O หนึ่งในสาม และการประมวลผลที่มีการล็อกจำนวนหนึ่งในสาม บนระบบที่มีตัวประมวลผลเดียว สามารถประมวลผลคำสั่งจริงได้ทีละหนึ่งคำสั่งเท่านั้น โดยไม่คำนึงว่า มีการล็อกอยู่หรือไม่ ในช่วงเวลาเดียวกัน ระบบตัวประมวลผลเดียว สามารถจัดการได้ 7.67 ของคำสั่ง ในรอบเวลาเดียวกันนั้น ระบบแบบหลายตัวประมวลผล สามารถจัดการได้ 14 คำสั่งสำหรับปัจจัยการปรับสเกล 1.83

บนระบบแบบหลายตัวประมวลผล มีตัวประมวลผลที่จัดการดำเนินการของโปรแกรมอยู่สองตัว แต่ยังคงมีตัวประมวลผลที่ถูกล็อกไว้ตัวเดียวเท่านั้น เพื่อให้เข้าใจได้ง่ายขึ้น มีการแสดงการช่วงชิงล็อกทั้งหมด ซึ่งส่งผลกระทบต่อตัวประมวลผล B ในรอบเวลาที่แสดง ระบบแบบหลาย ตัวประมวลผลจัดการได้ 14 คำสั่ง ดังนั้นปัจจัยการปรับสเกลคือ 1.83 เราขอหยุดที่สองตัวประมวลผล เนื่องจากถ้าตัวประมวลผลมากขึ้นจะทำให้สถานการณ์เปลี่ยนแปลงไป ขณะนี้ มีการใช้เวลาล็อกครบทั้ง 100 เปอร์เซ็นต์ ในหลายตัวประมวลผลแบบ four-way ปัจจัยการปรับสเกลจะเป็น 1.83 หรือน้อยกว่า

โปรแกรมจริงมักไม่ค่อยจะสมมาตรดังเช่นคำสั่งในภาพ สาธิต นอกจากนี้ เราพิจารณาการช่วงชิงเพียงมิติเดียวเท่านั้น คือ: การล็อก หากเราคำนึงถึงผลกระทบจากความเชื่อมโยงของแคชและความเกี่ยวข้องของ ตัวประมวลผลด้วย ปัจจัยการปรับสเกลจะต่ำกว่าค่านี้อย่างแน่นอน

ตัวอย่างนี้สาธิตให้เห็นว่า บ่อยครั้งที่ไม่สามารถทำให้เวิร์กโหลดรัน เร็วขึ้นโดยเพียงแต่เพิ่มตัวประมวลผล สิ่งที่ยังจำเป็นต้องทำคือการระบุ และการลดแหล่งของการช่วงชิงระหว่างเธรดให้เหลือน้อยที่สุด

การปรับสเกลขึ้นอยู่กับเวิร์กโหลด ผลของเกณฑ์เปรียบเทียบที่เผยแพร่บางอย่าง ระบุว่าการปรับสเกลได้ระดับสูงสามารถบรรลุผลได้ง่าย เกณฑ์เปรียบเทียบดังกล่าวส่วนใหญ่ สร้างขึ้นจากการรันชุดของโปรแกรมขนาดเล็กที่เน้น CPU ซึ่งแทบจะไม่ใช้ การบริการเคอร์นัลเลย ผลของเกณฑ์เปรียบเทียบเหล่านี้ แสดงถึงขอบเขตระดับบนของการปรับสเกลได้ ไม่ใช่ค่าที่จะได้รับจริง

จุดที่น่าสนใจอีกอย่างหนึ่งของหมายเหตุเกี่ยวกับเกณฑ์เปรียบเทียบคือ โดยทั่วไป one-way SMP จะรันช้ากว่า (ประมาณ 5-15 เปอร์เซ็นต์) ระบบที่มีตัวประมวลผล แบบเดียวกันเพียงตัวเดียวซึ่งกำลังรันเวอร์ชัน UP ของระบบปฏิบัติการ

เวลาตอบกลับของหลายตัวประมวลผล

ระบบแบบหลายตัวประมวลผลสามารถปรับปรุงเวลาดำเนินการของ แต่ละโปรแกรมภายใต้ขอบเขตที่โปรแกรมสามารถรันในโหมดหลายเธรดได้ เท่านั้น

การดำเนินการแบบขนานของส่วนประกอบต่างๆ ของโปรแกรมเดียวสามารถทำได้ หลายวิธีดังนี้:

- การเรียกไปยังรูทีนย่อย `libpthreads.a` ด้วยตนเอง (หรือในโปรแกรมรุ่นเก่า ให้เรียกไปยังรูทีนย่อย **fork()**) เพื่อสร้างหลายเธรดที่รันพร้อมกัน
- การประมวลผลโปรแกรมด้วยคอมไพเลอร์หรือตัวประมวลผลล่วงหน้าแบบขนาน ที่ตรวจพบลำดับของรหัสที่สามารถดำเนินการพร้อมกันได้ และสร้างหลายเธรดที่จะรันคู่ขนานกันไป
- การใช้ซอฟต์แวร์แพ็กเกจที่มีหลายเธรดในตัวอยู่แล้ว

ยกเว้นว่าใช้เทคนิคเหล่านี้หนึ่งเทคนิคขึ้นไป โปรแกรมจะรันไม่เร็วขึ้นใน ระบบแบบหลายตัวประมวลผลเมื่อเปรียบเทียบกับในระบบที่มีตัวประมวลผลแบบเดียวกันเพียงตัวเดียว ในข้อเท็จจริง ระบบแบบหลายตัวประมวลผลอาจประสบกับการล็อกโอเวอร์เฮดและความล่าช้ามากกว่า เนื่องจากการจัดส่งข้อมูลไปยังตัวประมวลผลต่างๆ ในเวลาที่แตกต่างกัน ดังนั้นจึงอาจทำงานได้ช้ากว่า

แม้ว่ามีการใช้เทคนิคที่เหมาะสมทั้งหมดแล้วก็ตาม แต่การพัฒนาสูงสุด ถูกจำกัดโดยกฎที่เรียกว่ากฎของ Amdahl:

- ถ้าเศษส่วน *x* ของเวลาดำเนินการตัวประมวลผลเดียวของโปรแกรม *t* สามารถประมวลผลแบบตามลำดับเท่านั้น การพัฒนาในเวลาดำเนินการในระบบแบบ หลายตัวประมวลผล n-way เมื่อเปรียบเทียบกับระบบที่มีตัวประมวลผลแบบเดียวกันเพียงตัวเดียว (ความเร็วที่เพิ่มขึ้น) มีการกำหนดโดยสมการดังนี้:

$$\text{speed up} = \frac{\text{uniprocessor time}}{\text{seq time + mp time}} = \frac{t}{xt + \frac{(x-1)t}{n}} = \frac{1}{x + \frac{x}{n}}$$

$$\lim_{n \to \infty} \text{speed-up} = \frac{1}{x}$$

รูปที่ 14. กฎของ Amdahl. กฎของ Amdahl ระบุว่าความเร็วที่เพิ่มขึ้น เท่ากับเวลาตัวประมวลผลเดียวหารด้วยเวลาในลำดับบวกกับเวลาของหลายตัวประมวล ผล หรือ 1 หารด้วย x บวก (x บน n) ขีดจำกัดความเร็วที่เพิ่มขึ้นเท่ากับ 1 หาร ด้วย x และ n จะเท่ากับอินฟินิตี

ตัวอย่างเช่น ถ้า 50 เปอร์เซ็นต์ของการประมวลผลของโปรแกรมต้องทำ ตามลำดับ และอีก 50 เปอร์เซ็นต์สามารถทำคู่ขนานพร้อมกันได้ การพัฒนาเวลา การตอบกลับสูงสุดจะน้อยกว่าปัจจัยของ 2 (ในระบบหลายตัวประมวลผลแบบ-idle 4-way ค่าสูงสุดของปัจจัยนี้คือ 1.6)

การจัดตารางเวลา SMP เธรด

ในสภาพแวดล้อม SMP การสนับสนุนการมีอยู่ของเธรดช่วยให้สามารถ ดำเนินการแอ็พพลิเคชันที่ใช้ประโยชน์ SMP ได้ง่ายขึ้น และถูกลง

การสนับสนุนเธรดแบ่งการควบคุมการดำเนินการของโปรแกรมออกเป็นสององค์ประกอบดังนี้:

- *กระบวนการ* คือชุดของรีซอร์สฟิสิคัลที่ต้องใช้ในการรัน โปรแกรม เช่น หน่วยความจำและการเข้าถึงไฟล์
- *เธรด* คือสถานะการดำเนินการของอินสแตนซ์ของโปรแกรม เช่น เนื้อหาปัจจุบันของทะเบียนการระบุคำสั่งและทะเบียน วัตถุประสงค์ ทั่วไป แต่ละเธรดรันภายในบริบทของกระบวนการที่กำหนดและใช้รีซอร์สของกระบวนการนั้น สามารถรันได้ หลายเธรดภายในกระบวนการเดียว ที่แบ่งใช้รีซอร์ส

Forking หลายกระบวนการเพื่อสร้างหลายโฟลว์การควบคุมเป็นเรื่องยาก และใช้ค่าใช้จ่ายสูง เนื่องจากแต่ละกระบวนการมีชุด ของรีซอร์สหน่วยความจำของ ตนเอง และต้องตั้งค่ากระบวนการระบบจำนวนมาก การสร้างหลายเธรด ภายในกระบวนการ เดียวใช้การประมวลผลและหน่วยความจำน้อยกว่า

การสนับสนุนเธรดมีอยู่สองระดับคือ:

- การสนับสนุน `libpthreads.a` ในสภาพแวดล้อมแอ็พพลิเคชัน โปรแกรม
- การสนับสนุนเคอร์นัลเธรด

แม้ว่าโดยทั่วไปแล้ว เธรดเป็นกลไกที่สะดวกและมีประสิทธิภาพในการใช้ ประโยชน์หลายการประมวลผล แต่เธรดมีข้อจำกัด ในการปรับสเกลได้ เนื่องจากเธรด แบ่งใช้รีซอร์สและสถานะของกระบวนการ ดังนั้นการล็อกและการจัดลำดับของ รีซอร์ส เหล่านี้จึงอาจจำกัดการปรับสเกลได้ในบางครั้ง

การประมวลผลตัวจัดตารางเวลาดีฟอลต์ของเวิร์กโหลดที่ย้าย

การแบ่งแยกระหว่างกระบวนการและเธรดไม่สามารถเห็นได้ในโปรแกรม ที่มีอยู่

ในข้อเท็จจริง เวิร์กโหลดที่ย้ายโดยตรงจากรีลีสก่อนหน้านี้ของระบบปฏิบัติการ สร้างกระบวนการในวิธีที่เคยทำเสมอมา กระบวนการใหม่แต่ละรายการมีการสร้างขึ้น ด้วยเธรดเดียว (เธรดแรกเริ่ม) ที่ช่วงชิง CPU กับเธรดของ กระบวนการอื่น

ดีฟอลต์แอ็ตทริบิวต์ของเธรดแรกเริ่มประกอบกับขั้นตอนวิธีของตัวจัดตารางเวลา ใหม่ ช่วยลดการเปลี่ยนแปลงในไดนามิก ระบบสำหรับเวิร์กโหลดที่ไม่เปลี่ยนแปลงให้เหลือน้อยที่สุด

ระดับความสำคัญสามารถจัดการโดยใช้คำสั่ง **nice** และ **renice** และการเรียกระบบ **setpri( )** และ **setpriority( )** ดังเช่นก่อน ตัวจัดตารางเวลาอนุญาตให้เธรดที่กำหนดรันนานที่สุดหนึ่งส่วนเวลา (โดยปกติ 10 ms) ก่อนบังคับให้เธรดนั้นให้เธรดที่จัดส่ง ได้ถัดไปซึ่งมีระดับความสำคัญเท่ากัน หรือสูงกว่า โปรดดู “การควบคุม contention สำหรับไมโครโพรเซสเซอร์” ในหน้า 132 สำหรับรายละเอียดเพิ่มเติม

ตัวแปรขั้นตอนวิธีการจัดตารางเวลา

ตัวแปรหลายตัวมีผลกระทบต่อการจัดตารางเวลาของเธรด

บางตัวเกี่ยวข้องกับการสนับสนุนของเธรด บางตัวเป็นข้อควรพิจารณาเกี่ยวกับ ความซับซ้อนของการจัดตารางเวลากระบวน การ:

ระดับความสำคัญ
ค่าระดับความสำคัญของเธรดเป็นตัวบ่งชี้พื้นฐานของการมีสิทธิก่อนใน การช่วงชิงเวลาตัวประมวลผล

ตำแหน่งในรันคิวของตัวจัดตารางเวลา
ตำแหน่งของเธรดในคิวของตัวจัดตารางเวลาของเธรดที่จัดส่งได้สะท้อนถึง จำนวนของเงื่อนไขก่อนหน้า

นโยบายการจัดตารางเวลา
แอ็ตทริบิวต์เกี่ยวกับเธรดนี้กำหนดสิ่งที่จะเกิดขึ้นกับเธรดที่กำลังรันเมื่อสิ้นสุด ส่วนเวลา

ขอบเขตการช่วงชิง
ขอบเขตการช่วงชิงของเธรดกำหนดว่า เธรดจะแข่งขันเฉพาะกับเธรดอื่นภายใน กระบวนการของตนหรือแข่งขันกับเธรดทั้งหมดในระบบ เธรดที่สร้างขึ้น โดยมีขอบเขตการช่วงชิงเป็นกระบวนการ มีการจัดตารางเวลาโดยไลบรารี ในขณะที่ เธรดที่สร้างขึ้นโดยมีขอบเขตเป็นระบบ มีการจัดตารางเวลาโดยเคอร์นัล ตัวจัดตารางเวลาไลบรารี ใช้พูลของเคอร์นัลเธรดเพื่อจัดตารางเวลา pthreads ที่มีขอบเขตเป็นกระบวนการ โดยทั่วไป ให้สร้าง pthreads ที่มีขอบเขตเป็นกระบวนการ ถ้ากำลังทำ I/O ขอบเขตเป็นกระบวนการมีประโยชน์ เมื่อมีการซิงโครไนซ์ระหว่างกระบวนการเป็นจำนวนมาก ขอบเขตการช่วงชิงคือหลักการ `libpthreads.a`

Affinity ตัวประมวลผล
ระดับการบังคับใช้ affinity ที่กระทบต่อประสิทธิภาพ

ชุดของข้อควรพิจารณาเหล่านี้อาจดูซับซ้อน แต่คุณสามารถเลือกจากแนวทางที่ แตกต่างกันสามแนวทางดังต่อไปนี้ เมื่อคุณกำลังจัดการกระบวนการที่กำหนด:

ดีฟอลต์
กระบวนการมีหนึ่งเธรดที่มีระดับความสำคัญแตกต่างกันไปตามการใช้ CPU และมีนโยบายการจัดตารางเวลาเป็น SCHED_OTHER

ควบคุมระดับกระบวนการ
กระบวนการอาจมีเธรดตั้งแต่หนึ่งรายการขึ้นไป แต่นโยบายการจัดตารางเวลาของ เธรดเหล่านี้ถูกปล่อยไว้เป็น SCHED_OTHER ดีฟอลต์ ซึ่งอนุญาตให้ใช้วิธีการที่มี อยู่ของการควบคุมค่า nice และระดับความสำคัญคงที่ วิธีการทั้งหมด เหล่านี้ส่งผลกระทบต่อเธรดทั้งหมดในกระบวนการเหมือนกัน ถ้ามีการใช้รูทีนย่อย **setpri( )** นโยบายการจัดตารางเวลาของเธรดทั้งหมดในกระบวนการ จะมีการตั้งค่าเป็น SCHED_RR

ควบคุมระดับเธรด
กระบวนการอาจมีเธรดตั้งแต่หนึ่งรายการขึ้นไป นโยบายการจัดตารางเวลาของเธรดเหล่านี้ มีการตั้งค่าเป็น SCHED_RR หรือ SCHED_FIFO*n* ตามความเหมาะสม ระดับความสำคัญ ของแต่ละเธรดมีการระบุค่าไว้และจัดการด้วยรูทีนย่อยระดับเธรด

นโยบายการจัดตารางเวลามีการอธิบายไว้ใน “นโยบายการจัดตารางเวลาสำหรับเธรด” ในหน้า 47

การปรับเธรด

เธรดผู้ใช้แสดงลำดับของการควบคุมที่เป็นอิสระภายในกระบวนการหนึ่ง

ถ้าเธรดผู้ใช้ต้องการเข้าถึงการบริการเคอร์นัล (เช่น การเรียกระบบ) เธรดผู้ใช้จะได้รับการบริการจากเคอร์นัลเธรดที่เชื่อมโยง เธรดผู้ใช้ มีการจัดเตรียมไว้ในซอฟต์แวร์แพ็กเกจหลายชนิด พร้อมกับไลบรารีที่โดดเด่นที่สุด คือไลบรารีแบบแบ่งใช้ pthreads (`libpthreads.a`) ด้วยการใช้ libpthreads เธรดผู้ใช้มีตำแหน่งอยู่ด้านบนของตัวประมวลผลเสมือน (VP) ซึ่งอยู่บนเคอร์นัลเธรดอีกทีหนึ่ง กระบวนการผู้ใช้แบบมัลติเธรด สามารถใช้โมเดลอย่างใดอย่างหนึ่งจากสองโมเดลดังนี้:

โมเดลเธรด 1:1

โมเดล 1:1 บ่งชี้ว่าเธรดผู้ใช้แต่ละรายการจะมีการแม็พเข้ากับเคอร์นัลเธรดเฉพาะ หนึ่งรายการ โมเดลนี้เป็นโมเดลดีฟอลต์บน AIX ทุกเวอร์ชัน ในโมเดลนี้ เธรดผู้ใช้แต่ละรายการถูกยึดไว้กับ VP และลิงก์กับเคอร์นัลเธรดเฉพาะหนึ่งรายการ VP ไม่จำเป็นต้องยึดกับ CPU จริง (ยกเว้นว่าทำการยึดกับตัวประมวลผล) อาจกล่าวได้ว่าเธรดที่ถูกยึดไว้กับ VP มีขอบเขตระบบ เนื่องจาก มีการจัดตารางเวลาพร้อมกับเธรดผู้ใช้อื่นทั้งหมดโดยตัวจัดตารางเวลาเคอร์นัล

โมเดลเธรด M:N

โมเดล M:N มีการนำมาใช้ใน AIX 4.3.1 และ ยังเป็นดีฟอลต์โมเดลในปัจจุบันด้วย ในโมเดลนี้ เธรดผู้ใช้หลายรายการสามารถแบ่งใช้ ตัวประมวลผลเสมือนเดียวกันหรือพูลเดียวกันของ VPs แต่ละ VP ทำหน้าที่เหมือนกับ CPU เสมือนที่มีอยู่สำหรับการดำเนินการรหัสผู้ใช้และการเรียกระบบ อาจกล่าวได้ว่าเธรดที่ถูกยึดไว้กับ VP มีขอบเขตโลคัลหรือกระบวนการ เนื่องจาก ไม่ได้มีการจัดตารางเวลาพร้อมกับเธรดอื่นทั้งหมดโดยตัวจัดตารางเวลาเคอร์นัล ไลบรารี pthreads จะจัดการจัดตารางเวลาของเธรดผู้ใช้กับ VP จากนั้น เคอร์นัลจะจัดตารางเวลาเคอร์นัลเธรดที่เชื่อมโยง สำหรับ AIX 4.3.2 ค่าดีฟอลต์คือมีหนึ่ง เคอร์นัลเธรดที่แม็พกับแปดเธรดผู้ใช้ ค่านี้สามารถปรับได้จากภายในแอ็พพลิเคชัน หรือผ่านทางตัวแปรสภาพแวดล้อม

ขึ้นอยู่กับชนิดของแอ็พพลิเคชัน ผู้ดูแลระบบสามารถเลือกที่จะใช้ เธรดโมเดลที่แตกต่างอื่นก็ได้ การทดสอบแสดงว่าบางแอ็พพลิเคชันสามารถทำงานได้ดีหว่ามากโดยใช้โมเดล 1:1 โมเดลเธรดดีฟอลต์ถูกเปลี่ยนกลับเป็น 1:1 จาก M:N ใน AIX 6.1 สำหรับ AIX ทุกเวอร์ชัน โดยการตั้งค่าตัวแปรสภาวะแวดล้อม **AIXTHREAD_SCOPE=S** สำหรับกระบวนการเฉพาะ เราสามารถตั้งค่าโมเดลเธรดเป็น 1:1 และจากนั้น เปรียบเทียบประสิทธิภาพกับประสิทธิภาพก่อนหน้านี้เมื่อโมเดลเธรดเป็น M:N

ถ้าคุณเห็นแอ็พพลิเคชันกำลังสร้างและลบเธรด อาจแสดงว่าเคอร์นัลเธรด ถูกใช้งานอย่าง *หนัก* เนื่องจากอัตราส่วนดีฟอลต์ 8:1 ของ เธรดผู้ใช้ต่อเคอร์นัลเธรด การใช้งานอย่างหนักนี้พร้อมกับโอเวอร์เฮด ของการจัดตารางเวลาไลบรารีอาจส่งผลกระทบต่อประสิทธิภาพได้ ในทางตรงกันข้าม เมื่อมีเธรดผู้ใช้ อยู่มากนับพันรายการ แสดงว่าอาจมีโอเวอร์เฮดน้อยเกินกว่าจะจัดตารางเวลาเธรด เหล่านั้นในพื้นที่ว่างผู้ใช้ในไลบรารี แทนการจัดการเคอร์นัลเธรดนับพันรายการ คุณควรลองเปลี่ยนขอบเขตเสมอถ้าคุณพบปัญหาประสิทธิภาพเมื่อใช้ pthreads; ในหลายกรณี ขอบเขตแบบระบบสามารถให้ประสิทธิภาพที่ดีขึ้น

ถ้าแอ็พพลิเคชันกำลังรันอยู่บนระบบ SMP จากนั้นถ้าเธรดผู้ใช้ไม่สามารถ จัดหา mutex เธรดจะพยายาม spin สูงสุดถึง 40 ครั้ง อาจเกิดขึ้นได้ง่าย ในกรณีที่ mutex มีอยู่ภายในช่วงเวลาสั้นๆ ดังนั้นจึงอาจคุ้มค่าที่จะ spin ในช่วงเวลาที่นานขึ้น เมื่อคุณเพิ่ม CPUs เพิ่มเติม ถ้าประสิทธิภาพถดถอยลง โดยปกติแล้ว บ่งชี้ถึงปัญหาการล็อก คุณอาจต้องการเพิ่มเวลา spin โดยการตั้งค่าตัวแปรสภาพแวดล้อม **SPINLOOPTIME=***n* โดยที่ *n* คือจำนวนของ spins ไม่ใช่เรื่องผิดปกติที่จะตั้งค่าให้สูงถึงหลักพัน ทั้งนี้ขึ้นอยู่กับความเร็วของ CPUs และจำนวนของ CPUs หลังจากการนับ spin เสร็จสิ้นแล้ว เธรดสามารถไปพักรอให้ mutex พร้อมใช้งาน หรือเธรดสามารถออกใช้การเรียกระบบ **yield( )** และเพียงแต่ยอมแพ้ CPU แต่ยังคงอยู่ในสถานะที่ดำเนินการได้แทนที่จะ ไปพัก โดยค่าดีฟอลต์ เธรดจะไปพัก แต่โดยการตั้งค่าตัวแปรสภาพแวดล้อม **YIELDLOOPTIME** เป็นตัวเลข เธรดจะให้ผลตามจำนวนครั้งที่ตั้งค่าไว้ก่อนจะ ไปพัก ในแต่ละครั้งที่เธรดได้รับ CPU หลังจากให้ผล เธรดสามารถพยายามจัดหา mutex ได้

กระบวนการผู้ใช้แบบมัลติเธรด บางอย่างที่ใช้ระบบย่อย malloc อย่างหนักอาจมีประสิทธิภาพดีขึ้น โดยการส่งออกตัวแปรสภาพแวดล้อม **MALLOCMULTIHEAP=1** ก่อนที่จะเริ่มต้นแอ็พพลิเคชัน การพัฒนาประสิทธิภาพอาจเกิดขึ้นได้ มากเป็นพิเศษสำหรับโปรแกรม multithreaded C++ เนื่องจากโปรแกรมชนิดนี้ใช้ระบบย่อย malloc ในทุกครั้งที่เรียกตัวสร้าง หรือตัวทำลาย การพัฒนาประสิทธิภาพที่มีอยู่ใดๆ จะเห็นผลได้ชัดเจนที่สุดเมื่อกระบวนการผู้ใช้แบบมัลติเธรดกำลังรันอยู่บนระบบ SMP และโดยเฉพาะเมื่อใช้เธรดขอบเขตระบบ (อัตราส่วน M:N เป็น 1:1) อย่างไรก็ตาม ในบางกรณี การพัฒนาประสิทธิภาพยังอาจเห็นได้ชัด ภายใต้สภาพอื่น และบนระบบแบบตัวประมวลผลเดียวด้วย

ตัวแปรสภาพแวดล้อมเธรด

ภายในกรอบงาน `libpthreads.a` มีการนำเสนอชุดของปุ่มปรับซึ่งอาจส่งผลกระทบต่อ ประสิทธิภาพของแอ็พพลิเคชัน

ถ้าเป็นไปได้ ให้ใช้ front-end shell script เพื่อเรียกใช้โปรแกรมที่ดำเนินการ ได้แบบไบนารี Shell script ควรระบุค่าใหม่ที่คุณต้องการแทนที่ ค่าดีฟอลต์ระบบของตัวแปรสภาพแวดล้อม ที่อธิบายในส่วนที่ตามมา

AIXTHREAD_COND_DEBUG

ตัวแปร *AIXTHREAD_COND_DEBUG* เก็บรักษารายการของตัวแปรเงื่อนไขสำหรับการใช้โดย debugger ถ้าโปรแกรมมีตัวแปรเงื่อนไขที่ใช้งานอยู่จำนวนมาก และสร้างและทำลายตัวแปรเงื่อนไข บ่อยครั้ง อาจทำให้โอเวอร์เฮดสำหรับการเก็บรักษารายการของตัวแปรเงื่อนไข สูงขึ้น การตั้งค่าตัวแปรเป็น **OFF** จะปิดใช้งานรายการ การเปิดตัวแปรนี้ ทำให้การดีบักเธรดแอ็พพลิเคชันง่ายขึ้น แต่อาจส่งผลให้โอเวอร์เฮดเพิ่มขึ้น

AIXTHREAD_ENRUSG

ตัวแปร *AIXTHREAD_ENRUSG* เปิดใช้งานหรือปิดใช้งานการรวบรวมรีซอร์ส pthread การเปิดช่วยให้สามารถ รวบรวมรีซอร์สของ pthreads ทั้งหมดในกระบวนการ แต่จะ เพิ่มโอเวอร์เฮด

AIXTHREAD_GUARDPAGES=n

```
*      +----------------------+
*      | pthread attr         |
*      +----------------------+ <--- pthread->pt_attr
*      | pthread struct       |
*      +----------------------+ <--- pthread->pt_stk.st_limit
*      | pthread stack        |
*      |   |                  |
*      |   V                  |
*      +----------------------+ <--- pthread->pt_stk.st_base
*      | RED ZONE             |
*      +----------------------+ <--- pthread->pt_guardaddr
*      | pthread private data |
*      +----------------------+ <--- pthread->pt_data
```

RED ZONE บนภาพสาธิตนี้เรียกว่า *Guardpage*

pthread attr, pthread และ ctx แสดงถึงส่วน PTH_FIXED ของ หน่วยความจำที่จัดสรรสำหรับ pthread

ขนาดไบต์โดยประมาณในแผนผัง ด้านล่างอยู่ในวงเล็บสำหรับ 32 บิต สำหรับ 64 บิต ขนาดที่มี PTH_FIXED จะใหญ่ขึ้นเล็กน้อยและข้อมูลหลัก จะเป็น 8 K นอกจากนั้นเหมือนกัน

```
*      +----------------------+
*      | page alignment 2     |
*      | [8K-4K+PTH_FIXED-a1] |
*      +----------------------+
*      | pthread ctx [368]    |
*      +----------------------+<--- pthread->pt_attr
*      | pthread attr [112]   |
*      +----------------------+ <--- pthread->pt_attr
*      | pthread struct [960] |
*      +----------------------+ <--- pthread
```

```
*      | pthread stack         |         pthread->pt_stk.st_limit
*      |   |[96K+4K-PTH_FIXED] |
*      |   V                   |
*      +-----------------------+ <--- pthread->pt_stk.st_base
*      | RED ZONE [4K]         |
*      +-----------------------+ <--- pthread->pt_guardaddr
*      | pthread key data [4K] |
*      +-----------------------+ <--- pthread->pt_data
*      | page alignment 1 (a1) |
*      | [<4K]                 |
*      +-----------------------+
```

RED ZONE บนภาพสาธิตนี้เรียกว่า Guardpage

จำนวนเลขฐานสิบของ guardpages ที่ต้องการเพิ่มเข้ากับท้ายของสแต็ก pthread คือ *n* ซึ่งจะแทนที่ค่าแอ็ตทริบิวต์ที่ระบุเมื่อเวลาสร้าง pthread ถ้าแอ็พพลิเคชันระบุสแต็กของตัวเอง จะไม่มี guardpages ถูกสร้าง ค่าดีฟอลต์คือ 0 และ *n* ต้องเป็นค่า บวก

ขนาด guardpage ในหน่วยไบต์ถูกกำหนดโดยการคูณ *n* ด้วย PAGESIZE Pagesize เป็นขนาดที่ระบบ กำหนด

AIXTHREAD_DISCLAIM_GUARDPAGES

ตัวแปร *AIXTHREAD_DISCLAIM_GUARDPAGES* จะควบคุมว่าสแต็ก guardpages จะถูกจัดการเมื่อสแต็ก pthread ถูกสร้างขึ้นหรือไม่ ถ้า **AIXTHREAD_DISCLAIM_GUARDPAGES=ON** guardpages จะถูกจัดการ ถ้าสแต็ก pthread ไม่มีการตั้งค่า guardpages ใดๆ การตั้งค่าตัวแปร **AIXTHREAD_DISCLAIM_GUARDPAGES** จะไม่มีผล

AIXTHREAD_MNRATIO

ตัวแปร *AIXTHREAD_MNRATIO* ควบคุมปัจจัยการปรับสเกลของไลบรารี อัตราส่วนนี้ใช้เมื่อสร้างและยุติ pthreads อาจเป็นประโยชน์สำหรับแอ็พพลิเคชัน ที่มีเธรดจำนวนมาก อย่างไรก็ตาม ควรทดสอบอัตราส่วน 1:1 เสมอเนื่องจากเป็นอัตราส่วนที่ให้ประสิทธิภาพดีขึ้น

AIXTHREAD_MUTEX_DEBUG

ตัวแปร *AIXTHREAD_MUTEX_DEBUG* เก็บรักษารายการของ mutexes ที่ใช้งานอยู่สำหรับการใช้โดย debugger ถ้าโปรแกรมมี mutexes ที่ใช้งานอยู่จำนวนมาก และสร้างและทำลาย mutexes บ่อยครั้ง อาจทำให้โอเวอร์เฮดสำหรับการเก็บรักษารายการของ mutexes สูงขึ้น การตั้งค่าตัวแปรเป็น ON ทำให้การดีบักเธรดแอ็พพลิเคชันง่ายขึ้น แต่อาจส่งผลให้โอเวอร์เฮดเพิ่มขึ้น การปล่อยให้ตัวแปรมีการตั้งค่าเป็น OFF จะปิดใช้งานรายการ

AIXTHREAD_MUTEX_FAST

ถ้าโปรแกรมประสบกับการเสื่อมถอยด้านประสิทธิภาพการทำงาน เนื่องจากมีการช่วงชิง mutex อย่างหนัก ด้วยเหตุนี้ การตั้งค่าตัวแปรนี้เป็น ON จะบังคับให้ไลบรารี pthread ใช้กลไกการล็อก mutex ที่ถูกปรับแต่ง ซึ่งทำงานบน mutex ส่วนตัวของกระบวนการเท่านั้น ต้องเริ่มต้น mutexes ส่วนตัวของกระบวนการ เหล่านี้โดยใช้รูทีน pthread_mutex_init และต้องทำลาย โดยใช้รูทีน pthread_mutex_destroy การปล่อยให้ตัวแปรมีการตั้งค่า เป็น OFF จะบังคับให้ pthread ไลบรารีใช้กลไกการล็อก mutex ดีฟอลต์

AIXTHREAD_READ_GUARDPAGES

ตัวแปร *AIXTHREAD_READ_GUARDPAGES* จะเปิดใช้งานหรือปิดใช้งานการเข้าถึงเพื่ออ่าน guardpages ที่ถูกเพิ่มเข้ากับท้ายของสแต็ก pthread สำหรับข้อมูลเพิ่มเติมเกี่ยวกับ guardpages ที่ถูกสร้างโดย pthread โปรดดูที่ "AIXTHREAD_GUARDPAGES=n" ในหน้า 78

AIXTHREAD_RWLOCK_DEBUG

ตัวแปร *AIXTHREAD_RWLOCK_DEBUG* เก็บรักษารายการของล็อกการอ่าน-การบันทึกสำหรับการใช้โดย debugger ถ้าโปรแกรมมีล็อก การอ่าน-การบันทึกที่ใช้งานอยู่จำนวนมาก และสร้างและทำลายล็อกการอ่าน-การ บันทึกบ่อยครั้ง อาจทำให้โอเวอร์เฮดสำหรับการเก็บรักษารายการของล็อกการอ่าน- การบันทึกสูงขึ้น การตั้งค่าตัวแปรเป็น OFF จะปิดใช้งานรายการ

AIXTHREAD_SUSPENDIBLE={ON|OFF}

การตั้งค่า ตัวแปร *AIXTHREAD_SUSPENDIBLE* เป็น ON ป้องกัน deadlock ในแอ็พพลิเคชันที่ใช้รูทีนต่อไปนี้พร้อมกับรูทีน **pthread_suspend_np** หรือรูทีน **pthread_suspend_others_np**:

- pthread_getrusage_np
- pthread_cancel
- pthread_detach
- pthread_join
- pthread_getunique_np
- pthread_join_np
- pthread_setschedparam
- pthread_getschedparam
- pthread_kill

ตัวแปรนี้ส่งผลกระทบต่อประสิทธิภาพเล็กน้อย

AIXTHREAD_SCOPE={P|S}

อ็อพชัน **P** หมายถึง ขอบเขตการยืนยันที่ใช้ *กระบวนการ* (M:N) ขณะที่อ็อพชัน **S** หมายถึง ขอบเขตการยืนยันที่ใช้ *ระบบ* (1:1). คุณต้องระบุอ็อพชันใดอ็อพชันหนึ่งเหล่านี้ ค่าดีฟอลต์ คือ **P**

การใช้ตัวแปรสภาพแวดล้อม *AIXTHREAD_SCOPE* มีผลกระทบต่อเธรด ที่สร้างขึ้นด้วยดีฟอลต์แอ็ตทริบิวต์เท่านั้น ดีฟอลต์แอ็ตทริบิวต์ จะนำมาใช้เมื่อพารามิเตอร์ *attr* ของ รูทีนย่อย **pthread_create( )** เป็น NULL

ถ้าเธรดผู้ใช้ถูกสร้างขึ้นด้วยขอบเขตของระบบ เธรดจะถูกเชื่อมกับเธรดเคอร์เนล และถูกกำหนดเวลาโดยเคอร์เนล เคอร์นัลเธรดดังกล่าวไม่มีการแบ่งใช้ กับเธรดผู้ใช้อื่นใด

ถ้าเธรดผู้ใช้ถูกสร้างขึ้นด้วยขอบเขตของกระบวนการ อาจมีการใช้ตัวกำหนดเวลาของผู้ใช้ ซึ่งหมายความว่า:

- ไม่มีเคอร์นัลเธรดเฉพาะ
- พักในโหมดผู้ใช้
- วางบนรันคิวผู้ใช้ในขณะกำลังรอตัวประมวลผล

- ขึ้นอยู่กับส่วนเวลาโดยตัวจัดตารางเวลาผู้ใช้

การทดสอบแสดงว่าแอ็พพลิเคชันบางรายการสามารถทำงานได้ดีขึ้นด้วย โมเดล 1:1

AIXTHREAD_SLPRATIO

ตัวแปรการปรับเธรด *AIXTHREAD_SLPRATIO* ควบคุม จำนวนของเคอร์นัลเธรดที่ควรจะเก็บสำรองไว้สำหรับเธรด ที่กำลังพักอยู่ โดยทั่วไป การสนับสนุน pthreads ที่กำลังพักอยู่ไม่ต้องการเคอร์นัลเธรดมากนัก เนื่องจากโดยทั่วไปแล้ว จะมีการตื่นทีละหนึ่งรายการ ลักษณะเช่นนี้ช่วยประหยัดรีซอร์สเคอร์นัล

AIXTHREAD_STK=n

ตัวแปรการปรับเธรด *AIXTHREAD_STK=n* ควบคุม จำนวนทศนิยมของไบต์ที่ควรจะจัดสรรสำหรับแต่ละ เธรด ค่านี้อาจถูกแทนที่โดย **pthread_attr_setstacksize**

AIXTHREAD_AFFINITY={default|strict|first-touch}

AIXTHREAD_AFFINITY ควบคุม การวางตำแหน่งของโครงสร้าง pthread, สแต็ก, และหน่วยเก็บ thread-local บนระบบที่เปิดใช้งานความสัมพันธ์แบบเสริมศักยภาพ

- อ็อพชันดีฟอลต์จะไม่พยายามวางตำแหน่งข้อมูลนี้ในแบบพิเศษ เพื่อให้ความเกิดความสมดุลในพื้นที่หน่วยความจำซึ่งใช้โดยกระบวนการตามที่ กำหนดโดยค่าติดตั้งระบบ
- อ็อพชัน strict จะวางข้อมูลนี้ในหน่วยความจำโลคัล ของ pthread เสมอซึ่งอาจส่งผลกระทบต่อประสิทธิภาพการทำงานบางอย่างใน ระหว่างการสร้าง pthread เนื่องจากข้อมูลที่มีอยู่ถูกย้ายจากพื้นที่หน่วยความจำหนึ่ง ไปยังอีกพื้นที่หนึ่ง อย่างไรก็ตาม อ็อพชันนี้อาจช่วยให้ประสิทธิภาพรันไทม์พัฒนาขึ้น
- อ็อพชัน first touch คล้ายกันในการวางข้อมูลไว้ในหน่วยความจำ โลคัลของ pthread แต่อ็อพชันนี้จะไม่พยายามย้ายข้อมูลใดๆ ภายใน หน่วยความจำ เธรดต้องการหน้า in-memory สำหรับข้อมูลนี้ (รวมถึง paging ในหน่วยความจำจากพื้นที่ paging) และจะมีการวาง แบบโลคัล อ็อพชันนี้ช่วยให้เกิดความสมดุลระหว่างเวลาสตาร์ทอัพและประสิทธิภาพ รันไทม์

AIXTHREAD_PREALLOC=n

ตัวแปร *AIXTHREAD_PREALLOC* กำหนด จำนวนไบต์ที่จะจัดสรรล่วงหน้าและทำให้ว่างในระหว่างการสร้าง เธรด แอ็พพลิเคชันหลายเธรดบางรายการอาจได้รับประโยชน์จากตัวแปรนี้ โดยการหลีกเลี่ยงการเรียกไปยัง sbrk() จากหลายเธรดพร้อมกัน

ค่าดีฟอลต์คือ 0 และ *n* ต้องเป็นค่า บวก

AIXTHREAD_HRT

ตัวแปร *AIXTHREAD_HRT=true* อนุญาตไทม์เอาต์ความละเอียดสูงสำหรับ `pthreads` ของแอ็พพลิเคชัน คุณต้องมีสิทธิ `root` หรือความสามารถ `CAP_NUMA_ATTACH` เพื่อเปิดใช้งานไทม์เอาต์ความละเอียดสูง ตัวแปรสภาวะแวดล้อมนี้ถูก ละเว้นถ้าคุณไม่มีสิทธิหรือความสามารถที่จำเป็น

MALLOCBUCKETS

Malloc buckets นำเสนอส่วนขยายบน buckets ที่เลือกกำหนดได้ของตัวจัดสรรดีฟอลต์ มีไว้เพื่อช่วยปรับปรุง ประสิทธิภาพ malloc สำหรับแอ็พพลิเคชันที่ออกใช้การร้องขอการจัดสรรขนาด เล็กจำนวนมาก เมื่อเปิดใช้งาน malloc buckets การร้องขอการจัดสรรที่อยู่ภายใน ช่วงที่กำหนดไว้แล้วของขนาดบล็อกจะมีการประมวลผลโดย malloc buckets การร้องขออื่นทั้งหมดจะมีการประมวลผลในลักษณะปกติ โดยตัวจัดสรรดีฟอลต์

Malloc buckets ไม่ได้เปิดใช้งานโดย ค่าดีฟอลต์ มีการเปิดใช้งานและตั้งค่าคอนฟิกก่อนหน้าการเริ่มต้นกระบวนการ โดย การตั้งค่าตัวแปรสภาพแวดล้อม *MALLOCTYPE* และ *MALLOCBUCKETS*

หากต้องการข้อมูลเพิ่มเติมเกี่ยวกับ malloc buckets ให้ดู *General Programming Concepts: Writing and Debugging Programs*

MALLOCMULTIHEAP={considersize,heaps:n}

มีความ ต้องการหลาย heaps เพื่อให้เธรดแอ็พพลิเคชันสามารถมีเธรดมากกว่าหนึ่งรายการ เพื่อออกใช้ **malloc( )**, **free( )**, และการเรียกรูทีนย่อย **realloc( )** ถ้ามี heap เดียว เธรดทั้งหมดที่พยายามทำการเรียก **malloc( )**, **free( )**, หรือ **realloc( )** จะถูกจัดลำดับ (นั่นคือมีการเรียกเพียงหนึ่งรายการเท่านั้น ในแต่ละครั้ง) ซึ่งส่งผลกระทบอย่างรุนแรงบนเครื่องแบบหลายตัวประมวลผล ถ้ามี heaps หลายรายการ แต่ละเธรดจะเรียกใช้ heap ของตนเอง ถ้า heaps ทั้งหมด ถูกใช้งานอยู่ เธรดใหม่ที่พยายามทำการเรียกจะต้องรอจนกว่า มี heaps พร้อมใช้งานหนึ่งรายการขึ้นไป การจัดลำดับยังคงมีอยู่ แต่โอกาสของการเกิดและผลกระทบเมื่อเกิดขึ้นจะ ลดลงอย่างมาก

การล็อก thread-safe มีการเปลี่ยนแปลง เพื่อจัดการแนวทางนี้ แต่ละ heap มีล็อกของตนเอง และรูทีนการล็อก "อัจฉริยะ" จะเลือก heap ที่จะพยายามป้องกัน การจัดลำดับ ถ้ามีการตั้งค่าอ็อพชัน **considersize** ในตัวแปรสภาพแวดล้อม *MALLOCMULTIHEAP* การเลือกจะยังพยายามเลือก heap ที่มีอยู่ใดๆ ซึ่งมีพื้นที่ว่างมากพอที่จะ จัดการกับการร้องขอ แทนที่จะเลือก heap ที่ไม่ได้ล็อกถัดไปเพียง อย่างเดียว

สามารถระบุได้มากกว่าหนึ่งอ็อพชัน (และในลำดับใดก็ได้) โดยคั่นด้วยเครื่องหมายจุลภาค ตัวอย่าง:

```
MALLOCMULTIHEAP=considersize,heaps:3
```

อ็อพชัน มีดังนี้:

considersize
: อ็อพชันนี้ใช้ขั้นตอนวิธีการเลือก heap ที่แตกต่างอื่น ซึ่งพยายามลดขนาดชุด ทำงานของกระบวนการให้เล็กที่สุด ค่าดีฟอลต์ไม่ใช้อ็อพชันนี้ และใช้ขั้นตอนวิธีที่เร็วกว่า

heaps:*n*
: ใช้อ็อพชันนี้เพื่อเปลี่ยนจำนวนของ heaps ช่วงที่ถูกต้องสำหรับ *n* คือ 1 ถึง 32 ถ้าคุณตั้งค่า *n* เป็นตัวเลขนอกช่วงนี้ (นั่นคือ ถ้า *n*<=0 หรือ *n*>32) *n* จะถูกตั้งค่า เป็น 32

ค่าดีฟอลต์สำหรับ *MALLOCMULTIHEAP* คือ NOT SET (ใช้ heap แรกเท่านั้น) ถ้ามีการตั้งค่าตัวแปรสภาพแวดล้อม *MALLOCMULTIHEAP* (ตัวอย่างเช่น `MALLOCMULTIHEAP=1`) เธรดแอ็พพลิเคชัน จะสามารถใช้ทั้ง 32 heaps ได้ การตั้งค่า `MALLOCMULTIHEAP=heaps:`*n* จะ จำกัดจำนวน heaps เป็น *n* แทนจำนวน 32 heaps

หากต้องการข้อมูลเพิ่มเติม ให้ดูส่วน Malloc Multiheap ใน *General Programming Concepts: Writing and Debugging Programs*

SPINLOOPTIME=n

ตัวแปร *SPINLOOPTIME* ควบคุม จำนวนครั้งที่ระบบจะพยายามเรียกใช้ mutex ที่ยุ่งหรือ spin lock โดยไม่ทำการดำเนินการที่สอง เช่น การเรียกเคอร์เนลเพื่อให้ กระบวนการ การควบคุมนี้มีไว้สำหรับระบบ MP ซึ่งหวังว่าล็อกที่ใช้อยู่โดย pthread อื่นที่กำลังรันอยู่ จะถูกรีลีส พารามิเตอร์จะทำงานภายใน libpthreads (เธรดผู้ใช้) เท่านั้น ถ้าโดยปกติแล้ว ล็อกมีอยู่ในช่วงเวลาสั้นๆ คุณอาจต้องการเพิ่มเวลา spin โดยตั้งค่าตัวแปรสภาพแวดล้อม นี้ จำนวนครั้งในการลองล็อกที่ยุ่งใหม่ก่อนให้ pthread อื่นคือ *n* ค่าดีฟอลต์คือ 40 และ *n* ต้องเป็นค่าบวก

เคอร์นัลพารามิเตอร์ *MAXSPIN* มีผลต่อ spinning ในเคอร์นัลล็อกรูทีน (โปรดดู "การใช้คำสั่ง schedo เพื่อแก้ไขพารามิเตอร์ MAXSPIN" ในหน้า 86)

YIELDLOOPTIME=n

ตัวแปร *YIELDLOOPTIME* ควบคุมจำนวนครั้งที่ระบบให้ตัวประมวลผล เมื่อพยายามจัดหา mutex ที่ยุ่งหรือ spin lock ก่อนพักไว้บนล็อก อย่างแท้จริง ตัวประมวลผลมีการให้ไปยังเคอร์นัลเธรดอื่น โดยสมมุติว่ามีรายการที่ดำเนินการได้อื่นซึ่งมีระดับความสำคัญสูงเพียงพอ ตัวแปรนี้มีประโยชน์ในแอ็พพลิเคชันที่ซับซ้อน ที่มีการใช้หลายล็อก จำนวนครั้งในการให้ตัวประมวลผล ก่อนที่จะบล็อกล็อกที่ยุ่งคือ *n* ค่าดีฟอลต์คือ 0 และ *n* ต้องเป็นค่า บวก

ตัวแปรสำหรับขอบเขตการช่วงชิงทั้งกระบวนการ

ตัวแปรสภาพแวดล้อมต่อไปนี้มีผลกระทบต่อการจัดตารางเวลาของ pthreads ที่สร้างขึ้นด้วยขอบเขตการช่วงชิงทั้งกระบวนการ

AIXTHREAD_MNRATIO=*p*:*k*

โดยที่ *k* คือจำนวนของเคอร์นัลเธรดที่ควรจะใช้เพื่อจัดการ *p* pthreads ที่รันได้ ตัวแปรสภาพแวดล้อมนี้ควบคุม ปัจจัยการปรับสเกลของไลบรารี อัตราส่วนนี้ใช้เมื่อสร้างและยุติ pthreads ตัวแปรใช้ได้กับขอบเขตทั้งกระบวนการเท่านั้น สำหรับขอบเขตทั้งระบบ ตัวแปรสภาพแวดล้อมนี้จะถูกเพิกเฉย ค่าติดตั้งดีฟอลต์คือ 8:1

AIXTHREAD_SLPRATIO=*k*:*p*

โดยที่ *k* คือจำนวนของเคอร์นัลเธรดที่ควรจะเก็บสำรองไว้สำหรับ *p* pthreads ที่กำลังพักอยู่ อัตราส่วนการพักคือจำนวนของเคอร์นัลเธรด ที่เก็บแยกไว้เพื่อสนับสนุน pthreads ที่กำลังพักอยู่โดยทั่วไป การสนับสนุน pthreads ที่กำลังพักอยู่ไม่ต้องการเคอร์นัลเธรดมากนัก เนื่องจากโดยทั่วไปแล้ว จะมีการตื่น ทีละหนึ่งรายการ ลักษณะเช่นนี้ช่วยประหยัดรีซอร์สเคอร์นัล สามารถระบุค่าเลขจำนวนเต็มบวก ใดๆ สำหรับ *p* และ *k* ถ้า *k*>*p* จะถือว่าอัตราส่วนเป็น 1:1 ค่าดีฟอลต์คือ 1:12

AIXTHREAD_MINKTHREADS=*n*

โดยที่ *n* คือจำนวนต่ำสุดของเคอร์นัลเธรดที่ควรจะ ใช้ ตัวจัดตารางเวลาไลบรารี จะไม่เรียกร้องคืนเคอร์นัลเธรดต่ำกว่าตัวเลขนี้ เคอร์นัลเธรดอาจถูกเรียกร้องคืนในเวลาใดก็ได้ในแบบเสมือน โดยทั่วไป มีความต้องการเคอร์นัลเธรดคืนเนื่องจากการยุติ pthread ค่าดีฟอลต์คือ 8

อ็อพชันการดีบักเธรด

ไลบรารี pthreads เก็บรักษารายการของ mutexes ที่ใช้งานอยู่ ตัวแปร เงื่อนไข และล็อกการอ่าน–การบันทึกสำหรับใช้โดย debugger

เมื่อเริ่มต้นล็อก ล็อกจะถูกเพิ่มลงในรายการ โดยสมมุติว่าล็อกยังไม่ได้มีอยู่ บนรายการ รายการมีการเก็บไว้เป็นรายการที่ลิงก์ ดังนั้นการกำหนดว่าลิงก์ใหม่ ไม่ได้มีอยู่แล้วในรายการจึงส่งผลกระทบต่อประสิทธิภาพเมื่อรายการมีขนาด ใหญ่ขึ้น ปัญหารุน

แรงขึ้นจากข้อเท็จจริงที่ว่ารายการมีการป้องกัน โดยล็อก (dbx__mutexes) ซึ่งใช้กับการค้นหา รายการ ในกรณีนี้ การเรียกอื่นไปยังรูทีนย่อย **pthread_mutex_init( )** จะถูกพักไว้ในขณะที่ทำการค้นหา

ถ้าตัวแปรสภาพแวดล้อมต่อไปนี้มีการตั้งค่าเป็น OFF ซึ่งเป็นค่าดีฟอลต์ รายการดีบักที่เหมาะสมจะถูกปิดใช้งานโดยสมบูรณ์ นั่นหมายความว่า คำสั่ง **dbx** (หรือ debugger ที่ใช้ pthread ดีบักไลบรารี) จะไม่แสดงอ็อบเจ็กต์ใดๆ

- **AIXTHREAD_MUTEX_DEBUG**
- **AIXTHREAD_COND_DEBUG**
- **AIXTHREAD_RWLOCK_DEBUG**

เมื่อต้องการตั้งค่าตัวแปรสภาพแวดล้อมเหล่านี้เป็น ON ให้ใช้คำสั่งต่อไปนี้:

```
# export variable_name=ON
```

เครื่องมือ SMP

เครื่องมือประสิทธิภาพทั้งหมดของระบบปฏิบัติการสนับสนุนเครื่อง SMP

เครื่องมือประสิทธิภาพบางอย่างนำเสนอสถิติการใช้ประโยชน์ตัวประมวลผล แต่ละตัว เครื่องมือประสิทธิภาพอื่นหาค่าเฉลี่ยของสถิติการใช้ประโยชน์ สำหรับตัวประมวลผลทั้งหมดและแสดงเฉพาะค่าเฉลี่ยเท่านั้น

ส่วนนี้อธิบายเครื่องมือที่ได้รับการสนับสนุนบนระบบ SMP เท่านั้น สำหรับรายละเอียดเกี่ยวกับเครื่องมือจัดการประสิทธิภาพอื่นทั้งหมด โปรดดูที่ส่วนที่เหมาะสม

คำสั่ง bindprocessor

ใช้คำสั่ง **bindprocessor** เพื่อโยงหรือยกเลิกการโยงเคอร์เนลเธรด ของการประมวลผลกับตัวประมวลผล

สิทธิการใช้งานแบบรากเป็นสิ่งที่จำเป็นเพื่อโยงหรือยกเลิกการโยงเธรด ในการประมวลผลที่คุณไม่ได้เป็นเจ้าของ

หมายเหตุ: คำสั่ง **bindprocessor** จะมีความหมายสำหรับระบบแบบมัลติโพรเซสเซอร์ แม้ว่าจะทำงานกับระบบยูนิโพรเซสเซอร์ การโยงจะไม่มีผลกระทบกับระบบใดๆ

หากต้องการเคียวรี่ตัวประมวลผลที่พร้อมใช้งาน ให้รันคำสั่งต่อไปนี้:

```
# bindprocessor -q
The available processors are:  0 1 2 3
```

เอาต์พุตจะแสดงจำนวนตัวประมวลผลแบบโลจิคัลสำหรับตัวประมวลผลที่พร้อมใช้งาน ซึ่งจะถูกใช้กับคำสั่ง **bindprocessor** ซึ่งสามารถมองเห็นได้

หากต้องการโยงการประมวลผลที่มี PID คือ 14596 กับตัวประมวลผล 1 ให้รันคำสั่งต่อไปนี้:

```
# bindprocessor 14596 1
```

ไม่มีข้อความส่งกลับที่กำหนดไว้หากคำสั่งดำเนินการเป็นผลสำเร็จ หากต้องการตรวจสอบว่า การประมวลผลโยงหรือยกเลิกการโยงกับตัวประมวลผล ให้ใช้คำสั่ง **ps –mo THREAD** ตามที่ได้แสดงไว้ใน “การใช้คำสั่ง ps” ในหน้า 123:

```
# ps -mo THREAD
USER   PID  PPID    TID ST  CP PRI SC    WCHAN        F     TT BND COMMAND
root  3292  7130      - A    1  60  1        -   240001  pts/0   - -ksh
```

```
   -     -     -  14309 S    1  60  1        -      400      -   - -
root 14596  3292      - A   73 100  1        -   200001  pts/0   1 /tmp/cpubound
   -     -     -  15629 R   73 100  1        -        0      -   1 -
root 15606  3292      - A   74 101  1        -   200001  pts/0   - /tmp/cpubound
   -     -     -  16895 R   74 101  1        -        0      -   - -
root 16634  3292      - A   73 100  1        -   200001  pts/0   - /tmp/cpubound
   -     -     -  15107 R   73 100  1        -        0      -   - -
root 18048  3292      - A   14  67  1        -   200001  pts/0   - ps -mo THREAD
   -     -     -  17801 R   14  67  1        -        0      -   - -
```

คอลัมน์ *BND* จะแสดงจำนวนของตัวประมวลผลที่การประมวลผลถูกโยงไว้ หรือแสดงเส้นประ (-) หากการประมวลผลไม่ได้โยงไว้ทั้งหมด

หากต้องการยกเลิกการโยงการประมวลผลที่มี PID คือ14596 ให้ใช้คำสั่งต่อไปนี้:

```
# bindprocessor -u 14596
# ps -mo THREAD
USER   PID  PPID    TID ST  CP PRI SC    WCHAN        F     TT BND COMMAND
root  3292  7130      - A    2  61  1        -   240001  pts/0   - -ksh
   -     -     -  14309 S    2  61  1        -      400      -   - -
root 14596  3292      - A  120 124  1        -   200001  pts/0   - /tmp/cpubound
   -     -     -  15629 R  120 124  1        -        0      -   - -
root 15606  3292      - A  120 124  1        -   200001  pts/0   - /tmp/cpubound
   -     -     -  16895 R  120 124  1        -        0      -   - -
root 16634  3292      - A  120 124  0        -   200001  pts/0   - /tmp/cpubound
   -     -     -  15107 R  120 124  0        -        0      -   - -
root 18052  3292      - A   12  66  1        -   200001  pts/0   - ps -mo THREAD
   -     -     -  17805 R   12  66  1        -        0      -   - -
```

เมื่อคำสั่ง **bindprocessor** ถูกนำมาใช้บนการประมวลผล เธรดทั้งหมดจะถูกโยงไว้กับตัวประมวลผลหนึ่งตัว และจะยกเลิกการโยงจากตัวประมวลผลก่อนหน้านี้ การยกเลิกการโยงการประมวลผลจะยกเลิกการโยงเธรดทั้งหมด เช่นกัน คุณไม่สามารถโยงหรือยกเลิกการโยงเธรดแต่ละตัวโดยใช้คำสั่ง **bindprocessor**

อย่างไรก็ตาม ภายในโปรแกรม คุณสามารถใช้การเรียกฟังก์ชัน **bindprocessor( )** เพื่อโยงเธรดแต่ละตัวไว้ ถ้าฟังก์ชัน **bindprocessor( )** ถูกใช้ภายในชิ้นส่วนของโค้ดที่โยงเธรดกับตัวประมวลผลไว้ เธรดเหล่านั้นจะยังคงอยู่กับตัวประมวลผลเหล่านั้น และไม่สามารถยกเลิกการโยงได้ ถ้าคำสั่ง **bindprocessor** ถูกใช้อยู่บนการประมวลผลนั้น เธรดทั้งหมดจะถูกโยงไว้กับตัวประมวลผลหนึ่งตัว และยกเลิกการโยงจากตัวประมวลผลก่อนหน้านี้ตามลำดับ การยกเลิกการโยงของตัวประมวลผลทั้งหมดจะยังคงยกเลิกเธรดทั้งหมดด้วยเช่นกัน

การประมวลผลไม่สามารถโยงไว้ได้จนกระทั่งเริ่มต้นแล้ว นั่นคือ การประมวลผลจะต้องมีอยู่เพื่อโยงไว้ เมื่อการประมวลผลไม่มีอยู่ ข้อผิดพลาดต่อไปนี้จะแสดงขึ้น :

```
# bindprocessor 7359 1
1730-002: Process 7359 does not match an existing process
```

เมื่อตัวประมวลผลไม่มีอยู่ ข้อผิดพลาดต่อไปนี้จะแสดงขึ้น:

```
# bindprocessor 7358 4
1730-001: Processor 4 is not available
```

หมายเหตุ: ห้ามใช้คำสั่ง **bindprocessor** บนการประมวลผล **kproc** ที่รออยู่

ข้อควรพิจารณาในการโยง:

มีปัญหาหลายอย่างที่ต้องพิจารณาก่อนคุณใช้ การโยงกระบวนการ

การโยงอาจมีประโยชน์สำหรับโปรแกรมที่ใช้ CPU สูงซึ่งพบการอินเตอร์รัปต์เพียงเล็กน้อย ในบางครั้งอาจเป็นตัวนับผลผลิตสำหรับโปรแกรม ปกติ เนื่องจากอาจหน่วงเวลาการจัดส่งของเธรดอีกครั้งหลัง I/O จนกว่า ตัวประมวลผลที่โยงกับเธรดไว้จะพร้อมใช้งาน ถ้า เธรดถูกบล็อกไว้ระหว่างการดำเนินการ I/O บริบทการประมวลผลส่วนใหญ่ ไม่น่าจะยังคงอยู่ในแคชของตัวประมวลผล ซึ่งโยงไว้ เธรดอาจให้บริการให้ดีกว่าถ้ามีการจัดส่ง ไปยังตัวประมวลผลที่พร้อมใช้งานถัดไป

การโยงไม่ได้ป้องกันการประมวลผลอื่นๆ จากการจัดส่งบนตัวประมวลผล ซึ่งคุณโยงไว้กับการประมวลผลของคุณ การโยงจะแตกต่างจากการแบ่งพาร์ติชัน การใช้ `rsets` หรือ `rsets` เฉพาะช่วยให้ สามารถกำหนดชุดของตัวประมวลผลโลจิคัลสำหรับเวิร์กโหลดที่ระบุโดยเฉพาะได้ ดังนั้น กระบวนการที่มีลำดับความสำคัญสูงกว่า อาจถูกจัดส่งบนตัวประมวลผลซึ่งคุณโยงไว้กับกระบวนการของคุณ ในกรณีนี้ กระบวนการของคุณจะไม่มีการจัดส่ง บนกระบวนการอื่น ดังนั้น ประสิทธิภาพของกระบวนการที่โยง จึงไม่เพิ่มขึ้น อย่างไรก็ตาม อาจได้ผลลัพธ์ที่ดีขึ้นถ้าคุณ เพิ่มลำดับความสำคัญของกระบวนการที่โยงไว้

ถ้าคุณโยงกระบวนการบนระบบที่โหลดอย่างหนัก คุณอาจทำให้ประสิทธิภาพลดลง เนื่องจากเมื่อตัวประมวลผลไม่ได้ทำงาน กระบวนการจะไม่สามารถรันได้บน ตัวประมวลผลที่ไม่ทำงาน หากไม่ใช่ตัวประมวลผลที่กระบวนการ โยงอยู่

ถ้ากระบวนการเป็นแบบมัลติเธรด การโยงการประมวลผลจะโยงเธรดทั้งหมด กับตัวประมวลผลเดียวกัน ดังนั้น กระบวนการไม่ได้ใช้ประโยชน์ ของการทำงานแบบหลายกระบวนการ และไม่ได้ปรับปรุงประสิทธิภาพ

หมายเหตุ: ใช้การโยงกระบวนการด้วยความระมัดระวัง เนื่องจากการโยงจะหยุดดุลยภาพในการโหลดตามธรรมชาติ ที่จัดเตรียมไว้โดย AIX และอาจทำให้ประสิทธิภาพโดยรวมของระบบลดลงได้ ถ้าเวิร์กโหลด ของระบบเปลี่ยนจากการโยงแรกเริ่มที่มีการมอนิเตอร์ ประสิทธิภาพของระบบอาจได้รับผลกระทบ ถ้าคุณใช้คำสั่ง **bindprocessor** ให้คอยมอนิเตอร์เครื่องเป็นประจำ เนื่องจากสภาวะแวดล้อมอาจเปลี่ยนไป ทำให้กระบวนการที่โยงไว้ส่งผลเชิงลบต่อประสิทธิภาพการทำงานของระบบ

การใช้คำสั่ง schedo เพื่อแก้ไขพารามิเตอร์ MAXSPIN

ถ้าเธรดต้องการจัดหาล็อกเมื่อเธรดอีกรายการหนึ่งเป็นเจ้าของล็อกอยู่ใน ปัจจุบันและกำลังรันบน CPU อื่น เธรดที่ต้องการล็อกจะ spin บน CPU จนกว่าเธรดเจ้าของล็อกจะรีลีสล็อกถึงตามค่าที่ระบุโดย พารามิเตอร์ที่ปรับได้ซึ่งเรียกว่า *MAXSPIN*

ค่าดีฟอลต์ของ *MAXSPIN* คือ 0x4000 (16384) สำหรับระบบ SMP และ 1 บนระบบ UP ถ้าคุณสังเกตเห็นเวลา idle หรือเวลารอ I/O มากขึ้นบนระบบที่ไม่เคยแสดงผลเช่นนี้มาก่อน นั่นอาจเป็นเพราะเธรดกำลัง จะพักบ่อยเกินไป ถ้าลักษณะนี้ทำให้เกิดปัญหาประสิทธิภาพ ให้ปรับ *MAXSPIN* เป็นค่าที่สูงขึ้น หรือตั้งค่าเป็น -1 ซึ่งหมายความว่า spin มากถึง 0xFFFFFFFF ครั้ง

เมื่อต้องการแก้ไข จำนวนครั้งที่จะ spin ก่อนจะไปพัก ให้ใช้อ็อพชัน **maxspin** ของคำสั่ง **schedo** เมื่อต้องการลดการใช้ CPU ที่อาจเกิดจาก spins ที่มากเกินไป ให้ลดค่าของ *MAXSPIN* เป็นดังนี้:

```
# schedo -o maxspin=8192
```

คุณอาจเห็น การเพิ่มขึ้นในการสลับบริบท ถ้าการสลับบริบทกลายเป็นปัญหาคอขวด ให้เพิ่ม *MAXSPIN*

เพื่อเปลี่ยนค่า คุณต้องเป็นผู้ใช้ราก

การวางแผนประสิทธิภาพและการดำเนินการ

โปรแกรมที่ทำงานไม่ถูกต้องไม่สามารถใช้งานได้ ทุกโปรแกรม ต้องเป็นที่พึงพอใจของกลุ่มผู้ใช้ ซึ่งบางครั้งเป็นกลุ่มขนาดใหญ่และมีหลากหลายประเภท ถ้า ประสิทธิภาพของโปรแกรมไม่สามารถยอมรับได้โดยสิ้นเชิงสำหรับผู้ใช้จำนวนมาก โปรแกรมนั้นจะไม่สามารถใช้งานได้ โปรแกรมที่ไม่ได้ใช้งานอยู่ไม่ได้ดำเนินการ ฟังก์ชันที่มุ่งหวังไว้

สถานการณ์นี้เป็นจริงสำหรับซอฟต์แวร์แพ็กเกจที่ได้รับไลเซนส์และแอ็พพลิเคชัน ที่ผู้ใช้เขียนขึ้น ผู้พัฒนาซอฟต์แวร์แพ็กเกจส่วนใหญ่ตะหนักถึงผลกระทบของ ประสิทธิภาพที่ไม่ดีและทุ่มเทความพยายามเพื่อทำให้โปรแกรมของพวกเขารันได้ เร็วที่สุดเท่าที่จะเป็นไปได้ โชคไม่ดีนัก พวกเขาไม่สามารถคาดการณ์สภาพแวดล้อมและการใช้งาน ทั้งหมดที่โปรแกรมของพวกเขาจะพบ ความรับผิดชอบขั้นสุดท้ายสำหรับประสิทธิภาพที่ยอมรับได้ ตกอยู่ที่บุคคลที่เลือกหรือเขียนแผนและติดตั้งซอฟต์แวร์แพ็กเกจ

ส่วนนี้จะอธิบายขั้นตอนที่โปรแกรมเมอร์หรือผู้ดูแลระบบสามารถมันใจว่าโปรแกรมที่เขียนขึ้นใหม่หรือที่ซื้อใหม่มีประสิทธิภาพที่ยอมรับได้ (ในทุกที่ที่คำว่า *โปรแกรมเมอร์* แสดงอยู่เพียงคำเดียว ให้หมายความรวมถึง ผู้ดูแลระบบและบุคคลอื่นใดที่รับผิดชอบต่อความสำเร็จสุดท้าย ของโปรแกรม)

เพื่อให้ได้ประสิทธิภาพที่ยอมรับได้ในโปรแกรม ให้ระบุและกำหนดระดับการยอม รับได้ตั้งแต่เริ่มต้นโครงการ และต้องให้ความสำคัญกับการประเมินและรีซอร์ส ที่ต้องใช้เพื่อให้บรรลุผลอยู่เสมอ แม้ว่าวิธีการนี้ฟังดูเป็นเรื่องมูลฐาน แต่โครงการเขียนโปรแกรม บางโครงการกลับตั้งใจที่จะละเลย การเขียนโปรแกรมใช้นโยบายที่อาจจะอธิบายคร่าวๆ เกี่ยวกับ การออกแบบ โค้ด ดีบัก และอาจจะเอกสาร และถ้าเรามีเวลา ต้องแก้ไขประสิทธิภาพ

วิธีเดียวที่สามารถคาดการณ์ได้ว่าโปรแกรมจะทำงานอย่างถูกต้องเมื่อใช้งานจริง ไม่เพียงแต่ในทางตรรกะคือ การรวมข้อควรพิจารณาเกี่ยวกับประสิทธิภาพไว้ใน กระบวนการวางแผนและการพัฒนาซอฟต์แวร์ ในบางครั้ง การวางแผนขั้นสูงนับเป็นสิ่งที่สำคัญอย่างยิ่ง เมื่อกำลังติดตั้งซอฟต์แวร์ที่มีอยู่ เนื่องจากผู้ติดตั้งมีอิสระน้อยกว่า ผู้พัฒนา

แม้ว่ารายละเอียดของกระบวนการนี้อาจดูเหมือนเป็นภาระสำหรับโปรแกรมขนาด เล็ก แต่จำไว้ว่าเรามี "ระเบียบวาระ" ที่สอง ไม่เฉพาะแต่โปรแกรมใหม่เท่านั้นที่ต้องมีประสิทธิภาพ ที่น่าพึงพอใจ เรายังต้องทำให้มั่นใจว่าการเพิ่มโปรแกรมนั้นลงในโปรแกรมที่มีอยู่ ไม่ทำให้ประสิทธิภาพของโปรแกรมอื่นที่รันอยู่บนระบบนั้น ด้อยลง

identification คอมโพเนนต์เวิร์กโหลด

ไม่ว่าโปรแกรมนั้นจะเป็นโปรแกรมใหม่หรือเพิ่งซื้อ โปรแกรมขนาดเล็กหรือใหญ่ ผู้พัฒนา โปรแกรมติดตั้งและผู้ใช้ที่คาดหวังจะมีสมมุติฐานเกี่ยวกับการใช้โปรแกรม

สมมุติฐานบางส่วนเหล่านี้อาจจะเป็น:

- ผู้ที่กำลังจะใช้โปรแกรม
- สถานการณ์ที่โปรแกรมต้องรัน
- ความถี่ของสถานการณ์เหล่านั้นที่จะเกิดขึ้น และเวลาที่เป็นชั่วโมง วัน เดือน หรือปี
- สถานการณ์เหล่านั้นจะยังต้องการใช้ของโปรแกรมที่มีอยู่เพิ่มเติม
- ระบบที่โปรแกรมจะรัน
- ปริมาณข้อมูลที่จะถูกจัดการ และตำแหน่งที่จะถูกจัดการ
- ข้อมูลที่สร้างหรือโปรแกรมที่จะใช้ด้วยวิธีอื่น

เว้นเสียแต่แนวคิดเหล่านี้จะถูกนำออกมาเป็นส่วนหนึ่งของกระบวนการออกแบบ ซึ่งแนวคิดเหล่านั้นอาจไม่ชัดเจน และโปรแกรมเมอร์จะมีสมมุติฐานที่แตกต่างจาก ที่ผู้ใช้คาดหวัง แม้ว่าในกรณีที่ไม่มีความสำคัญ โปรแกรมเมอร์ยังคงเป็นผู้ใช้ซึ่งทิ้งสมมุติฐานที่ไม่ชัดเจน ซึ่งทำให้เป็นไปได้ยากที่จะเปรียบเทียบการออกแบบกับสมมุติฐานด้วยวิธีที่แม่นยำ ที่แย่กว่านั้นเป็นไปไม่ได้ที่จะระบุข้อกำหนดเกี่ยวกับผลการทำงานโดยไม่มีความเข้าใจที่แท้จริงของงาน ที่ถูกดำเนินการ

เอกสารคู่มือเกี่ยวกับข้อกำหนดผลการทำงาน

ในการระบุและการรับรองข้อกำหนดเกี่ยวกับผลการทำงาน สิ่งสำคัญคือ การระบุเหตุผลเบื้องหลังข้อกำหนดโดยเฉพาะ นี่คือส่วนของกระบวนการวางแผนความสามารถทั่วไป ผู้ใช้อาจมีพื้นฐานสำหรับข้อกำหนดของคำสั่ง สำหรับสมมุติฐานเกี่ยวกับตรรกะของโปรแกรมที่ไม่ตรงกับ สมมุติฐานของโปรแกรมเมอร์

ที่จุดต่ำสุด ชุดของข้อกำหนดเกี่ยวกับผลการทำงานควรทำเป็นเอกสารต่อไปนี้:

- การตอบสนองเวลาตอบสนองสูงสุดที่พบกับเวลาส่วนใหญ่สำหรับชนิดที่แตกต่างของการโต้ตอบ กับคอมพิวเตอร์ของผู้ใช้ พร้อมกับนิยามของ *เวลาส่วนใหญ่* เวลาตอบสนองจะถูกวัดค่าจากเวลาที่ผู้ใช้ดำเนินการกับการดำเนินการที่บอกว่า "ไป" จนกว่าผู้ใช้จะได้รับผลตอบกลับที่เพียงพอจากคอมพิวเตอร์ เพื่อดำเนินการกับภารกิจต่อ ซึ่งเป็นช่วงเวลารอที่เกี่ยวข้องของผู้ใช้ แต่ไม่ใช่จากรายการไปยังรูทีนย่อยจนกว่าคำสั่งเขียนในครั้งแรก
  ถ้า ผู้ใช้ปฏิเสธความสนใจในเรื่องของเวลาตอบสนอง และบ่งชี้ว่า เฉพาะผลลัพธ์ของความสนใจนั้น คุณสามารถได้รับคำถามที่ว่า "สิบเท่าของการประมาณการปัจจุบันของเวลาในการประมวลผล แบบเดี่ยว" จะสามารถยอมรับได้ ถ้าคำตอบคือ "ใช่" คุณสามารถดำเนินการอภิปรายถึงทรูพุตได้ หรือ คุณสามารถดำเนินการกับการอภิปรายเวลาตอบสนองด้วย ความสนใจของผู้ใช้
- เวลาตอบสนองที่สามารถยอมรับได้สำหรับเวลาส่วนที่เหลือ เวลาตอบสนองที่ยาวนาน สามารถเป็นสาเหตุทำให้ผู้ใช้คิดถึงเหตุการณ์ที่ระบบหยุดทำงาน คุณยังต้องการระบุ *เวลาที่เหลือ* ด้วย ตัวอย่างเช่น จำนวนนาทีสูงสุดของวัน 1 เปอร์เซ็นต์ของการโต้ตอบ เวลาตอบสนองสำหรับการลดระดับสามารถเป็นต้นทุน หรือเกิดความเสียหายในเวลาของวันโดยเฉพาะ
- ทรูพุตที่ต้องการและเวลาที่จะเข้าแทนที่ ซึ่งไม่ใช่ข้อควรพิจารณาอย่างไม่เป็นทางการ ตัวอย่างเช่น ข้อกำหนดสำหรับหนึ่งโปรแกรม อาจรันสองครั้งต่อวัน: ที่ 10:00 a.m. และ 3:15 p.m. ถ้าสิ่งนี้คือโปรแกรมที่จำกัด CPU ซึ่งรันโดยใช้เวลา 15 นาที และถูกวางแผนที่จะรันบนระบบหลายผู้ใช้ และการต่อรองในการเรียงลำดับ
- ขนาดและการจัดเวลาของระยะเวลาทรูพุตสูงสุด
- การผสมผสานคำร้องขอที่คาดการณ์ไว้และวิธีการผสารที่ผันแปรกับเวลา
- จำนวนของผู้ใช้ต่อเครื่องและจำนวนทั้งหมดของผู้ใช้ ถ้านี่คือแอ็พพลิเคชันหลายผู้ใช้ คำอธิบายนี้ควรสอดแทรกเวลาที่ผู้ใช้เหล่านั้นล็อกออนและล็อกออฟ เช่นเดียวกับอัตราของการกดแป้นพิมพ์ คำร้องขอที่เสร็จสิ้น และเวลาที่คิด คุณอาจต้องการตรวจสอบเวลาที่คิดซึ่งผันแปรตามคำร้องขอก่อนหน้า และคำร้องขอถัดไป
- สมมุติฐานใดๆ ที่ผู้ใช้สร้างขึ้นในเครื่องที่เวิร์กโหลด จะถูกรัน ถ้าผู้ใช้มีเครื่องที่มีอยู่แล้วในใจ ให้ตรวจสอบว่า คุณทราบแล้วแต่เนิ่นๆ เช่นเดียวกัน ถ้าผู้ใช้มีชนิด ขนาด ต้นทุน ตำแหน่ง การเชื่อมต่อระหว่างกัน หรือตัวแปรอื่นๆ ที่จะจำกัดความสามารถในการตอบสนองข้อกำหนดก่อนหน้า สมมุติฐานจะกลายเป็นส่วนหนึ่งของข้อกำหนด ความพึงพอในจะไม่ได้นำมาประเมินผลบนระบบ ซึ่งพัฒนาโปรแกรม ทดสอบโปรแกรม หรือติดตั้งในครั้งแรก

การประมาณการข้อกำหนดรีซอร์สของเวิร์กโหลด

เว้นเสียแต่ว่าคุณได้ซื้อซอฟต์แวร์แพ็กเก็ตที่มาพร้อมกับเอกสารคู่มือข้อกำหนดรีซอร์สโดยละเอียด การประมาณการรีซอร์สสามารถเป็นภารกิจที่ยาก ในกระบวนการวางแผนผลการทำงาน

ความยากมีหลายสาเหตุดังนี้:

- มีหลายวิธีในการทำภารกิจใดๆ คุณสามารถเขียนโปรแกรม C (หรือภาษาระดับสูงอื่นๆ) สคริปต์ shell สคริปต์ **perl** สคริปต์ **awk** สคริปต์ **sed** ไดอะล็อกหน้าต่าง AIX และอื่นๆ เทคนิคบางอย่างที่อาจเหมาะกับอัลกอริธึม และสำหรับโปรแกรมเมอร์จะมีราคาสูง จากมุมมองในด้านผลการทำงาน

  คำแนะนำที่มีประโยชน์ นั่นคือ ระดับของการทำงาน ข้อควรระวังเพิ่มเติมเป็นสิ่งจำเป็น เพื่อให้มั่นใจว่า ไม่ได้รับผลการทำงานที่ทำให้ประหลาดใจ พิจารณาวอลุ่มข้อมูล และจำนวนของการวนซ้ำที่หมายความถึงโครงสร้างที่อาจเกิดอันตรายได้
- ต้นทุนที่แน่ชัดของการประมวลผลเดี่ยวคือความยากในการนิยาม ความยากนี้ไม่ใช่เพียงแค่เทคนิค แต่เป็นปรัชญา ถ้าตัวอย่างของโปรแกรมจำนวนมาก รันโดยผู้ใช้จำนวนมากคือการแบ่งใช้เพจของข้อความโปรแกรม ซึ่งการประมวลผลควรใช้กับเพจของหน่วยความจำเหล่านี้หรือไม่? ระบบปฏิบัติการจะปล่อยเพจไฟล์ที่ใช้ในหน่วยความจำเพื่อจัดเตรียมผล กระทบของการแคชสำหรับโปรแกรม ที่เข้าถึงข้อมูลใหม่ โปรแกรมควรเข้าถึงข้อมูลอีกครั้งเพื่อใช้พื้นที่ ที่ถูกใช้เพื่อเก็บข้อมูลหรือไม่? กระบวนการขนาดเล็กของการวัดค่า เช่น นาฬิกาของระบบสามารถเป็นสาเหตุของค่าความต่างในเวลา CPU ที่แอ็ตทริบิวต์ไปยังอินสแตนซ์ของโปรแกรมเดียวกัน

  วิธีทั้งสองนี้จะทำงานกับความกำกวม ความผันแปรของรายงานรีซอร์ส สิ่งแรกคือ ละเว้นความกำกวม และกำจัดซอร์สของความผันแปรจนกระทั่งการวัดค่าเป็นที่ยอมรับถึง ความสอดคล้องกัน วิธีที่สองคือ ความพยายามในการทำการวัดที่เป็นจริงได้ และอธิบายถึงผลลัพธ์ในเชิงสถิติ หมายเหตุ อัตราผลประโยชน์จะมีผลกับสหสัมพันธ์กับสถานการณ์ของการใช้จริง
- ระบบจะเป็นการทำงานแบบเฉพาะงานเพื่อรันอินสแตนซ์เดี่ยวของโปรแกรมเดี่ยว และจะมี daemon ที่รันอยู่เสมอ ซึ่งมีกิจกรรมการสื่อสารบ่อย และมีเวิร์กโหลดจากผู้ใช้จำนวนมาก กิจกรรมเหล่านี้จะเพิ่มขึ้นในเชิงเส้น ตัวอย่างเช่น การเพิ่มจำนวนของอินสแตนซ์ของโปรแกรมที่กำหนดไว้ อาจส่งผลทำให้ข้อความของโปรแกรมใหม่ถูกนำมาใช้ เนื่องจากโปรแกรมส่วนใหญ่จะอยู่ในหน่วยความจำ อย่างไรก็ตาม การประมวลผล อาจส่งผลทำให้ contention เพิ่มขึ้นสำหรับแคชของตัวประมวลผล ดังนั้น จึงไม่ได้ทำเฉพาะ การประมวลผลอื่นๆที่มีการแบ่งใช้เวลาของตัวประมวลผลที่มาใหม่ แต่การประมวลผลทั้งหมดอาจพบกับวงรอบต่อคำสั่งมากขึ้น นั่นคือ การลดความเร็วลงของตัวประมวลผล เนื่องจากผลลัพธ์ของความถี่แคชเพิ่มเติมนั้นหายไป

ให้ทำการประมาณการด้วยความเป็นจริงตามที่สถานการณ์ที่ระบุอนุญาต ให้ใช้คำแนะนำต่อไปนี้:
- ถ้าโปรแกรมมีอยู่ให้วัดค่าการติดตั้งที่มีอยู่ซึ่งจะรวมไว้กับข้อกำหนด ของคุณเอง วิธีที่ดีที่สุดคือ การใช้เครื่องมือการประมาณความสามารถ เช่น BEST/1
- ถ้าไม่มีการติดตั้งที่เหมาะสมซึ่งพร้อมใช้งาน ให้ทำการติดตั้งแบบทดสอบและ วัดค่าเวิร์กโหลดจากการทดสอบ
- ถ้าเป็นไปไม่ได้ที่จะสร้างเวิร์กโหลดสำหรับการทดสอบซึ่งตรงกับข้อกำหนด ให้วัดค่าการโต้ตอบแต่ละค่า และใช้ผลลัพธ์จากการอินพุต เพื่อจำลอง
- ถ้าโปรแกรมไม่มีอยู่ ให้ค้นหาโปรแกรมที่เปรียบเทียบได้ ซึ่งใช้ภาษาเดียวกันและโครงสร้างทั่วไป และทำการวัดค่า อีกครั้ง ข้อสรุปเกี่ยวกับภาษา การดูแลเพิ่มเติมที่จำเป็นในการพิจารณาความสามารถในการนำมาเปรียบเทียบ
- ถ้าไม่มีโปรแกรมที่สามารถนำมาเปรียบเทียบอยู่ ให้พัฒนาต้นแบบของอัลกอริธึมหลัก ในภาษาที่วางแผนไว้ วัดค่าต้นฉบับ และสร้างแบบจำลองเวิร์กโหลด
- หากการวัดของชนิดใดๆ ที่เป็นไปไม่ได้ คุณควรทำการศึกษา ให้มั่นใจ ถ้ามีความต้องการคาดคะเนข้อกำหนดรีซอร์ส ในระหว่างขั้นตอนการวางแผน จึงเป็นสิ่งสำคัญที่โปรแกรมที่ใช้จริงจะถูกวัดค่า ที่ขั้นตอนแรกๆ ของการพัฒนา

โปรดระลึกไว้ว่า independent software vendors (ISV) มีคำแนะนำในการวัดขนาด สำหรับแอ็พพลิเคชัน

ในการประมาณการรีซอร์ส เราสนใจในมิติทั้งสี่ (ไม่มีลำดับ):

เวลา CPU
: ต้นทุนตัวประมวลผลของเวิร์กโหลด

การเข้าถึงดิสก์
อัตราที่เวิร์กโหลดสร้างการอ่านหรือเขียนดิสก์

ทราฟิก LAN
จำนวนของแพ็กเก็ตที่เวิร์กโหลดสร้าง และจำนวนไบต์ของข้อมูล ที่แลกเปลี่ยน

หน่วยความจำที่ใช้จริง
จำนวนของ RAM ที่เวิร์กโหลดต้องการ

ส่วนต่อไปนี้จะกล่าวถึงวิธีพิจารณาค่าเหล่านี้ในสถานการณ์ต่างๆ

การวัดค่ารีซอร์สเวิร์กโหลด

ถ้าโปรแกรมจริง โปรแกรมที่สามารถเปรียบเทียบได้ หรือต้นแบบ พร้อมใช้งานสำหรับการวัดค่า ตัวเลือกของเทคนิคจะขึ้นอยู่กับหลายๆ ปัจจัย

ปัจจัยเหล่านี้คือ:

- ไม่ว่าระบบกำลังประมวลผลงานอื่นอยู่นอกเหนือจากเวิร์กโหลด ที่เราต้องการวัดค่า
- ไม่ว่าเราจะมีสิทธิในการเครื่องมือที่อาจลดระดับผลการทำงานลง ตัวอย่างเช่น ระบบนี้คือระบบที่ใช้งานจริง หรือใช้เฉพาะกับช่วงเวลาของการวัด?
- ระดับที่เราสามารถจำลองหรือเฝ้าดูเวิร์กโหลดที่แท้จริง

การวัดเวิร์กโหลดที่สมบูรณ์แบบบนระบบเฉพาะงาน:

การใช้ระบบเฉพาะงานคือสถานการณ์ที่อยู่ในความคิด เนื่องจาก เราสามารถใช้การวัดที่สอดแทรกการใช้ระบบ พร้อมกับต้นทุนของการประมวลผล

หากต้องการวัดค่าผลการทำงานของระบบทั้งหมดสำหรับกิจกรรมของระบบโดยส่วนใหญ่ ให้ใช้คำสั่ง **vmstat** :

```
# vmstat 5 >vmstat.output
```

ซึ่ง จะแสดงภาพของสถานะของระบบทุกๆ 5 วินาทีในระหว่างที่รันการวัดค่า ชุดแรกของเอาต์พุต **vmstat** จะมีข้อมูลการสะสมจากการบูตครั้งล่าสุด เพื่อเริ่มต้นคำสั่ง **vmstat** ชุดที่เหลืออยู่จะมีผลลัพธ์สำหรับช่วงเวลาก่อนหน้า ในกรณีนี้คือ 5 วินาที ชุดตัวอย่างของเอาต์พุต **vmstat** บนระบบ จะดูคล้ายกับที่แสดงดังต่อไปนี้:

```
kthr     memory             page              faults        cpu
----- ----------- ------------------------ ------------ -----------
 r  b   avm   fre  re  pi  po  fr   sr  cy  in   sy  cs us sy id wa
 0  1 75186   192   0   0   0   0    1   0 344 1998 403  6  2 92  0
```

หากต้องการวัดค่า CPU และกิจกรรมดิสก์ ให้ใช้คำสั่ง **iostat** :

```
# iostat 5 >iostat.output
```

ซึ่ง จะแสดงภาพของสถานะของระบบทุกๆ 5 วินาทีในระหว่างที่รันการวัดค่า ชุดแรกของเอาต์พุต **iostat** จะมีข้อมูลการสะสมจากการบูตครั้งล่าสุดเพื่อเริ่มต้นคำสั่ง **iostat** ชุดที่เหลืออยู่จะมีผลลัพธ์สำหรับช่วงเวลาก่อนหน้า ในกรณีนี้คือ 5 วินาที ชุดตัวอย่างของเอาต์พุต **iostat** บนระบบที่ดูคล้ายกับที่แสดงดังต่อไปนี้:

```
tty:      tin         tout   avg-cpu:  % user    % sys     % idle    % iowait
          0.0          0.0              19.4      5.7       70.8       4.1
```

```
Disks:        % tm_act     Kbps      tps    Kb_read   Kb_wrtn
hdisk0           8.0      34.5       8.2         12       164
hdisk1           0.0       0.0       0.0          0         0
cd0              0.0       0.0       0.0          0         0
```

หากต้องการวัดค่าหน่วยความจำ ให้ใช้คำสั่ง **svmon** คำสั่ง **svmon -G** จะให้ภาพของการใช้หน่วยความจำทั้งหมด สถิติอยู่ในเงื่อนไขของ เพจขนาด 4 KB:

```
# svmon -G

               size      inuse       free        pin    virtual
memory        65527      65406        121       5963      74711
pg space     131072      37218

               work       pers       clnt       lpage
pin            5972          0          0           0
in use        54177       9023       2206           0
```

ในตัวอย่างนี้ หน่วยความจำขนาด 256 MB ของเครื่องจะถูกใช้งานจนหมด ประมาณ 83 เปอร์เซ็นต์ของ RAM จะถูกใช้งานสำหรับเซ็กเมนต์การทำงาน หน่วยความจำอ่าน/เขียน ของโปรแกรมที่รัน (ส่วนที่เหลือจะใช้สำหรับการทำแคชไฟล์) ถ้ามีการประมวลผลที่ใช้เวลาในการรันยาวนาน ซึ่งเรากำลังสนใจอยู่ เราสามารถตรวจทานข้อกำหนดหน่วยความจำเหล่านั้นในรายละเอียดได้ ตัวอย่างต่อไปนี้ จะพิจารณาหน่วยความจำที่ใช้โดยการประมวลผลของผู้ใช้ hoetzel

```
# ps -fu hoetzel
     UID   PID  PPID   C    STIME    TTY  TIME CMD
 hoetzel 24896 33604   0 09:27:35  pts/3  0:00 /usr/bin/ksh
 hoetzel 32496 25350   6 15:16:34  pts/5  0:00 ps -fu hoetzel

# svmon -P 24896

------------------------------------------------------------------------------
     Pid Command        Inuse      Pin     Pgsp  Virtual   64-bit    Mthrd    LPage
   24896 ksh             7547     4045     1186     7486        N        N        N

  Vsid     Esid Type Description          LPage   Inuse   Pin Pgsp Virtual
     0        0 work kernel seg              -    6324  4041 1186  6324
6a89aa        d work shared library text     -    1064     0    0  1064
72d3cb        2 work process private         -      75     4    0    75
401100        1 pers code,/dev/hd2:6250      -      59     0    -     -
 3d40f        f work shared library data     -      23     0    0    23
16925a        - pers /dev/hd4:447            -       2     0    -     -
```

เซ็กเมนต์การทำงาน (5176) พร้อมกับเพจ 4 หน้าที่ใช้อยู่ คือต้นทุนของอินสแตนซ์นี้ของโปรแกรม **ksh** ต้นทุนเพจ 2619 ของไลบรารีที่แบ่งใช้และต้นทุน 58 เพจของโปรแกรม **ksh** จะกระจายระหว่างโปรแกรมที่รันอยู่ทั้งหมด และอินสแตนซ์ทั้งหมดของโปรแกรม **ksh** ตามลำดับ

ถ้าเราเชื่อว่า ระบบขนาด 256 MB ของเรามีขนาดใหญ่เกินกว่าความจำเป็น ให้ใช้คำสั่ง **rmss** เพื่อลดขนาดที่มีผลกระทบของเครื่อง และวัดค่าเวิร์กโหลดใหม่อีกครั้ง ถ้าการเพิจเพิ่มขึ้นหรือเวลาตอบสนองที่ช้าลง เราต้องลดหน่วยความจำที่มากเกินไป เทคนิคนี้สามารถดำเนินการได้ จนกว่าเราจะค้นหาขนาดที่รันเวิร์กโหลดของเราโดยไม่มีการลดระดับลง โปรดดู “การประเมินผลข้อกำหนดด้านหน่วยความจำด้วยคำสั่ง rmss” ในหน้า 155 สำหรับข้อมูลเพิ่มเติมเกี่ยวกับ เทคนิคนี้

คำสั่งหลักสำหรับการวัดค่าการใช้เน็ตเวิร์กคือ โปรแกรม **netstat** ตัวอย่างต่อไปนี้จะแสดงกิจกรรมของอินเตอร์เฟสโทเค็นริงที่ระบุเฉพาะ:

```
# netstat -I tr0 5
   input    (tr0)     output            input   (Total)    output
 packets  errs  packets  errs colls   packets  errs  packets  errs colls
35552822 213488 30283693     0     0  35608011 213488 30338882     0     0
     300      0      426     0     0       300      0      426     0     0
     272      2      190     0     0       272      2      190     0     0
     231      0      192     0     0       231      0      192     0     0
     143      0      113     0     0       143      0      113     0     0
     408      1      176     0     0       408      1      176     0     0
```

บรรทัดแรกของรายงาน แสดงทราฟิกของเน็ตเวิร์กที่สะกสมตั้งแต่ การบูตครั้งล่าสุด บรรทัดถัดมาแต่ละบรรทัดแสดงกิจกรรมสำหรับช่วงเวลาก่อนหน้า 5 วินาที

การดำเนินการวัดเวิร์กโหลดให้เสร็จสิ้นบนระบบที่ใช้จริง:

เทคนิคของการวัดบนระบบที่ใช้จริง จะคล้ายกับที่ใช้บนระบบเฉพาะงาน แต่เราต้องทำด้วยความระมัดระวัง เพื่อหลีกเลี่ยงผลการทำงานของระบบที่ลดระดับลง

ซึ่งมีความเป็นไปได้สำหรับทูลที่เหมาะสมคือ คำสั่ง **vmstat** ซึ่งจะจัดหาข้อมูลเกี่ยวกับหน่วยความจำ I/O และการใช้ CPU ในรายงานเดี่ยว ถ้าช่วงเวลา **vmstat** ถูกเก็บไว้ค่อนข้างนานสมเหตุสมผล ตัวอย่างเช่น 10 วินาที ต้นทุนเฉลี่ยจะต่ำลงตามความสัมพันธ์ โปรดดู “การระบุรีซอร์สที่จำกัดประสิทธิภาพ” ในหน้า 37 สำหรับข้อมูลเพิ่มเติมเกี่ยวกับการใช้คำสั่ง **vmstat**

การวัดค่าเวิร์กโหลดบางส่วนบนระบบที่ใช้จริง:

ด้วยเวิร์กโหลดบางส่วน เราให้ความหมายของการวัดค่าส่วนของเวิร์กโหลดของระบบที่ใช้จริง สำหรับการถ่ายโอนที่อาจเป็นไปได้หรือทำซ้ำระบบอื่นๆ

เนื่องจากนี่คือระบบที่ใช้งานจริง เราต้องไม่รุกล้ำ เท่าที่จะทำได้ ในเวลาเดียวกัน เราต้องวิเคราะห์เวิร์กโหลดในรายละเอียด เพื่อแบ่งแยกระหว่างส่วนที่เราสนใจ และส่วนที่เราไม่สนใจ ในการทำการวัดค่าเป็นบางส่วน เราต้องค้นหาองค์ประกอบของเวิร์กโหลด ที่สนใจทั่วไป นั่นคือ:

- โปรแกรมเดียวกันหรือชุดของโปรแกรมอื่นๆ ที่เกี่ยวข้องหรือไม่?
- งานจะถูกดำเนินการโดยผู้ใช้ของระบบที่ระบุไว้ตั้งแต่หนึ่งรายขึ้นไปใช่หรือไม่?
- งานจะมาจากเทอร์มินัลที่ระบุไว้ตั้งแต่หนึ่งเทอร์มินัลขึ้นไปใช่หรือไม่?

ซึ่งขึ้นอยู่กับความเป็นธรรมดา เราควรใช้หนึ่งในคำสั่งต่อไปนี้

```
# ps -ef | grep pgmname
# ps -fuusername, . . .
# ps -ftttyname, . . .
```

เพื่อระบุการประมวลผลในสิ่งที่น่าสนใจ และรายงานการใช้เวลา CPU ที่สะสมไว้ของการประมวลผลเหล่านั้น เราสามารถใช้คำสั่ง **svmon** (อย่างรอบครอบ) เพื่อประเมินผลการใช้หน่วยความจำของการประมวลผล

การวัดค่าโปรแกรมแต่ละโปรแกรม:

เครื่องมือจำนวนมากที่พร้อมใช้งานสำหรับการวัดค่าการใช้รีซอร์ส ของโปรแกรมแต่ละโปรแกรม โปรแกรมเหล่านี้บาง โปรแกรมมีความสามารถในการวัดค่าเวิร์กโหลดที่ครอบคลุมได้ด้วยเช่นกัน แต่จะมีการก้าวก่ายมากเกินไปสำหรับการใช้ บน ระบบที่ใช้งานจริง

เครื่องมือเหล่านี้จะถูกกล่าวถึงในรายละเอียดในส่วนที่กล่าวถึงการปรับการใช้งานให้น้อยที่สุด ของรีซอร์สที่ระบุเฉพาะ เครื่อง มือที่ดูเด่นชัดในเรื่องนี้คือ :

svmon วัดหน่วยความจำที่ใช้จริงซึ่งใช้โดยการประมวลผล กล่าวถึงใน “การใช้หน่วยความจำ” ในหน้า 139

time วัดเวลาในการทำงานที่ผ่านไป และการใช้ CPU ของโปรแกรมแต่ละโปรแกรม กล่าวถึงใน “การใช้คำสั่ง time เพื่อ ประเมินการใช้ microprocessor” ในหน้า 121

tprof วัดการใช้งาน CPU ที่เกี่ยวข้องของโปรแกรม ไลบรารีรูทีนย่อย และเคอร์เนลของระบบปฏิบัติการ กล่าวถึงใน เครื่อง มือการสร้างโปรไฟล์ ในส่วนของ *Performance Tools Guide and Reference*

vmstat -s
วัดค่าการโหลด I/O ที่สร้างขึ้นโดยโปรแกรม กล่าวถึงใน “การประเมินค่าดิสก์ I/O ทั้งหมดด้วยคำสั่ง vmstat” ในหน้า 205

การประมาณการรีซอร์สที่ต้องการโดยโปรแกรมใหม่

การสร้างและการออกแบบใหม่ที่ใช้แทนในระหว่างการโค้ดเฟสการคาดการณ์ defy แต่คำแนะนำต่อไปนี้สามารถช่วยคุณได้รับ ข้อกำหนดทั่วไป

เป็นไปไม่ได้ที่จะสร้างการประมาณการที่แน่ชัดของโปรแกรมที่ไม่ได้เขียน จากจุดเริ่มต้น โปรแกรมต่ำสุดต้องการสิ่งต่อไปนี้:

- ประมาณ 50 มิลลิวินาทีของเวลา CPU ซึ่งเป็นเวลาของระบบ
- หน่วยความจำจริง
  - หนึ่งเพจสำหรับข้อความโปรแกรม
  - ประมาณ 15 เพจ (2 เพจที่ถูกตรึงไว้) สำหรับเซ็กเมนต์ใช้งาน (ข้อมูล)
  - เข้าถึง `libc.a` โดยปกตินี่จะถูกแบ่งใช้กับโปรแกรมอื่นๆ ทั้งหมดและถูกพิจารณาเป็นส่วนหนึ่งของต้นทุนพื้นฐานของ ระบบปฏิบัติการ
- ประมาณ 12 เพจในการดำเนินการกับดิสก์ I/O ถ้าโปรแกรมไม่ได้ถูกคอมไพล์ ทำสำเนา หรือใช้ หรือ ไม่จำเป็นต้องมี

สำหรับข้อความข้างต้น ให้เพิ่มงบประมาณต้นทุนพื้นฐานสำหรับความต้องการ ด้วยการออกแบบ (หน่วยที่กำหนดไว้สำหรับ วัตถุประสงค์ที่ใช้เป็นตัวอย่างเท่านั้น):

- เวลา CPU
  - การใช้ CPU ของโปรแกรมปกติที่ไม่มีระดับของการวนซ้ำที่สูง หรือการเรียกรูทีนย่อยที่ไม่สามารถวัดค่าได้
  - ถ้าโปรแกรมที่นำเสนอมีอัลกอริธึมที่มีราคาสูง ให้พัฒนาต้นแบบและวัดค่าอัลกอริธึม
  - ถ้าโปรแกรมที่นำเสนอใช้รูทีนไลบรารีที่มีราคาสูง เช่น X หรือโครงสร้าง Motif หรือรูทีนย่อย **printf( )** ให้วัดค่าการใช้ CPU ด้วยโปรแกรมที่มีขนาดเล็ก
- หน่วยความจำจริง

- อนุญาตให้มี 350 บรรทัดของโค้ดต่อเพจของข้อความโปรแกรม ซึ่งมีขนาดประมาณ 12 ไบต์ต่อบรรทัด โปรดระลึกไว้ว่า ลักษณะของการโค้ดและอ็อพชันของการคอมไพล์ อาจทำให้เกิดความแตกต่างในเรื่องของปัจจัยหรือในเรื่องของทิศทาง งบประมาณนี้ใช้สำหรับเพจที่ถูกสัมผัสในสถานการณ์จำลองตัวอย่าง ถ้าการออกแบบของคุณวางรูทีนย่อยที่ประมวลผลบ่อย ที่ส่วนท้ายของโปรแกรมเรียกทำงาน เพจเหล่านั้นจะไม่ใช้หน่วยความจำจริง
  - การอ้างอิงกับไลบรารีที่แบ่งใช้ที่นอกเหนือจาก `libc.a` จะเพิ่มข้อกำหนดของหน่วยความจำเท่านั้น เพื่อขยายหน่วยความจำที่ไลบรารีเหล่านั้นไม่ได้แบ่งใช้กับโปรแกรมอื่น หรืออินสแตนซ์ของโปรแกรมที่ต้องการประมาณการ เมื่อต้องการวัดค่าขนาดของไลบรารีเหล่านั้น ให้เขียนโปรแกรมขนาดเล็กและใช้เวลาในการรันยาวนาน ซึ่งอ้างอิงไลบรารีเหล่านั้นและใช้คำสั่ง **svmon -P** พร้อมกับ การประมวลผล
  - ประมาณการจำนวนของหน่วยเก็บที่ต้องการโดยโครงสร้างข้อมูล ที่ระบุในการออกแบบ ปัดขึ้นให้เป็นจำนวนเพจใกล้เคียงที่สุด
  - ในการรันที่ใช้เวลาสั้น การดำเนินการกับดิสก์ I/O แต่ละกระบวนการจะใช้หนึ่งเพจของหน่วยความจำ สมมุติว่าเพจมีสภาพพร้อมใช้งานแล้ว ห้ามสมมุติให้โปรแกรมรอเพจของโปรแกรมอื่น ที่ต้องการล้างข้อมูล
- ดิสก์ I/O
  - สำหรับลำดับ I/O แต่ละ 4096 ไบต์ การอ่านหรือเขียนเป็นสาเหตุหนึ่งของการดำเนินการ I/O เว้นเสียแต่ไฟล์ได้ถูกเข้าถึงเพียงพอแล้ว เพจบางเพจ ยังคงอยู่ในหน่วยความจำ
  - สำหรับการสุ่ม I/O แต่ละการเข้าถึง ที่มีขนาดเล็ก กับเพจอื่นๆ ที่มีขนาด 4096 ไบต์ เป็นสาเหตุหนึ่งของการดำเนินการ I/O เว้นเสียแต่ไฟล์ได้ถูกเข้าถึงเพียงพอแล้ว ซึ่งบางเพจยังคงอยู่ในหน่วยความจำ
  - แต่ละลำดับการอ่านหรือเขียนเพจขนาด 4 KB ในไฟล์ที่มีขนาดใหญ่อาจใช้ 100 หน่วย แต่ละการสุ่มการอ่านหรือเขียนเพจขนาด 4 KB จะใช้ 300 หน่วย โปรดจำไว้ว่า ไฟล์จริงจะไม่ถูกเก็บตามลำดับบนดิสก์ แม้ว่าไฟล์เหล่านั้นจะถูกเขียนและอ่านตามลำดับโดยโปรแกรม และในเวลาต่อมา ต้นทุน CPU ของการเข้าถึงดิสก์ที่แท้จริงจะใกล้เคียงกับต้นทุนการเข้าถึงแบบสุ่ม มากกว่าต้นทุนการเข้าถึงตามลำดับ
- I/O การสื่อสาร
  - ถ้าดิสก์ I/O คือ Network File System (NFS) สำหรับระบบไฟล์ที่ mount แบบรีโมต ดิสก์ I/O จะถูกดำเนินการบนเซิร์ฟเวอร์ แต่ไคลเอ็นต์อาจพบกับความต้องการ CPU และหน่วยความจำที่สูงกว่า
  - RPC ของชนิดการสร้างโหลด CPU RPC ที่นำเสนอในการออกแบบควรมีขนาดเล็ก ถูกแบ็ตช์ ถูกทำเป็นต้นฉบับ และถูกวัดค่าในอนาคต
  - แต่ละลำดับ NFS ของการอ่านหรือเขียนเพจขนาด 4 KB ใช้ประมาณ 600 หน่วย บนไคลเอ็นต์ แต่ละการสุ่ม NFS ของการอ่านหรือเขียนเพจขนาด 4 KB ใช้ประมาณ 1000 หน่วยบนไคลเอ็นต์
  - การเรียกดูเว็บและการให้บริการของเว็บจะหมายความถึง I/O ของเน็ตเวิร์กที่สามารถนำมาพิจารณาได้ ด้วยการเปิดและปิดการเชื่อมต่อ TCP ค่อยข้างบ่อย

การแปลงการประเมินระดับโปรแกรมเป็นการประเมินเวิร์กโหลด

วิธีการที่ดีที่สุดสำหรับการประเมินความต้องการรีซอร์สสูงสุดและตามปกติ คือการใช้โมเดลการจัดคิว เช่น BEST/1

สามารถใช้สแตติกโมดูลได้ แต่คุณต้องเสี่ยงต่อ การประเมินสูงเกินไปหรือการประเมินต่ำเกินไปสำหรับรีซอร์สสูงสุด ในกรณีใดกรณีหนึ่งนั้น คุณ ต้องเข้าใจว่าหลายโปรแกรมในเวิร์กโหลดโต้ตอบกันอย่างไรจากจุดยืนของ ความต้องการรีซอร์ส

ถ้าคุณกำลังสร้างสแตติกโมเดล ให้ใช้ช่วงเวลาที่เป็น เวลาตอบกลับเลวร้ายที่สุดที่ยอมรับได้ซึ่งระบุสำหรับโปรแกรมที่ใช้งานหรือต้องการ บ่อยที่สุด (โดยปกติ เป็นโปรแกรมเดียวกัน) กำหนด โปรแกรมซึ่งโดยปกติ จะรันในระหว่างช่วงเวลาแต่ละช่วงขึ้นอยู่กับจำนวนผู้ใช้ ที่คาดการณ์ของคุณ เวลาลิงก์ อัตราการป้อนข้อมูล และชุดของการดำเนินงาน ที่คาดไว้

ใช้คำแนะนำต่อไปนี้:

- เวลา CPU
  - เพิ่มความต้องการ CPU ของโปรแกรมทั้งหมดที่กำลังรันในระหว่าง ช่วงเวลา รวมความต้องการ CPU ของดิสก์และ I/O การสื่อสารที่โปรแกรมจะทำ
  - ถ้าจำนวนนี้มากกว่า 75 เปอร์เซ็นต์ของเวลา CPU ที่มีอยู่ในระหว่าง ช่วงเวลา ให้ลองใช้ผู้ใช้น้อยลงและ CPUs มากขึ้น
- หน่วยความจำจริง
  - ความต้องการหน่วยความจำของระบบปฏิบัติการเทียบส่วนกับจำนวนของ หน่วยความจำฟิสิคัล เริ่มต้นด้วย 6 ถึง 8 MB สำหรับตัวระบบปฏิบัติการเอง ตัวเลขที่ต่ำกว่าใช้ สำหรับระบบสแตนด์อะโลน ตัวเลขถัดมาใช้สำหรับระบบที่เชื่อมต่อกับ LAN และใช้ TCP/IP และ NFS
  - เพิ่มความต้องการเซกเมนต์การทำงานของอินสแตนซ์ทั้งหมด ของโปรแกรมที่จะกำลังรันในระหว่างช่วงเวลา รวมถึงพื้นที่ว่าง ที่ประเมินสำหรับโครงสร้างข้อมูลของโปรแกรม
  - เพิ่มความต้องการหน่วยความจำของเซกเมนต์ข้อความของแต่ละโปรแกรมเฉพาะ ที่จะกำลังรัน (สำเนาหนึ่งของข้อความโปรแกรมใช้สำหรับอินสแตนซ์ทั้งหมดของ โปรแกรมนั้น) ในผลรวมนั้น โปรดจำว่ารูทีนย่อยใดๆ (และอย่างเดียว) ที่มาจากไลบรารีแบบไม่ได้แบ่ง ใช้จะเป็นส่วนประกอบหนึ่งของโปรแกรมที่ดำเนินการได้ แต่ตัวไลบรารี จะไม่อยู่ในหน่วยความจำ
  - เพิ่มจำนวนของพื้นที่ว่างที่ใช้โดยแต่ละไลบรารีแบบแบ่งใช้ซึ่งจะใช้โดยโปรแกรมใดๆ ในเวิร์กโหลด ในผลรวม อีกครั้งหนึ่งสำเนาให้บริการทั้งหมด
  - เพื่อให้มีพื้นที่ว่างเพียงพอสำหรับการแคชไฟล์และรายการที่ว่างบางอย่าง การคาดการณ์หน่วยความจำทั้งหมดของคุณไม่ควรเกินกว่า 80 เปอร์เซ็นต์ของขนาด ของเครื่องที่จะใช้
- ดิสก์ I/O
  - เพิ่มจำนวนของ I/Os ที่ใช้โดยแต่ละอินสแตนซ์ของแต่ละโปรแกรม แยกผลรวมของ I/Os ไปยังไฟล์ขนาดเล็ก (หรือแบบสุ่มไปยังไฟล์ขนาดใหญ่) ออกจาก การอ่านหรือการบันทึกตามลำดับอย่างแท้จริงของไฟล์ขนาดใหญ่ (มากกว่า 32 KB)
  - ลบ I/Os ที่คุณเชื่อว่าจะได้รับบริการจากหน่วยความจำ เร็กคอร์ด ใดๆ ที่อ่านหรือบันทึกในช่วงเวลาก่อนหน้านี้อาจยังคงมีอยู่ในช่วงเวลา ปัจจุบัน นอกจากนั้น ตรวจสอบขนาดของเครื่องที่นำเสนอและ ความต้องการ RAM ทั้งหมดของเวิร์กโหลดของเครื่อง พื้นที่ว่างใดๆ ที่เหลืออยู่หลังจากความต้องการของระบบปฏิบัติการและความต้องการของเวิร์กโหลด อาจมีหน้าไฟล์ที่อ่านหรือบันทึกล่าสุด ถ้าการออกแบบแอ็พพลิเคชันของคุณ เป็นแบบที่มีโอกาสเป็นไปได้สูงที่คุณจะใช้ข้อมูลที่เข้าถึงล่าสุดซ้ำอีกครั้ง คุณสามารถคำนวณค่าเผื่อสำหรับผลการแคช โปรดจำไว้ว่า การใช้ซ้ำเป็นการใช้ซ้ำที่ระดับหน้า ไม่ใช่ที่ระดับเร็กคอร์ด ถ้าโอกาสของการใช้ซ้ำ ของเร็กคอร์ดที่กำหนดต่ำ แต่มีเร็กคอร์ดจำนวนมากต่อหน้า อาจเป็นไปได้ว่าบางเร็กคอร์ดที่ต้องการในช่วงเวลาที่กำหนดอาจอยู่ในหน้า เดียวกันกับเร็กคอร์ดอื่นที่ใช้ล่าสุด
  - เปรียบเทียบความต้องการ I/O สุทธิ (ดิสก์ I/Os ต่อวินาทีต่อดิสก์) กับความ สามารถโดยประมาณของดิสก์ไดร์ฟปัจจุบัน ถ้าความต้องการแบบสุ่มหรือตามลำดับ มากกว่า 75 เปอร์เซ็นต์ของความสามารถที่ตรงกันทั้งหมดของดิสก์ซึ่งจะจัดเก็บข้อมูลแอ็พพลิเคชัน การปรับ (และอาจเป็นการขยาย) จะเป็นสิ่งจำเป็นเมื่อแอ็พพลิเคชันอยู่ในการผลิต
- I/O การสื่อสาร
  - คำนวณการใช้แบนด์วิธของเวิร์กโหลด ถ้าการใช้แบนด์วิธทั้งหมดของ โหนดทั้งหมดบน LAN มากกว่า 70 เปอร์เซ็นต์ของแบนด์วิธ ปกติ (50 เปอร์เซ็นต์สำหรับอีเทอร์เน็ต) คุณอาจจะต้องการใช้เครือข่าย ที่มีแบนด์วิธสูงขึ้น
  - ทำการวิเคราะห์คล้ายกันสำหรับความต้องการ CPU, หน่วยความจำ และ I/O ของโหลดที่เพิ่มซึ่งจะวางบนเซิร์ฟเวอร์

หมายเหตุ: โปรดจำไว้ว่าคำแนะนำเหล่านี้มีไว้สำหรับการใช้ *เฉพาะ* เมื่อ ไม่สามารถทำการประเมินแบบครอบคลุมเท่านั้น การประเมินเฉพาะแอ็พพลิเคชันใดๆ ที่สามารถใช้ร่วมกับคำแนะนำจะช่วยปรับปรุงความถูกต้องของการประเมิน ได้เป็นอย่างมาก

การออกแบบและการนำโปรแกรมไปปฏิบัติซึ่งมีประสิทธิภาพ

ถ้าคุณได้พิจารณารีซอร์สที่จะจำกัดความเร็วของโปรแกรมของคุณ คุณสามารถไปยังส่วนที่กล่าวถึงเทคนิคที่เหมาะสมได้โดยตรง เพื่อลดปริมาณการใช้รีซอร์สนั้น

หรืออีกนัยหนึ่ง คือ โปรแกรมจะถูกทำให้เกิดความสมดุล และนั่นคือข้อแนะนำทั้งหมดในส่วนนี้ที่จะนำมาใช้ หากโปรแกรมถูกนำไปปฏิบัติ ให้ดำเนินการ “การระบุรีซอร์สที่จำกัดประสิทธิภาพ” ในหน้า 37 ต่อไป

โปรแกรมที่จำกัดตัวประมวลผล

ถ้าโปรแกรมคือตัวประมวลผลที่จำกัด เนื่องจากโปรแกรมนั้นประกอบด้วยการคำนวณเชิงตัวเลขทั้งหมด อัลกอริธึมที่เลือกไว้จะกระทบกับผลการทำงานของโปรแกรมอย่างรุนแรง

ความเร็วสูงสุดของโปรแกรมที่จำกัดตัวประมวลผลจะถูกพิจารณาโดย:

- อัลกอริธึมที่ใช้
- ซอร์สโค้ดและโครงสร้างข้อมูลที่สร้างโดยโปรแกรมเมอร์
- ลำดับของคำสั่งในภาษาเครื่องที่สร้างด้วยคอมไพลเลอร์
- ขนาดและโครงสร้างของแคชของตัวประมวลผล
- สถาปัตยกรรมและอัตราสัญญาณนาฬิกาของตัวประมวลผลเอง (โปรดดู “การพิจารณาถึงความเร็วของไมโครโพรเซสเซอร์” ในหน้า 458)

การอธิบายเกี่ยวกับอัลกอริทึมที่เป็นทางเลือกอย่นอกเหนือขอบเขตของชุดหัวข้อนี้ ซึ่งสมมุติได้ว่า ประสิทธิภาพในการคำนวณจะถูกนำมาพิจารณา ในการเลือกอัลกอริธึม

อัลกอริธึมที่กำหนดไว้ เฉพาะรายการที่อยู่ในรายการส่วนหน้าเท่านั้นที่โปรแกรมเมอร์ สามารถส่งผลกระทบถึงได้คือ ซอร์สโค้ด อ็อพชันคอมไพลเลอร์ที่ใช้โครงสร้างข้อมูลที่อาจเกิดขึ้นได้ ส่วนต่อไปนี้จะเกี่ยวข้องกับเทคนิคที่สามารถนำมาใช้ เพื่อปรับปรุงประสิทธิภาพของโปรแกรมแต่ละโปรแกรมที่ผู้ใช้ มีซอร์สโค้ด ถ้าซอร์สโค้ดไม่มีอยู่ ให้พยายามใช้เทคนิคการปรับหรือเทคนิคการจัดการเวิร์กโหลด

การออกแบบและการโค้ดสำหรับการใช้แคชอย่างมีประสิทธิผล

การใช้หน่วยเก็บอย่างมีประสิทธิผลหมายความถึงการเก็บคำสั่งและข้อมูล ที่ต้องนำมาใช้

ตัวประมวลผลมีลำดับชั้นของหน่วยความจำแบบหลายระดับ:

1. คำสั่งไพพ์ไลน์และเรจิสเตอร์ CPU
2. คำสั่งและข้อมูลแคช และการแปลบัฟเฟอร์โดยรอบให้สอดคล้องกัน
3. RAM
4. ดิสก์

คำสั่งและข้อมูลที่ย้ายขึ้นในลำดับชั้น คำสั่งจะย้ายไปยังหน่วยเก็บที่เร็วกว่าระดับล่าง แต่ยังมีขนาดเล็กกว่า และแพงกว่า หากต้องการขอรับผลการทำงานที่อาจเป็นไปได้จากเครื่องที่กำหนดไว้ โปรแกรมเมอร์ต้องทำการใช้หน่วยเก็บที่พร้อมใช้งานอย่างมีประสิทธิผล ที่แต่ละระดับ

อุปสรรคในบรรลุเป้าหมายของหน่วยเก็บที่มีประสิทธิภาพคือความจริงที่ว่า หน่วยเก็บจะถูกจัดสรรในบล็อกที่มีความยาวไม่เปลี่ยนแปลง เช่น บรรทัดแคช และเพจหน่วยความจำจริง ซึ่งไม่สอดคล้องกับขอบเขตภายในโปรแกรม หรือโครงสร้างข้อมูล โปรแกรมและโครงสร้างข้อมูลที่ถูกออกแบบมา โดยไม่พิจารณาถึงลำดับชั้นของหน่วยเก็บที่ทำให้การใช้งานของหน่วยเก็บที่ถูกจัดสรรไว้ไม่มีประสิทธิภาพ พร้อมกับผลการทำงานที่ตรงข้ามกับผลกระทบในระบบที่มีขนาดเล็กและ มีโหลดมาก

การพิจารณาลำดับชั้นหน่วยเก็บหมายถึงการทำความเข้าใจและ การปรับปรุงหลักการของการโปรแกรมมิงอย่างมีประสิทธิภาพในสภาวะแวดล้อมที่แคช หรือหน่วยความจำเสมือน เทคนิคการทำแพ็กเกจใหม่อาจมีอัตราผลประโยชน์ในการปรับปรุงที่สำคัญ โดยไม่มีการบันทึก และการเขียนโค้ดใหม่ใดๆ ควรถูกออกแบบมาพร้อมกับการใช้หน่วยเก็บ ที่มีประสิทธิภาพ

มีสองคำที่ควรนำมาอภิปรายถึงการใช้อย่างมีประสิทธิภาพของหน่วยเก็บตามลำดับชั้น คือ: *ตำแหน่งของการอ้างอิง* และ *ชุดการทำงาน*

- ตำแหน่งของการอ้างอิงของโปรแกรมคือ ระดับที่แอดเดรสการประมวลผลคำสั่ง และข้อมูลที่อ้างอิงถูกทำคลัสเตอร์ในพื้นที่หน่วเก็บขนาดเล็ก ในระหว่างช่วงเวลาที่กำหนด
- ชุดการทำงานของโปรแกรมในระหว่างช่วงเวลาเดียวกันนั้น คือชุดของบล็อกหน่วยเก็บที่ใช้งานอยู่ หรือ โค้ดหรือข้อมูลที่จองบล็อกเหล่านั้น

โปรแกรมที่มีตำแหน่งการอ้างอิงที่ดีจะมีชุดการทำงานที่ต่ำ เนื่องจากบล็อกที่ใช้งานอยู่จะถูกแพ็กด้วยโค้ดหรือข้อมูล ที่เรียกใช้งาน โปรแกรมที่ทำหน้าที่เทียบเท่ากับตำแหน่งการอ้างอิงที่ไม่ดี จะมีชุดการทำงานที่ใหญ่กว่า เนื่องจากมีบล็อกจำนวนมากที่ต้องการจัดให้เหมาะสมกับช่วงของแอดเดรสที่กำลังเข้าถึง

เนื่องจากแต่ละบล็อกจะใช้จำนวนเวลาในการโหลดลงในระดับของ ลำดับชั้นที่กำหนด วัตถุประส่งของโปรแกรมที่มีประสิทธิภาพนี้สำหรับ ระบบที่มีหน่วยเก็บเป็นลำดับชั้น จะถูกออแบบและทำโค้ดแพ็กเกจด้วยวิธีที่ ชุดการทำงานยังคงเหลือเป็นส่วนเล็ก

รูปภาต่อไปนี้จะแสดงการทดสอบที่ดีและไม่ดีที่ระดับของรูทีนย่อย เวอร์ชันแรกของโปรแกรมคือ การทำแพ็กเกจในลำดับที่มีการเขียน รูทีนแรก **PriSub1** จะมี entry point ของโปรแกรม ซึ่งจะใช้รูทีนย่อยหลัก **PriSub2** และ **PriSub3** ฟังก์ชันที่ใช้ไม่บ่อยบางตัวของโปรแกรมต้องการรูทีนย่อยสำรอง **SecSub1** และ **SecSub2** สำหรับบางโอกาส ข้อผิดพลาดเกี่ยวกับรูทีนย่อย **ErrSub1** และ **ErrSub2** อาจเกิดขึ้นได้

Poor Locality of Reference, Large Working Set

Page 1			Page 2			Page 3
PriSub1	SecSub1	ErrSub1	PriSub2	SecSub2	ErrSub2	PriSub3

Good Locality of Reference, Small Working Set

Page 1			Page 2			Page 3
PriSub1	PriSub2	PriSub3	SecSub1	SecSub2	ErrSub1	ErrSub2

รูปที่ 15. ตำแหน่งของการอ้างอิง. ครึ่งหนึ่งของส่วนบนของรูปภาพประกอบ อธิบายถึงวิธีที่โปรแกรมไบนารีถูกทำแพ็กเกจ ซึ่งแสดงตำแหน่งของการอ้างอิง คำสั่งสำหรับ **PriSub1** อยู่ในไบนารีที่สามารถเรียกทำงานได้ในครั้งแรก ตามด้วยคำสั่งสำหรับ **SecSub1**, **ErrSub1**, **PriSub2**, **SecSub2**, **ErrSub2** และ **PriSub3** ในการเรียกทำงานนี้ คำสั่งสำหรับ **PriSub1**, **SecSub1** และ **ErrSub1** จะจองเพจแรกของหน่วยความจำ คำสั่งสำหรับ **PriSub2**, **SecSub2**, **ErrSub2** จองเพจที่สองของหน่วยความจำ และคำสั่งสำหรับ **PriSub3** จะจองเพจที่สามของหน่วยความจำ **SecSub1** และ **SecSub2** จะถูกใช้ไม่บ่อย และ **ErrSub1** และ **ErrSub2** จะไม่ค่อยถูกใช้ ดังนั้น การทำแพ็กเกจของโปรแกรมนี้จะแสดงตำแหน่งของการอ้างอิงที่ไม่ดี และอาจใช้หน่วยความจำเพิ่มเติมมากกว่าที่ต้องการ ในส่วนที่สองของรูปภาพประกอบ **PriSub1**, **PriSub2** และ **PriSub3** จะอยู่ที่ถัดออกไปและจองเพจแรกของหน่วยความจำ **PriSub3** is **SecSub1**, **SecSub2** และ **ErrSub1** ต่อไปนี้ซึ่งจองเพจที่สองของหน่วยความจำทั้งหมด ท้ายสุด **ErrSub2** คือส่วนท้ายและจองเพจที่สามของหน่วยความจำ เนื่องจาก **ErrSub2** อาจไม่ต้องการ และจะลดข้อกำหนดด้านหน่วยความจำลงหนึ่งเพจ ในกรณีนี้

เวอร์ชันแรกของโปรแกรมจะมีตำแหน่งของการอ้างที่ต่ำ เนื่องจาใช้สามเพจของหน่วยความจำเพื่อรันในกรณีปกติ รูทีนที่สองและข้อผิดพลาดจะแยดออกจากพาธหลักของโปรแกรมในสามส่วน แบบฟิสิคัล

เวอร์ชันที่ปรับปรุงแล้วของโปรแกรมจะวางรูทีนหลักไว้ใกล้กัน และวางฟังก์ชันที่ใช้งานต่ำไว้ด้านหลัง รูทีนข้อผิดพลาดที่จำเป็น (ไม่ค่อยได้ใช้) จะอยู่ทางซ้ายที่ส่วนท้ายของโปรแกรมเรียกทำงาน ฟังก์ชันปกติส่วนใหญ่ของโปรแกรม สามารถจัดการได้ด้วยหนึ่งดิสก์ของการอ่าน และหนึ่งเพจของหน่วยความจำแทนสามเพจที่กล่าวไว้ก่อนหน้านี้

โปรดจำไว้ว่า ตำแหน่งของการอ้างอิงและชุดการทำงานถูกนิยามด้วยเวลา ตามลำดับ ถ้าโปรแกรมทำงานในขั้นตอนต่างๆ แต่ละขั้นตอนจะใช้เวลาและใช้ชุดของรูทีนย่อยต่างกัน ลองลดชุดการทำงานลงในแต่ละขั้นตอน

ทะเบียนและไปป์ไลน์

โดยทั่วไป การจัดสรรและการใช้ประโยชน์พื้นที่ว่างทะเบียนและทำให้ไปป์ไลน์เต็มอยู่เสมอเป็นหน้าที่ของคอมไพเลอร์

หน้าที่หลักของโปรแกรมเมอร์คือการหลีกเลี่ยงโครงสร้างที่เป็นอุปสรรคต่อ เทคนิคการใช้ประโยชน์สูงสุดของคอมไพเลอร์ ตัวอย่างเช่น ถ้าคุณใช้รูทีนย่อยอย่างใดอย่างหนึ่ง ในลูปที่สำคัญของโปรแกรม อาจเหมาะสมที่คอมไพเลอร์จะ inline รูทีนย่อยนั้นเพื่อลดเวลาการดำเนินการให้เหลือน้อยที่สุด อย่างไรก็ตาม ถ้า รูทีนย่อยมีการจัดแพ็กเกจในโมดูล .c อื่น คอมไพเลอร์จะไม่สามารถ inlined รูทีนย่อยนั้นได้

แคชและ TLBs

แคชสามารถเก็บ Translation lookaside buffers (TLBs) ซึ่งมีการแม็พจากแอดเดรสเสมือนกับแอดเดรสจริงของเพจที่ถูกใช้ล่าสุดของ ข้อความคำสั่งหรือข้อมูล

ตัวประมวลผลจะมีแคชตั้งแต่หนึ่งแคชขึ้นไปขึ้นอยู่กับสถาปัตยกรรมและแบบจำลองของตัวประมวลผล เพื่อเก็บสิ่งต่อไปนี้:

- ส่วนของโปรแกรมที่เรียกใช้งาน
- ข้อมูลที่ใช้โดยโปรแกรมที่เรียกใช้งาน
- TLBs

ถ้าแคชเกิดหายไป การโหลดบรรทัดแคชที่สมบูรณ์สามารถใช้วงรอบตัวประมวลผล จำนวนมาก ถ้าแคช TLB เกิดหายไป การคำนวณการแม็พเพจเสมือนกับเพจจริง อาจใช้วงรอบจำนวนมาก ต้นทุนที่แน่นอนจะขึ้นอยู่กับการนำไปปฏิบัติ

แม้ว่าโปรแกรมและข้อมูลจะอยู่ในแคชที่เหมาะสม บรรทัดหรือรายการ TLB ที่ใช้เพิ่มเติม (นั่นคือ ตำแหน่งที่ตั้งของการอ้างอิงที่ต่ำกว่า) วงรอบ CPU เพิ่มเติมต้องถูกนำมาใช้เพื่อขอรับทุกสิ่งที่โหลดเข้ามา เว้นเสียแต่คำสั่งและข้อมูล จะถูกนำกลับมาใช้หลายครั้ง การใช้ของการโหลดข้อมูลเหล่านั้น จะแตกเป็นเศษของเวลาในการประมวลผลโปรแกรมทั้งหมด ซึ่งมีผลกับผลการทำงานของระบบที่ลดลง

เทคนิคโปรแกรมมิงที่ดีจะเก็บบรรทัดหลัก การไหลของโปรแกรม ที่บีบอัดได้เท่าที่ทำได้ โพรซีเดอร์หลักและรูทีนย่อยทั้งหมดที่เรียก จะถูกกระทำอย่างต่อเนื่อง เงื่อนไขความเป็นไปได้ที่ต่ำ เช่น ข้อผิดพลาดที่คลุมเครือควรนำมาทดสอบเฉพาะในบรรทัดหลัก ถ้าเงื่อนไขเกิดขึ้นจริง การดำเนินการควรแทนที่ในรูทีนย่อยที่แยกต่างหาก รูทีนย่อยทั้งหมดควรถูกจัดกลุ่มเข้าด้วยกันที่ส่วนท้ายของโมดูล ซึ่งการจัดเรียงนี้จะลดความสามารถที่เป็นไปได้ ซึ่งโค้ดการใช้งานต่ำ จะถูกนำมาใช้กับพื้นที่ในบรรทัดแคชการใช้ระดับสูง ในโมดูลขนาดใหญ่ รูทีนการใช้งานที่ต่ำกว่าทั้งหมด อาจจองเพจที่ไม่ได้ถูกอ่านไว้ในหน่วยความจำ

หลักการเดียวกันนี้ประยุกต์ใช้กับโครงสร้างข้อมูล แม้ว่าในบางครั้ง จำเป็นต้องเปลี่ยนโค้ดเพื่อชดเชยกฎของการคอมไพล์เกี่ยวกับโครงร่างข้อมูล

ตัวอย่างเช่น การดำเนินการกับแมทริกซ์บางอยาง เช่น การจัดการกับแมทริกซ์ ซึ่งเกี่ยวข้องกับอัลกอริธึมนั้น ถ้าโค้ดมีตำแหน่งของการอ้างอิงที่ไม่ดี การดำเนินการกับแมทริกซ์จะเกี่ยวข้องกับการเข้าถึงข้อมูลแมทริกซ์ตามลำดับ เช่น องค์ประกอบของแถวที่ดำเนินการบนองค์ประกอบ แต่ละคอมไพลเลอร์มีกฎที่ระบุเฉพะ เกี่ยวกับโครงร่างหน่วยเก็บของแมทริกซ์ คอมไพลเลอร์ FORTRAN จะวางโครงร่างแมทริกซ์ ในรูปแบบคอลัมน์หลัก (นั่นคือ องค์ประกอบทังหมดของคอลัมน์ 1 จะตามด้วยองค์ประกอบทั้งหมดของคอลัมน์ 2 และอื่นๆ) คอมไพลเลอร์ C จะวางโครงร่างแมทริกซ์ในรูปแบบของแถวหลัก ถ้าแมทริกซ์มีขนาดเล็ก องค์ประกอบแถวและคอลัมน์สามารถมีได้ในแคชข้อมูล และตัวประมวลผลและการอิงดัชนี สามารถรันที่ความเร็วเต็มที่ได้ อย่างไรก็ตาม เนื่องจากขนาดของแมทริกซ์ที่เพิ่มขึ้นเรื่อยๆ ตำแหน่งของการอ้างอิงของการดำเนินการแถว/คอลัมน์จะแย่ลง ในจุดที่ข้อมูลไม่สามารถรักษาไว้ในแคชได้อีกต่อไป ในความเป็นจริง รูปแบบการเข้าถึงของการดำเนินการแถว/คอลัมน์จะสร้างการรูปแบบการ thrash สำหรับแคชที่สตริงขององค์ประกอบที่ประเมินผลมีขนาดใหญ่กว่าแคช บังคับให้เข้าถึงองค์ประกอบภายนอก จากนั้น ทำซ้ำการเข้าถึงรูปแบบอีกครั้งสำหรับข้อมูลเดียวกัน

โซลูชันโดยทั่วไปสำหรับรูปแบบการเข้าถึงแมทริกซ์จะระบุเฉพาะกับการดำเนินการในบล็อก ดังนั้น การดำเนินการจำนวนมากบนส่วนประกอบเดียวกัน สามารถดำเนินการได้ขณะที่ยังคงอยู่ในแคช เทคนิคทั่วไปนี้จะถูกกำหนดด้วย *strip mining*

ผู้เชี่ยวชาญในเรื่องการวิเคราะห์ตัวเลขจะถูกถามโค้ดเวอร์ชันของอัลกอริธึมการจัดการกับแมทริกซ์ ที่ทำขึ้นเพื่อใช้ strip mining และเทคนิคการทำให้เหมาะสมที่สุด ผลลัพธ์คือการปรับปรุง 30 ส่วนในผลการทำงานสำหรับแมทริกซ์การคูณ รูทีนที่ปรับแล้วจะอยู่ในไลบรารี Basic Linear Algebra Subroutines (BLAS) พาธ `/usr/lib/libblas.a` ชุดของรูทีนย่อยการปรับผลการทำงานที่ใหญ่กว่าคือ ไลเซนส์โปรแกรม Engineering and Scientific Subroutine Library (ESSL)

ฟังก์ชันและอินเตอร์เฟสของ Basic Linear Algebra Subroutines แสดงอยู่ใน *AIX Version 7.1 Technical Reference* สภาวะแวดล้อมรันไทม์ FORTRAN ต้องถูกติดตั้งเพื่อใช้ไลบรารี ผู้ใช้ควรใช้ไลบรารีนี้ สำหรับการดำเนินการกับแมทริกซ์และเวกเตอร์ของตนเอง เนื่องจากรูทีนย่อยได้ถูกปรับระดับที่ผู้ใช้บรรลุเป้าหมาย ด้วยตนเอง

ถ้าโครงสร้างข้อมูลถูกควบคุมโดยโปรแกรมเมอร์ ประสิทธิภาพในการทำงานอื่นๆ อาจเกิดขึ้นได้ หลักการโดยทั่วไปคือ การแพ็กข้อมูลที่ใช้บ่อยเข้าด้วยกัน เมื่อเป็นไปได้ ถ้าโครงสร้างที่มีข้อมูลการควบคุมการประเมินผล และข้อมูลโดยละเอียดที่เข้าถึงเป็นครั้งคราว ตรวจสอบให้แน่ใจว่า การควบคุมข้อมูล จะถูกจัดสรรไว้ในไบต์ที่ต่อเนื่องกัน ซึ่งจะเพิ่มความน่าจะเป็นที่ข้อมูลการควบคุมทั้งหมดจะถูกโหลดลงในแคชเดี่ยว (หรืออย่างน้อยที่สุดด้วยค่าต่ำสุดของ) แคชที่หายไป

ตัวประมวลผลก่อนและการใช้ประโยชน์จากคอมไพลเลอร์

มีระดับของการ optimization หลายระดับที่ให้ระดับของคอมไพลเลอร์ที่แตกต่าง ที่เป็นอิสระในการจัดเรียงคำสั่ง

โปรแกรมเมอร์ผู้ที่ต้องการขอรับผลการทำงานที่อาจเป็นไปได้สูงสุด จากโปรแกรมที่กำหนดซึ่งรันอยู่บนเครื่องที่กำหนดไว้ ต้องเข้ากับข้อควรพิจารณาทั้งหลาย:

- มีตัวประมวลผลก่อนที่สามารถจัดเรียงโครงสร้างซอร์สโค้ดใหม่ ในรูปแบบที่เทียบเท่ากับโมดูลซอร์ส ซึ่งสามารถคอมไพล์ได้ ในโค้ดที่สามารถเรียกทำงานได้อย่างมีประสิทธิภาพ
- เพียงแค่มีความต่างของสถาปัตยกรรมทั้งหมด มีอ็อพชันคอมไพลเลอร์ทั้งหลาย เพื่ออนุญาตให้การคอมไพล์สูงสุดสำหรับความต่างที่ระบุ หรือตั้งค่าความต่าง
- โปรแกรมเมอร์สามารถใช้คุณลักษณะ **#pragma** เพื่อแจ้งให้คอมไพลเลอร์ C ของลักษณะมุมมองของโปรแกรมบางอย่างที่อนุญาตให้คอมไพลเลอร์สร้างโค้ดได้อย่างมีประสิทธิภาพ โดยผ่อนผันข้อสรุปที่แย่ที่สุดบางอย่าง

โปรแกรมเมอร์ผู้ที่ไม่สามารถทำการทดลองได้ ควรทำการออปติไมซ์ ความแตกต่างในเรื่องของผลการทำงาน ระหว่างโค้ดที่ออปติไมซ์แล้วและยังไม่ออปติไมซ์จะมีขนาดใหญ่เสมอ ซึ่งเป็นการ optimization พื้นฐาน (อ็อพชัน **-O** ของคำสั่งคอมไพลเลอร์) ควรนำมาใช้เสมอ เฉพาะข้อยกเว้นเท่านั้นที่กำลังทดสอบสถานการณ์ ที่มีความต้องการสร้างโค้ดเฉพาะ เช่น การวิเคราะห์ผลการทำงานในระดับของคำสั่ง โดยใช้เครื่องมือ **tprof**

เทคนิคเหล่านี้ให้อัตราผลประโยชน์การปรับปรุงผลการทำงานเพิ่มเติมสำหรับโปรแกรมบางโปรแกรม แต่การกำหนดการรวมกันที่ให้อัตราผลประโยชน์เกี่ยวกับผลการทำงานที่ดีที่สุด สำหรับโปรแกรมที่ระบุเฉพาะอาจต้องการความสามารถในการพิจารณาการคอมไพล์ใหม่และการวัด

สำหรับการกล่าวถึงเทคนิคสำหรับการใช้คอมไพลเลอร์ให้มีประสิทธิภาพ โปรดดู *คู่มือการ optimization และการปรับสำหรับ XL Fortran, XL C และ XL C++*

ระดับของการ optimization

ระดับที่คอมไพลเลอร์จะออปติไมซ์โค้ดที่สร้างขึ้น จะถูกควบคุมโดยแฟล็ก **-O**

ไม่มีการ optimization
การไม่มีเวอร์ชันของแฟล็ก **-O** คอมไพลเลอร์จะสร้างโค้ด โดยไม่มีการเรียงลำดับคำสั่งใหม่ หรือความพยายามที่จะปรับปรุง ผลการทำงาน

-O หรือ -O2
แฟล็กที่เท่าเทียมกันเหล่านี้อาจทำให้คอมไพลเลอร์ออปติไมซ์ตามพื้นฐานของสมมุติฐานอย่างต่อเนื่อง เกี่ยวกับการเรียงลำดับโค้ดใหม่ เฉพาะการผ่อนผันที่ชัดเจนเท่านั้น เช่น คำสั่ง **#pragma** จะถูกใช้ ระดับนี้จะดำเนินการโดยไม่มีไพพ์ไลน์ของซอฟต์แวร์ ไม่มีการวนลูป หรือการคาดเดา และยังมีข้อจำกัด เกี่ยวกับจำนวนของหน่วยความจำที่คอมไพลเลอร์สามารถนำมาใช้ได้

-O3 แฟล็กนี้จะสั่งให้คอมไพลเลอร์ใช้เทคนิคการ optimization และใช้เป็นหน่วยความจำที่มากเท่าที่จำเป็นสำหรับการoptimization สูงสุด ระดับของการ optimization นี้อาจส่งผลถึงการเปลี่ยนแปลงการทำงานกับโปรแกรม หากโปรแกรมคำนึงถึงข้อยกเว้นของการอ้างอิงดัชนี เครื่องหมาย zero หรือผลกระทบเกี่ยวกับความแม่นยำของการ

คำนวณการเรียงลำดับใหม่ ผลกระทบเหล่านี้ สามารถหลีกเลี่ยงได้ที่ต้นทุนของผลการทำงานบางอย่างโดยใช้อ็อพชัน **-qstrict** ร่วมกับ **-O3** อ็อพชัน **-qhot** ที่ใช้ร่วมกับ **-O3** จะเปิดใช้งานการคาดเดาทั่วไปและการไม่วนลูป ผลลัพธ์ของการเปลี่ยนแปลงเหล่านี้คือ รูทีนที่ซับซ้อนและมีขนาดใหญ่ ซึ่งควรมีผลการทำงานที่เหมือนกันหรือดีกว่าอ็อพชัน **-O3** (ในการเชื่อมกับ **-qstrict** หรือ **-qhot**) ซึ่งมีอ็อพชัน **-O** ในเวอร์ชันก่อนหน้านี้ของคอมไพลเลอร์

-O4 แฟล็กนี้เทียบเท่ากับ **-O3 -qipa** พร้อมด้วยการสร้างสถาปัตยกรรมแบบอัตโนมัติ และการปรับอ็อพชันที่เกี่ยวข้องกับแพล็ตฟอร์มนั้น

-O5 แฟล็กนี้คล้ายกับ **-O4** ยกเว้นในกรณีของ **-qipa = level = 2**

การรวบรวมแพล็ตฟอร์มของฮาร์ดแวร์ที่ระบุเฉพาะ

มีหลายๆ สิ่งที่คุณควรพิจารณาก่อนที่จะรวบรวม แพล็ตฟอร์มฮาร์ดแวร์ที่ระบุเฉพาะ

ระบบสามารถใช้ชนิดต่างๆ ของตัวประมวลผลได้ โดยการใช้อ็อพชัน **-qarch** และ **-qtune** คุณสามารถออปติไมซ์โปรแกรมสำหรับคำสั่งพิเศษ และความแรงของตัวประมวลผลเหล่านี้

ให้ปฏิบัติตามแนวทางเหล่านี้:

- ถ้าโปรแกรมของคุณรันเฉพาะบนระบบเดี่ยว หรือบนกลุ่มของระบบที่มีชนิดของตัวประมวลผลที่เหมือนกัน ให้ใช้อ็อพชัน **-qarch** เพื่อระบุชนิดของตัวประมวลผล
- ถ้าโปรแกรมของคุณจะรันอยู่บนระบบที่มีชนิดของตัวประมวลผลที่ต่างกัน และคุณสามารถระบุชนิดของตัวประมวลผลหนึ่งชนิดเป็นชนิดที่สำคัญที่สุด ให้ใช้ค่าติดตั้ง **-qarch** และ **-qtune** ที่เหมาะสม ผู้ใช้ FORTRAN และ HPF สามารถใช้คำสั่ง **xxlf** และ **xxlhpf** เพื่อเลือกค่าติดตั้งเหล่านี้แบบโต้ตอบ
- ถ้าโปรแกรมของคุณมีแนวโน้มที่จะรันอยู่บนช่วงเต็มของการนำตัวประมวลผลไปใช้งาน และไม่มีแนวโน้มสำหรับการใช้ชนิดหลักของตัวประมวลผลหนึ่งชนิด ห้ามใช้ **-qarch** หรือ **-qtune**

อ็อพชัน C สำหรับผลการทำงานของรูทีนย่อย string.h

ระบบปฏิบัติการจะแสดงความสามาราถในการฝังรูทีนย่อย string ลงในแอ็พพลิเคชันโปรแกรม แทนการใช้รูทีนย่อยจาก `libc.a` เพื่อประหยัดการเรียกและส่งคืนเวลาในการเชื่อมต่อ

หากต้องการฝังรูทีนย่อย string ในซอร์สโค้ดของแอ็พพลิเคชันต้องมีคำสั่งต่อไปนี้ ก่อนที่จะใช้รูทีนย่อย:

```
#include <string.h>
```

ลักษณะการโค้ด C และ C++ สำหรับผลการทำงานที่ดีที่สุด

ในหลายๆ กรณี ต้นทุนของผลการทำงานของโครงสร้าง C จะไม่เด่นชัด และบางครั้งจะเป็นตัวนับ

สถานการณ์เหล่านี้บางสถานการณ์มีดังต่อไปนี้:

- เมื่อมีความเป็นไปได้ ให้ใช้ *int* แทน *char* หรือ *short*

  ในหลายๆ กรณี หน่วยข้อมูล *char* และ *short* จะใช้คำสั่งเพิ่มเติม ในการจัดการ คำสั่งพิเศษนี้จะมีต้นทุนทางด้านเวลา ยกเว้นในอาร์เรย์ขนาดใหญ่ และพื้นที่ใดๆ ที่ถูกบันทึกไว้โดยการใช้ชนิดข้อมูลที่เล็กกว่าที่มีมากกว่าอ็อฟเซ็ต โดยขนาดที่เพิ่มขึ้นของโปรแกรมเรียกทำงาน
- ถ้าคุณได้ใช้ *char* ให้ทำ *unsigned* หากเป็นไปได้

  อักขระที่ลงนามแล้ว จะใช้คำสั่งอื่นๆ อีกสองคำสั่งที่มากกว่า *อักขระที่ไม่ได้ลงนาม* ในแต่ละครั้งที่ตัวแปรถูกโหลดในเรจิสเตอร์

- ใช้ตัวแปรโลคัล (แบบอัตโนมัติ) แทนการใช้ตัวแปรแบบโกลบอล เมื่อสามารถกระทำได้

  ตัวแปรโกลบอลต้องการคำสั่งเพิ่มเติมเพื่อเข้าถึงตัวแปรโลคัล ได้ดีกว่า และ ในข้อมูลที่ไม่มีอยู่ที่มีความขัดแย้งกัน คอมไพลเลอร์จะสมมุติว่า ตัวแปรโกลบอลใดๆ ได้ถูกเปลี่ยนแปลงโดยการเรียกรูทีนย่อย การเปลี่ยนแปลงนี้ไม่ผลตรงข้ามกับการทำให้เหมาะสม เนื่องจาก ค่าตัวแปรใดๆ ที่ใช้หลังจากเรียกรูทีนย่อยจะถูกโหลดใหม่
- เมื่อมีความจำเป็นในการเข้าถึงตัวแปรโกลบอล (ซึ่งไม่ได้แบ่งใช้กับเธรดอื่นๆ) ให้ทำสำเนาของค่าตัวแปรโลคัลและใช้สำเนานั้น

  เว้นเสียแต่ตัวแปรโกลบอล ได้ถูกเข้าถึงเพียงครั้งเดียว ซึ่งมีสิทธิภาพในการทำงานสูง เพื่อใช้สำเนาโลคัล
- ใช้โค้ดไบรารีแทนสตริงเพื่อเร็กคอร์ดและทดสอบสถานการณ์ สตริงที่ใช้ทั้งพื้นที่ข้อมูลและพื้นที่คำสั่ง ตัวอย่างเช่น ลำดับต่อไปนี้:

  ```
  #define situation_1 1
  #define situation_2 2
  #define situation_3 3
  int situation_val;

  situation_val = situation_2;
  . . .
  if (situation_val == situation_1)
  . . .
  ```

  มีประสิทธิภาพมากกว่าลำดับต่อไปนี้:

  ```
  char situation_val[20];

  strcpy(situation_val,"situation_2");
  . . .
  if ((strcmp(situation_val,"situation_1"))==0)
  . . .
  ```
- เมื่อสตริงมีความจำเป็นต้องมี ให้ใช้สตริงที่มีความยาวคงที่แทนสตริงความยาวตัวแปรที่ยกเลิกค่า null

  ตระกูล **mem*( )** ของรูทีน เช่น **memcpy( )** จะเร็วกว่ารูทีน **str*( )** ที่สอดคล้องกัน เช่น **strcpy( )** เนื่องจากรูทีน **str*( )** ต้องตรวจสอบไบต์แต่ละตัวสำหรับ null และรูทีน **mem*( )** ไม่ต้อง

เวลาในการทำงานของคอมไพลเลอร์

มีหลายปัจจัยที่ส่งผลถึงเวลาในการทำงานของ คอมไพลเลอร์

ในระบบปฏิบัติการ คอมไพลเลอร์ C สามารถเรียกใช้ได้ด้วยคำสั่งสองคำสั่งที่ต่างกัน คือ: **cc** และ **xlc** คำสั่ง **cc** จะถูกใช้เพื่อเรียกใช้งานคอมไพลเลอร์ C ของระบบ ซึ่งเป็นสาเหตุทำให้คอมไพลเลอร์ C รันอยู่ในโหมด **langlevel=extended** โหมดนี้ อนุญาตให้ใช้การคอมไพล์ของโปรแกรม C ที่มีอยู่ซึ่งไม่ใช่การเข้ากันได้กับ ANSI และยังใช้ เวลาของตัวประมวลผลด้วย

ถ้าโปรแกรมที่คอมไพล์คือ การเข้ากันได้กับ ANSI โปรแกรมนั้นจะมีประสิทธิภาพมากขึ้น ในการเรียกคอมไพล์เลอร์ C ด้วยคำสั่ง **xlc**

การใช้แฟล็ก **-O3** ทางอ้อมจะสอดแทรกอ็อพชัน **-qmaxmem** ไว้ อ็อพชันนี้อนุญาตให้คอมไพลเลอร์ใช้เป็นหน่วยความจำที่มากเท่าที่จำเป็นสำหรับ การใช้ประโยชน์สูงสุด สถานการณ์นี้อาจมีผลกระทบสองอย่างคือ:

- สำหรับระบบหลายผู้ใช้ การคอมไพล์ **-O3** อาจใช้หน่วยความจำที่เพียงพอ เพื่อให้มีผลกระทบกับผลการทำงานที่ให้ประสบการณ์กับผู้ใช้อื่น

- สำหรับระบบที่มีหน่วยความจำที่ใช้จริงเพียงเล็กน้อย การคอมไพล์ **-O3** ขนาดใหญ่อาจใช้หน่วยความจำที่เพียงพอเพื่อทำให้อัตราการเพจสูง และการคอมไพล์ช้าลง

โปรแกรมที่จำกัดหน่วยความจำ

หากโปรแกรมเมอร์คุ้นเคยกับความพยายามในการจำกัดการกำหนดแอดเดรส ในสภาวะแวดล้อมแบบ DOS เซ็กเมนต์หน่วยความจำเสมือนขนาด 256 MB ดูเหมือนจะเพียงพอเพื่อทำให้มีประสิทธิผล โปรแกรมเมอร์จะทดสอบเพื่อละเว้นข้อจำกัดหน่วยเก็บ และโค้ดสำหรับความยาวพาธน้อยที่สุด และง่ายที่สุด แต่น่าเสียดาย ที่มีข้อเสียเปรียบในทัศนคตินี้

หน่วยความจำเสมือนมีขนาดใหญ่ แต่มีความเร็วที่ผันแปร คุณใช้หน่วยความจำมากเท่าใด ก็จะพบกับความช้ามากเท่านั้น และความสัมพันธ์จะไม่ใช่ในเชิงเส้น ตราบเท่าที่จำนวนทั้งหมดของหน่วยเก็บเสมือน จะถูกสัมผัสโดยโปรแกรมทั้งหมด (นั่นคือ ผลรวมของชุดการทำงาน) จะมีค่าน้อยกว่าจำนวนของหน่วยความจำที่ใช้จริงที่ไม่ได้ตรึงไว้ในเครื่อง หน่วยความจำเสมือนจะดำเนินการด้วยความเร็วของหน่วยความจำที่ใช้จริง เนื่องจากผลรวมของชุดการทำงานของโปรแกรมที่เรียกทำงานทั้งหมดจะส่งผ่านจำนวนกรอบของเพจ ที่พร้อมใช้งาน ผลการทำงานของหน่วยความจำจะลดระดับลงอย่างรวดเร็ว (ถ้าการควบคุมโหลดของหน่วยความจำ VMM ถูกปิดใช้งาน) ตามการเรียงลำดับของความสำคัญมากสุดสองแบบ เมื่อระบบเข้าใกล้จุดนี้ ระบบจะเรียกว่า *การ thrash* ซึ่งจะใช้เวลาในการเพจทั้งหมด และไม่มีประโยชน์ในเรื่องขงการทำงานที่ต้องทำ เนื่องจากการประมวลผลกำลังพยายามดึงการประมวลผลหน่วยเก็บที่จำเป็นกลับมาใช้ เพื่อให้เหมาะสมกับชุดการทำงาน ถ้าการควบคุมโหลดของหน่วยความจำ VMM แอ็คทีฟอยู่ การควบคุมนั้นสามารถหลีกเลี่ยงการ thrash แบบถาวรด้วยตนเอง แต่ในเรื่องของต้นทุน จะเป็นการเพิ่มเวลาตอบสนอง

การลดระดับเนื่องจากการใช้หน่วยความที่ไม่มีประสิทธิภาพจะมีค่ามากกว่า การใช้แคชที่ไม่มีประสิทธิภาพ เนื่องจากความแตกต่างในเรื่องของความเร็วระหว่าง หน่วยความจำและดิสก์จะมีค่าที่สูงกว่าความเร็วระหว่างแคชและหน่วยความจำ ตำแหน่งที่แคชหายไปสามารถใช้วงรอบ CPU จำนวนมากได้ ข้อบกพร่องของเพจ ใช้เวลา 10 มิลลิวินาทีหรือมากกว่านั้น ซึ่งน้อยกว่าวงรอบ 400 000 CPU

แม้ว่าการควบคุมโหลดของหน่วยความจำ VMM สามารถทำให้มั่นใจว่า สถานการณ์ของการ thrash ที่เพิ่งจะเริ่มต้นขึ้นไม่ได้เป็นการเริ่มต้นแบบถาวรด้วยตนเอง ข้อบกพร่องของเพจที่ไม่จำเป็นยังคงเป็นต้นทุน ในเรื่องของเวลาในการตอบสนองที่ลดระดับลงและทรูพุตที่ลดลง (โปรดดู “การปรับการควบคุมโหลดหน่วยความจำ VMM ด้วยคำสั่ง schedo” ในหน้า 162)

โครงสร้างรหัสที่เพจได้:

เพื่อลดชุดทำงานรหัสของโปรแกรมให้เหลือน้อยที่สุด วัตถุประสงค์ทั่วไปคือ การรวมรหัสที่ดำเนินการบ่อยเข้าไว้ในพื้นที่ขนาดเล็ก ที่แยกต่างหากจาก รหัสที่ดำเนินการไม่บ่อยนัก

รายละเอียดคือ:

- อย่าวางบล็อกที่ยาวของรหัสการจัดการข้อผิดพลาดไว้ในบรรทัด ให้วางในรูทีนย่อย แยกต่างหาก ถ้าเป็นไปได้ ควรวางไว้ในซอร์สโค้ดโมดูลแยกเฉพาะ การดำเนินการนี้ไม่ได้ใช้กับ พาธข้อผิดพลาดเพียงอย่างเดียวเท่านั้น หากแต่ยังใช้กับอ็อพชันของฟังก์ชันใดๆ ที่ไม่ได้ใช้บ่อยด้วย
- อย่าจัดโครงสร้างโหลดโมดูลตามอำเภอใจ พยายามตรวจสอบให้แน่ใจว่าอ็อบเจ็กต์โมดูล ที่เรียกใช้บ่อยมีการวางตำแหน่งไว้ใกล้กับผู้เรียกของโมดูลนั้นให้มากที่สุดเท่าที่เป็นไปได้ อ็อบเจ็กต์โมดูล ที่ประกอบด้วย (ดีที่สุด) รูทีนย่อยซึ่งไม่ได้เรียกใช้บ่อยนักควรมีการวางไว้ใกล้กัน ที่ตอนท้ายของโหลดโมดูล หน้าที่มีอ็อบเจ็กต์โมดูลดังกล่าวแทบจะ ไม่มีการอ่าน

โครงสร้างข้อมูลที่เพจได้:

เพื่อลดชุดทำงานข้อมูลให้เหลือน้อยที่สุด ให้พยายามเน้นข้อมูลที่ใช้บ่อย และหลีกเลี่ยงการอ้างอิงถึงหน้าหน่วยเก็บเสมือนที่ไม่จำเป็น

รายละเอียดคือ:

- ใช้รูทีนย่อย **malloc( )** หรือ **calloc( )** เพื่อร้องขอพื้นที่ว่างในจำนวน ที่คุณต้องการจริงเท่านั้น อย่าร้องขอ แล้วเริ่มต้นอาร์เรย์ขนาดสูงสุด เมื่อสถานการณ์จริงใช้พื้นที่เพียงแค่เศษเสี้ยวของอาร์เรย์เท่านั้น เมื่อคุณเรียกใช้ หน้าใหม่เพื่อเริ่มต้นองค์ประกอบอาร์เรย์ นั่นหมายถึงคุณกำลังบังคับให้ VMM ขโมยหน้าหน่วยความจริงจากบุคคลอื่น ในภายหลัง จะส่งผลให้เกิด page fault เมื่อกระบวนการที่เป็นเจ้าของหน้านั้นพยายามเข้าถึงหน้านั้นอีกครั้งหนึ่ง ความแตกต่าง ระหว่างรูทีนย่อย **malloc( )** และ **calloc( )** ไม่ใช่เพียงแต่ ในอินเตอร์เฟสเท่านั้น
- เนื่องจากรูทีนย่อย **calloc( )** มีหน่วยเก็บที่จัดสรรเป็นศูนย์ ดังนั้นจึง เรียกใช้หน้าทุกหน้าที่มีการจัดสรร ในขณะที่รูทีนย่อย **malloc( )** เรียกใช้เฉพาะหน้าแรกเท่านั้น ถ้าคุณใช้รูทีนย่อย **calloc( )** เพื่อจัดสรร พื้นที่ขนาดใหญ่ แล้วใช้พื้นที่ดังกล่าวแค่เพียงเล็กน้อยเมื่อเริ่มต้น นั่นหมายความว่าคุณเพิ่มโหลดที่ไม่จำเป็นบนระบบ ไม่เฉพาะแต่หน้าเท่านั้นที่มีการเริ่มต้น ถ้ามีการเรียกร้องหน่วยความจำจริงของหน้าเหล่านั้นใหม่ หน้าที่เริ่มต้นแล้ว และไม่เคยมีการใช้ต้องถูกบันทึกออกไปยังพื้นที่ว่างการเพจ กรณีนี้ทำให้สิ้นเปลือง ทั้ง I/O และสล็อตพื้นที่ว่างการเพจ
- รายการที่ลิงก์ของโครงสร้างขนาดใหญ่ (เช่น บัฟเฟอร์) อาจก่อให้เกิดปัญหา คล้ายกัน ถ้าโปรแกรมของคุณทำการค้นหาคีย์เฉพาะตามห่วงโซ่ที่กำหนดจำนวนมาก ให้พิจารณาการจัดทำลิงก์และคีย์แยกต่างหากจากข้อมูล หรือการใช้ แนวทาง hash-table แทน
- ความเป็นโลคัลของการอ้างอิงหมายถึงความเป็นโลคัลในด้านเวลา ไม่เฉพาะแต่ในพื้นที่ว่างที่อยู่เท่านั้น ควรเริ่มต้นโครงสร้างข้อมูลก่อนหน้าจะใช้เพียงเล็กน้อยเท่านั้น (หรือในเวลาที่ใช้จริง) ในระบบ ที่มีโหลดอย่างหนัก โครงสร้างข้อมูลที่มีอยู่เป็นเวลานาน ตั้งแต่การเริ่มต้นจนถึงการใช้ อาจเสี่ยงต่อการถูกขโมยเฟรมได้ โปรแกรมของคุณ อาจประสบกับ page fault ที่ไม่จำเป็นเมื่อเริ่มต้นใช้โครงสร้าง ข้อมูล
- ในลักษณะคล้ายกัน ถ้าใช้โครงสร้างขนาดใหญ่ตั้งแต่เนิ่นๆ แล้วปล่อยทิ้งไว้ไม่ใช้ งานในเวลาที่เหลือของโปรแกรม ควรจะรีลีสโครงสร้างนั้น ไม่เพียงพอที่จะใช้รูทีนย่อย **free( )** เพื่อปล่อยพื้นที่ว่างซึ่งถูกจัดสรรด้วยรูทีนย่อย **malloc( )** หรือ **calloc( )** เนื่องจากรูทีนย่อย **free( )** รีลีสเฉพาะช่วงที่อยู่ของโครงสร้างนั้นเท่านั้น หากต้องการรีลีสหน่วยความจำจริง และพื้นที่ว่างการเพจ ให้ใช้รูทีนย่อย **disclaim( )** เพื่อปล่อย พื้นที่ว่างด้วย การเรียก **disclaim( )** ควรดำเนินการก่อนการเรียก **free( )**

การใช้ในทางที่ผิดของหน่วยเก็บที่ตรึงไว้:

หากต้องการหลีกเลี่ยงการทำงานเป็นวงกลมและการหมดเวลาใช้งาน เศษส่วนขนาดเล็กของระบบ ต้องถูกตรึงไว้ในหน่วยความจำที่ใช้จริง

สำหรับโค้ดและข้อมูลนี้ แนวคิดของชุดการทำงานจะไม่มีความหมาย เนื่องจากข้อมูลที่ตรึงไว้ทั้งหมดจะอยู่ในหน่วยเก็บจริงตลอดเวลา ไม่ว่าจะใช้หรือไม่ก็ตาม โปรแกรมใดๆ (เช่น ไดร์เวอร์อุปกรณ์ที่เขียนโดยผู้ใช้) ที่ตรึงโค้ดหรือข้อมูลไว้ต้องได้รับการออกแบบด้วยความระมัดระวัง (หรือนำมาพิเคราะห์ ถ้าพอร์ตแล้ว) เพื่อมั่นใจว่า จำนวนต่ำสุดของหน่วยเก็บที่ตรึงไว้จะถูกนำมาใช้ ตัวอย่างคำเตือนบางส่วน มีดังต่อไปนี้:

- โค้ดจะถูกตรึงไว้บนโมดูลพื้นฐานที่โหลด (ไฟล์เรียกทำงาน) ถ้าคอมโพเนนต์มีโมดูลอ็อบเจ็กต์บางตัว ที่ต้องตรึงไว้ และโมดูลอ็อบเจ็กต์บางตัวที่สามารถเพจได้ แพ็กเกจที่มีโมดูลอ็อบเจ็กต์จะถูกตรึงไว้ในโมดูลที่โหลดแยกจากกัน
- การตรึงโมดูลหรือโครงสร้างข้อมูล เนื่องจากอาจมีปัญหา ที่ไม่สามารถรับผิดชอบได้ ตัวออกแบบควรเข้าใจเงื่อนไขภายใต้ข้อมูล ที่สามารถบังคับให้มีได้ และข้อบกพร่องของเพจที่สามารถเกิดขึ้นได้ ในจุดนั้น

- โครงสร้างที่ตรึงไว้ซึ่งมีขนาดที่ต้องการเป็นการพึ่งพาการโหลด เช่น พูลของบัฟเฟอร์ ควรสามารถปรับแต่งได้โดยผู้ดูแลระบบ

คำแนะนำการติดตั้งที่เกี่ยวข้องกับประสิทธิภาพ

มีหลายประเด็นที่ควรพิจารณาก่อนหน้าและในระหว่าง กระบวนการติดตั้ง

คำแนะนำการติดตั้งระบบปฏิบัติการล่วงหน้า

ต้องพิจารณาสองสถานการณ์ดังนี้:

- การติดตั้งระบบปฏิบัติการบนระบบใหม่
  ก่อนที่คุณจะเริ่มต้น กระบวนการติดตั้ง ต้องแน่ใจว่าคุณได้ตัดสินใจเกี่ยวกับขนาดและที่ตั้งของ ระบบดิสก์ไฟล์และพื้นที่ว่างการเพจ และคุณเข้าใจวิธีการสื่อสาร การตัดสินใจเหล่านั้นกับระบบปฏิบัติการ
- การติดตั้งระดับใหม่ของระบบปฏิบัติการบนระบบที่มีอยู่
  ถ้า คุณกำลังอัพเกรดเป็นระดับใหม่ของระบบปฏิบัติการ ให้ทำดังต่อไปนี้:
  - ตรวจสอบว่าคุณกำลังใช้ไฟล์ /etc/tunables/nextboot อยู่หรือไม่
  - ถ้าคุณไม่ได้ใช้ไฟล์ /etc/tunables/nextboot ให้ตรวจสอบไฟล์ /etc/tunables/lastboot.log โดยละเอียดหลังจาก รีบูตครั้งแรก

คำแนะนำในการติดตั้งไมโครโพรเซสเซอร์ล่วงหน้า

ใช้พารามิเตอร์การกำหนดตารางเวลาไมโครโพรเซสเซอร์ดีฟอลต์ เช่น ระยะเวลาในการแบ่งเวลา

นอกจากคุณมีประสบการณ์ในการมอนิเตอร์และการปรับอย่างกว้างขวาง ด้วยเวิร์กโหลดสำหรับคอนฟิกูเรชันเฉพาะ คุณไม่ควรเปลี่ยนพารามิเตอร์เหล่านี้ในขณะติดตั้ง

โปรดดู “ผลการทำงานของไมโครโพรเซสเซอร์” ในหน้า 112 สำหรับข้อแนะนำหลังการติดตั้ง

คำแนะนำในการติดตั้งหน่วยความจำล่วงหน้า

ห้ามเปลี่ยนแปลงหน่วยความจำ–threshold จนกระทั่งคุณมีประสบการณ์เกี่ยวกับ การตอบกลับของระบบไปยังเวิร์กโหลดที่เกิดขึ้นจริง

โปรดดู “ผลการทำงานของหน่วยความจำ” ในหน้า 139 สำหรับข้อแนะนำหลังการติดตั้ง

คำแนะนำในการติดตั้งดิสก์ล่วงหน้า

กลไกสำหรับการนิยามและการขยายโลจิคัลวอลุ่ม มีความพยายามในการทำตัวเลือกดีฟอลต์ที่เป็นไปได้มากที่สุด อย่างไรก็ตาม ผลการทำงานของดิสก์ I/O ที่น่าพึงพอใจจะมีเพิ่มมากขึ้น หากโปรแกรมติดตั้งของระบบ ปรับให้เหมาะสมกับขนาดและตำแหน่งของโลจิคัลวอลุ่มในหน่วยเก็บข้อมูล และข้อกำหนดเกี่ยวกับเวิร์กโหลด

ข้อแนะนำมีดังต่อไปนี้:

- หากเป็นไปได้ ค่าดีฟอลต์กลุ่มวอลุ่ม นั่นคือ rootvg ควรมีฟิสิคัลวอลุ่ม ซึ่งระบบได้ติดตั้งไว้ในครั้งแรก กำหนดกลุ่มวอลุ่มอื่นๆ ตั้งแต่หนึ่งกลุ่มขึ้นไป เพื่อควบคุมฟิสิคัลวอลุ่มอื่นในระบบ ข้อแนะนำนี้มีการจัดการระบบเช่นเดียวกับผลการทำงานที่มีผลประโยชน์
- ถ้ากลุ่มวอลุ่มประกอบด้วยฟิสิคัลวอลุ่มตั้งหนึ่งหนึ่งกลุ่มขึ้นไป คุณอาจได้รับผลการทำงานโดย :

- นิยามกลุ่มวอลุ่มในครั้งแรกด้วยฟิสิคัลวอลุ่มเดี่ยว
- นิยามโลจิคัลวอลุ่มภายในกลุ่มวอลุ่มใหม่ นิยามนี้ ทำให้การจัดสรรเจอร์นัลกลุ่มวอลุ่มของกลุ่มวอลุ่ม บนฟิสิคัลวอลุ่มแรก
- การเพิ่มฟิสิคัลวอลุ่มที่เหลืออยู่ให้กับกลุ่มวอลุ่ม
- นิยามระบบไฟล์ที่มีกิจกรรมสูงบนฟิสิคัลวอลุ่มที่เพิ่มใหม่
- นิยามระบบไฟล์ที่มีกิจกรรมต่ำมาก หากมี บนฟิสิคัลวอลุ่มที่มีเจอร์นัลโลจิคัลวอลุ่ม สิ่งนี้จะส่งผลกับผลการทำงานเท่านั้น หาก I/O เป็นสาเหตุของการทำรายการบันทึกการทำงานระบบไฟล์ที่เจอร์นับ (JFS)

  วิธีนี้ จะแยกกิจกรรม I/O ที่เจอร์เนลแล้วออกจาก I/O ข้อมูลกิจกรรมสูง การเพิ่มความเป็นไปได้ของการซ้อนทับ เทคนิคนี้สามารถมีผลกระทบกับผลการทำงานของเซิร์ฟเวอร์ NFS ได้ เนื่องจากทั้งข้อมูลและการเขียนเจอร์นัล ต้องเสร็จสิ้นก่อนที่สัญญาณ NFS ของ I/O จะเสร็จสิ้นสำหรับการดำเนินการเขียน

- ในโอกาสแรก ให้นิยามหรือขยายโลจิคัลวอลุ่ม เพื่อเพิ่มขนาดให้มากสุดตามที่คาดการณ์ไว้ หากต้องการเพิ่มความเป็นไปได้ที่ผลการทำงานที่สำคัญของโลจิคัลวอลุ่ม จะดำเนินการอย่างต่อเนื่อง และในตำแหน่งที่ต้องการ นิยามหรือขยายโลจิคัลวอลุ่มในครั้งแรก
- โลจิคัลวอลุ่มที่มีการใช้งานสูงควรใช้ส่วนของดิสก์ไดร์ฟจำนวนมาก ถ้าอ็อพชัน **RANGE ของฟิสิคัลวอลุ่ม** บนหน้าจอเพิ่มโลจิคัลวอลุ่ม ของโปรแกรม SMIT (วิธีลัด: **smitty mklv**) จะถูกตั้งค่า **สูงสุด** โลจิคัลวอลุ่มใหม่จะถูกแบ่งระหว่างฟิสิคัลวอลุ่มของกลุ่มวอลุ่ม (หรือ ชุดของฟิสิคัลวอลุ่มที่แสดงไว้ อย่างชัดเจน)
- ถ้าระบบมีไดร์ฟที่มีชนิดที่ต่างกัน (หรือคุณพยายามตัดสินเลือกไดร์ฟ ที่ต้องการเรียงลำดับ) ให้พิจารณาคำแนะนำต่อไปนี้:
  - วางขนาดของไฟล์ที่มีขนาดใหญ่ซึ่งเข้าถึงได้บนดิสก์ไดร์ฟที่มีความเร็วสูงสุด ที่พร้อมใช้งาน
  - ถ้าคุณคาดการณ์ถึงความถี่ของการแอ็กเซสตามลำดับกับไฟล์ขนาดใหญ่บนดิสก์ไดร์ฟ ที่มีความเร็วสูงสุด ให้จำกัดจำนวนของดิสก์ไดร์ฟต่อดิสก์อะแด็ปเตอร์
  - เมื่อเป็นไปได้ ให้พ่วงต่อไดร์ฟที่มีความสำคัญตามข้อกำหนดผลการทำงานที่มีวอลุ่มสูง กับอะแด็ปเตอร์ที่มีความเร็วสูง อะแด็ปเตอร์เหล่านี้มีคุณลักษณะ เช่น ความสามารถในการเขียนแบบ back-to-back ซึ่งจะไม่พร้อมใช้งานกับดิสก์อะแด็ปเตอร์อื่นๆ
  - บนดิสก์ไดร์ฟขนาดเล็ก โลจิคัลวอลุ่มที่จะพักลำดับไฟล์ที่เข้าถึงบ่อยซึ่งมีขนาดใหญ่ ควรถูกจัดสรรใน outer_edge ของฟิสิคัลวอลุ่ม ดิสก์เหล่านี้มีบล็อกจำนวนมากต่อแทร็กในส่วนนอก ซึ่งจะปรับปรุงผลการทำงานตามลำดับ
  - สำหรับ SCSI บัสรุ่นเก่า ไดร์ฟที่มีหมายเลขที่สูงสุด (หมายเลขที่มีแอดเดรส SCSI มากที่สุด ซึ่งตั้งค่าไว้บนฟิสิคัลไดร์ฟ) จะมีระดับความสำคัญสูงสุด ข้อกำหนดคุณสมบัติถัดมาจะพยายามรักษาความเข้ากันได้ ด้วยข้อกำหนดคุณสมบัติเดิม ดังนั้น ลำดับจากระดับความสำคัญสูงสุดไปยังระดับความสำคัญต่ำสุดเป็นดังนี้: 7-6-5-4-3-2-1-0-15-14-13-12-11-10-9-8

    ในสถานการณ์ส่วนใหญ่ ผลกระทบนี้จะไม่สามารถสังเกตเห็นได้ แต่การดำเนินการกับไฟล์ลำดับที่มีขนาดใหญ่ จะทราบถึงการแยกไดร์ฟที่มีหมายเลขต่ำออกจากการเข้าถึงบัส คุณควรปรับแต่งดิสก์ไดร์ฟที่พักข้อมูล ที่มีเวลาตอบสนองที่สำคัญมากที่แอดเดรสสูงสุดบน SCSI บัสแต่ละตัว

    คำสั่ง **lsdev -Cs scsi** จะรายงานเกี่ยวกับการกำหนดแอดเดรสปัจจุบันสำหรับ SCSI บัสแต่ละตัว สำหรับอะแด็ปเตอร์ SCSI รุ่นเก่า แอดเดรส SCSI คือหมายเลขเลขในคู่ของหมายเลขอันดับที่สี่ ในเอาต์พุต ในตัวอย่างเอาต์พุตต่อไปนี้ ดิสก์ 400 GB หนึ่งตัวจะอยู่ที่ SCSI แอดเดรส 4 ตัวอื่นจะอยู่ที่แอดเดรส 5 เทปไดร์ฟขนาด 8mm จะอยู่ที่แอดเดรส 1 และไดร์ฟ CDROM จะอยู่ที่แอดเดรส 3

    ```
    cd0    Available 10-80-00-3,0 SCSI Multimedia CD-ROM Drive
    hdisk0 Available 10-80-00-4,0 16 Bit SCSI Disk Drive
    hdisk1 Available 10-80-00-5,0 16 Bit SCSI Disk Drive
    rmt0   Available 10-80-00-1,0 2.3 GB 8mm Tape Drive
    ```

– ไฟล์ขนาดใหญ่ที่มีการใช้งานอย่างหนัก และโดยปกติแล้ว เข้าถึงแบบสุ่ม เช่น ฐานข้อมูล ต้องมีการกระจายบนฟิสิคัลวอลุ่ม สองตัวขึ้นไป

หลักการที่เกี่ยวข้อง:

"ผลการทำงานของโลจิคัลวอลุ่มและดิสก์ I/O" ในหน้า 195
หัวข้อนี้เน้นถึงผลการทำงานของโลจิคัลวอลุ่มและดิสก์ไดร์ฟที่พ่วงต่อ แบบโลคัล

การวางตำแหน่งของพื้นที่ว่างการเพจและขนาด:

ข้อแนะนำทั่วไปคือผลรวมของขนาดของพื้นที่ว่างการเพจควรจะเท่ากับ สองเท่าของขนาดหน่วยความจำจริงของเครื่องเป็นอย่างน้อย ขนาดหน่วยความจำจริง สูงสุดคือ 256 MB (512 MB ของพื้นที่ว่างการเพจ)

หมายเหตุ: สำหรับหน่วยความจำที่ใหญ่กว่า 256 MB มีข้อแนะนำดังต่อไปนี้:

```
พื้นที่ว่างการเพจทั้งหมด = 512 MB + (ขนาดหน่วยความจำ - 256 MB) * 1.25
```

อย่างไรก็ตาม ด้วย การจัดสรรพื้นที่ว่างเพจที่เลื่อนออกไป คำแนะนำนี้อาจใช้พื้นที่ว่างดิสก์มากกว่า จำนวนที่ต้องการจริง โปรดดู "การจัดสรรพื้นที่ว่างหน้า" ในหน้า 170 สำหรับข้อมูลเพิ่มเติม

สิ่งที่ดีที่สุดคือควรมีพื้นที่ว่างการเพจที่มีขนาดโดยคร่าวๆ เท่ากันหลายพื้นที่ โดย แต่ละพื้นที่อยู่บนฟิสิคัลดิสก์ไดร์ฟที่แตกต่างกัน ถ้าคุณตัดสินใจที่จะสร้างพื้นที่ว่างการเพจเพิ่มเติม ให้สร้างบนฟิสิคัลวอลุ่มที่โหลดน้อยกว่าฟิสิคัลวอลุ่ม ใน rootvg เมื่อจัดสรรบล็อกพื้นที่ว่างการเพจ VMM จะจัดสรรสี่บล็อกหมุนเวียนจากแต่ละพื้นที่ว่างการเพจที่ใช้งานอยู่ซึ่งมี พื้นที่ว่าง ในขณะที่ระบบกำลังบูต เฉพาะพื้นที่ว่างการเพจหลัก (hd6) เท่านั้นที่ใช้งานอยู่ ในเวลาต่อมา บล็อกพื้นที่ว่างการเพจทั้งหมดที่จัดสรรในระหว่างบูต จะอยู่บนพื้นที่ว่างการเพจหลัก ซึ่งหมายความว่าพื้นที่ว่างการเพจหลัก ควรมีขนาดใหญ่กว่าพื้นที่ว่างการเพจรองพอประมาณ พื้นที่ว่างการเพจรองทั้งหมด ควรมีขนาดเท่ากันเพื่อให้มั่นใจว่าขั้นตอนวิธีที่ดำเนินการตามรอบ สามารถทำงานได้อย่างมีประสิทธิภาพ

คำสั่ง **lsps -a** ให้ภาพถ่ายของระดับการใช้ประโยชน์ ปัจจุบันของพื้นที่ว่างการเพจทั้งหมดบนระบบหนึ่ง คุณยังสามารถใช้รูทีนย่อย **psdanger( )** เพื่อกำหนดว่าการใช้ประโยชน์พื้นที่ว่างการเพจเข้าใกล้ระดับ วิกฤตมากน้อยเพียงใด ดังเช่นตัวอย่าง โปรแกรมต่อไปนี้จะใช้รูทีนย่อย **psdanger( )** เพื่อแสดงข้อความคำเตือนเมื่อมีการใช้งานเกินกว่า threshold :

```
/* psmonitor.c
  Monitors system for paging space low conditions. When the condition is
  detected, writes a message to stderr.
  Usage:    psmonitor [Interval [Count]]
  Default:  psmonitor 1 1000000
*/
#include <stdio.h>
#include <signal.h>
main(int argc,char **argv)
{
  int interval = 1;        /* seconds */
  int count = 1000000;     /* intervals */
  int current;             /* interval */
  int last;                /* check */
  int kill_offset;         /* returned by psdanger() */
  int danger_offset;       /* returned by psdanger() */


  /* are there any parameters at all? */
  if (argc > 1) {
```

```
    if ( (interval = atoi(argv[1])) < 1 ) {
      fprintf(stderr,"Usage: psmonitor [ interval [ count ] ]\n");
      exit(1);
    }
    if (argc > 2) {
      if ( (count = atoi( argv[2])) < 1 ) {
         fprintf(stderr,"Usage: psmonitor [ interval [ count ] ]\n");
         exit(1);
      }
    }
  }
  last = count -1;
  for(current = 0; current < count; current++) {
    kill_offset = psdanger(SIGKILL); /* check for out of paging space */
    if (kill_offset < 0)
      fprintf(stderr,
        "OUT OF PAGING SPACE! %d blocks beyond SIGKILL threshold.\n",
        kill_offset*(-1));
    else {
      danger_offset = psdanger(SIGDANGER); /* check for paging space low */
      if (danger_offset < 0) {
        fprintf(stderr,
          "WARNING: paging space low. %d blocks beyond SIGDANGER threshold.\n",
          danger_offset*(-1));
        fprintf(stderr,
          "                          %d blocks below SIGKILL threshold.\n",
          kill_offset);
      }
    }
      if (current < last)
        sleep(interval);
  }
}
```

ความหมายของประสิทธิภาพการทำมิเรอร์ดิสก์:

จากจุดยืนด้านประสิทธิภาพ การทำมิเรอร์ต้องเสียค่าใช้จ่ายสูง การทำมิเรอร์ ที่มีการตรวจสอบการบันทึกยิ่งต้องเสียค่าใช้จ่ายมากขึ้น (การหมุนเวียนดิสก์พิเศษต่อการบันทึก) และการทำมิเรอร์ ที่มีทั้งการตรวจสอบการบันทึกและความสม่ำเสมอของ Mirror Write ต้องเสียค่าใช้จ่ายมากที่สุด (การหมุนเวียน ดิสก์บวกกับการค้นหา Cylinder 0)

ถ้าใช้การทำมิเรอร์อยู่และความสม่ำเสมอของ Mirror Write เปิดอยู่ (ตามค่า ดีฟอลต์) ให้พิจารณาการระบุตำแหน่งสำเนาในพื้นที่ด้านนอกของดิสก์ เนื่องจากข้อมูลความสม่ำเสมอของ Mirror Write จะถูกบันทึกลงใน Cylinder 0 เสมอ จากจุดยืนด้านการเงิน การทำมิเรอร์ที่มีการบันทึกเพียงอย่างเดียวเป็นสิ่งที่แพง แม้ว่าโดยปกติแล้ว คำสั่ง **lslv** จะแสดงความสม่ำเสมอของ Mirror Write เป็นเปิด สำหรับโลจิคัลวอลุ่มที่ไม่ได้มิเรอร์ แต่กระบวนการจริงไม่ได้เกิดขึ้นยกเว้นว่า ค่า COPIES มากกว่าหนึ่ง การตรวจสอบการบันทึกมีค่าดีฟอลต์เป็นปิด เนื่องจากมีความหมาย (และค่าใช้จ่าย) สำหรับโลจิคัลวอลุ่มที่ไม่ได้มิเรอร์

มีอ็อพชันความสม่ำเสมอของ mirror write ที่เรียกว่า Passive Mirror Write Consistency (MWC) กลไกดีฟอลต์สำหรับการทำให้มั่นใจว่าความสอดคล้องของ mirror write เป็น MWC ที่ใช้งานอยู่ MWC ที่ใช้งานอยู่นำเสนอการกู้คืนแบบด่วนเมื่อเวลารีบูตหลังจากเกิด ความเสียหาย อย่างไรก็ตาม ประโยชน์นี้ต้องแลกกับประสิทธิภาพการบันทึกที่ด้อยลง โดยเฉพาะในกรณีของการบันทึกแบบสุ่ม การปิดใช้งาน MWC ที่ใช้งานอยู่ ช่วยตัดปัญหาประสิทธิภาพการบันทึกที่ด้อยลงนี้ได้ แต่เมื่อรีบูตหลังจาก

ความเสียหาย คุณต้องใช้คำสั่ง **syncvg –f** เพื่อซิงโครไนซ์กลุ่มวอลุ่มทั้งกลุ่ม ด้วยตนเองก่อนที่ผู้ใช้จะสามารถเข้าถึงกลุ่มวอลุ่มนั้นได้ เพื่อให้สำเร็จตามนี้ ต้องปิดใช้งาน `automatic vary-on` ของกลุ่มวอลุ่ม

การเปิดใช้งาน Passive MWC ไม่ได้เพียงแต่ตัดปัญหาประสิทธิภาพการบันทึก ที่เชื่อมโยงกับ MWC ที่ใช้งานอยู่เท่านั้น แต่โลจิคัลวอลุ่มจะถูกรีซิงโครไนซ์โดย อัตโนมัติในขณะกำลังเข้าถึงพาร์ติชันด้วย นี่หมายความว่าผู้ดูแลระบบไม่ต้อง ซิงโครไนซ์โลจิคัลวอลุ่มด้วยตนเอง หรือไม่ต้องปิดใช้งาน `automatic vary-on` ข้อเสียของ Passive MWC คือการดำเนินงานอ่านอาจช้าลงจนกว่าจะรีซิงโครไนซ์พาร์ติชันทั้งหมดแล้ว

คุณสามารถเลือกอ็อพชันความสม่ำเสมอของ mirror write ภายใน SMIT ในขณะที่สร้างหรือเปลี่ยนโลจิคัลวอลุ่ม อย่างใดอย่างหนึ่ง อ็อพชันที่เลือกมีผลบังคับใช้เฉพาะถ้าโลจิคัลวอลุ่มมีการทำมิเตอร์เท่านั้น (สำเนา > 1)

ความหมายของประสิทธิภาพ LVs ที่มิเรอร์และ striped:

การทำมิเรอร์และ striping โลจิคัลวอลุ่มรวมการมีอยู่ของข้อมูล RAID 1 เข้ากับประสิทธิภาพ RAID 0 โดยทั้งหมดผ่านทางซอฟต์แวร์

โลจิคัลวอลุ่มไม่สามารถมิรเรอร์และ striped ใน เวลาเดียวกัน กลุ่มวอลุ่มที่มีโลจิคัลวอลุ่มซึ่ง striped และมิรเรอร์ไม่สามารถอิมพอร์ตเข้าใน AIX

คำแนะนำในการติดตั้งการสื่อสารล่วงหน้า

สำหรับการหาตำแหน่งที่ถูกต้องของอะแด็ปเตอร์และคำแนะนำเกี่ยวกับผลการทำงานที่หลากหลาย โปรดดู *PCI Adapter Placement Reference*

โปรดดูสรุปของข้อแนะนำในการปรับการสื่อสารใน “การปรับประสิทธิภาพ TCP และ UDP” ในหน้า 282 และ “การปรับประสิทธิภาพพูล mbuf” ในหน้า 319

ระบบที่ใช้ POWER4

มีประเด็นประสิทธิภาพหลายอย่างที่เกี่ยวข้องกับเซิร์ฟเวอร์ที่ใช้ POWER4

หากต้องการข้อมูลที่เกี่ยวข้อง ให้ดูคู่มือ “ผลการทำงานของระบบไฟล์” ในหน้า 257, “การจัดการรีซอร์ส” ในหน้า 43, และ IBM Redbooks® *บทนำและคู่มือการปรับตัวประมวลผล POWER4*

POWER4การพัฒนาประสิทธิภาพ

POWER4 microprocessor มีการพัฒนาประสิทธิภาพหลายอย่าง

- นับเป็นสิ่งที่ดีที่สุดสำหรับ symmetric multiprocessing (SMP) ดังนั้นจึงให้รูปแบบขนานสำหรับคำสั่งที่ดีขึ้น
- ผลิตภัณฑ์นี้มีการจัดตารางเวลาที่ดีขึ้นสำหรับคำสั่งและ prefetching ข้อมูล และ กลไกการคาดการณ์สาขาที่มีประสิทธิภาพมากขึ้น
- มีแบนด์วิธหน่วยความจำที่สูงกว่า POWER3 microprocessor และออกแบบมาเพื่อทำงานในสภาพความถี่ที่สูงขึ้นมาก

การเปรียบเทียบไมโครโพรเซสเซอร์

ตารางต่อไปนี้จะเปรียบเทียบลักษณะมุมมองหลักของความแตกต่างระหว่างไมโครโพรเซสเซอร์ IBM

ตารางที่ 1. การเปรียบเทียบตัวประมวลผล

	POWER3	RS64	POWER4
ความถี่	450 MHz	750 MHz	> 1 GHz
หน่วยแบบไม่อิงดัชนี	3	2	2
หน่วยแบบอิงดัชนี	2	1	2
หน่วยการโหลด/เก็บ	2	1	2
หน่วยการแบรนช์/อื่นๆ	1	1	2
ความกว้างของการจัดส่ง	4	4	5
การคาดการณ์แบรนช์	ไดนามิก	สแตตกิ	ไดนามิก
ขนาด I-cache	32 KB	128 KB	64 KB
ขนาด D-cache	128 KB	128 KB	32 KB
ขนาด L2-cache	1, 4, 8 MB	2, 4, 8, 16 MB	1.44
ขนาด L3-cache	N/A	N/A	วัดด้วยจำนวนของตัวประมวลผล
การเรียกข้อมูลก่อน	ใช่	ไม่ใช่	ใช่

การพัฒนาการปรับสเกลได้ของระบบบน POWER4

ในระบบบน POWER4 ระบบปฏิบัติการมี ข้อดีในการปรับสเกลได้หลายอย่างเมื่อเปรียบเทียบกับระบบก่อนหน้านี้ ทั้งในแง่ของเวิร์กโหลดและประสิทธิภาพ

การปรับสเกลได้ของเวิร์กโหลด หมายถึงความสามารถในการจัดการกับ แอ็พพลิเคชันเวิร์กโหลดที่เพิ่มมากขึ้น *การปรับสเกลได้ของประสิทธิภาพ* หมายถึงการรักษาระดับ ประสิทธิภาพที่ยอมรับได้ เมื่อรีซอร์สซอฟต์แวร์เพิ่มขึ้นเพื่อตอบสนองต่อ ความต้องการของเวิร์กโหลดที่ใหญ่ขึ้น

ข้อมูลต่อไปนี้เป็นการเปลี่ยนแปลงการปรับสเกลได้ที่สำคัญที่สุดบางส่วน

Pinned หน่วยความจำที่แบ่งใช้สำหรับฐานข้อมูล

AIX ช่วยให้สามารถเก็บรักษาเพจหน่วยความจำไว้ ในหน่วยความจำจริงได้ตลอดเวลา กลไกนี้ เรียกว่า *pinning หน่วยความจำ*

Pinning พื้นที่หน่วยความจำห้ามไม่ให้เพจเจอร์ขโมยหน้าจากหน้า ที่กลับไปยังพื้นที่หน่วยความจำที่ pinned

การสนับสนุนหน่วยความจำขนาดใหญ่กว่า

ขนาดของหน่วยความจำจริงสูงสุดที่สนับสนุนโดยเคอร์เนล 64 บิต ขึ้นอยู่กับระบบฮาร์ดแวร์

ขนาดนี้จะอิงตามข้อกำหนดหน่วยความจำที่ใช้จริง ณ เวลาบูตของฮาร์ดแวร์ของระบบ และคอนฟิกูเรชัน I/O ที่เป็นไปได้ ซึ่งสนับสนุนโดยเคอร์เนลแบบ 64 บิต ไม่มีข้อกำหนดของขนาดพื้นที่การเพจต่ำสุดสำหรับเคอร์เนลแบบ 64 บิต

เคอร์เนลแบบ 64 บิต

ระบบปฏิบัติการ AIX จัดเตรียมเคอร์เนล 64 บิตที่ แก้ไขภาวะคอขวดซึ่งขัดขวางผลผลิตบนระบบ 32 ทิศทาง

เนื่องจาก AIX 7.1 เคอร์เนลแบบ 64 บิตคือเคอร์เนลที่พร้อมใช้งานเท่านั้น ระบบ POWER4 จะถูกออปติไมซ์สำหรับเคอร์เนลแบบ 64 บิต ซึ่งมีแนวโน้มที่จะเพิ่มความสามารถในการวัดระบบ RS/6000 System p ระบบจะถูกออปติไมซ์สำหรับการรันแอ็พพลิเคชันแบบ 64 บิตบนระบบ POWER4 เคอร์เนลแบบ 64 บิตยังปรับปรุงความสามารถในการวัดโดยอนุญาตให้ใช้จำนวนของหน่วยความจำฟิสิคัล ที่มีขนาดใหญ่กว่า

นอกจากนี้ JFS2 คือระบบไฟล์ดีฟอลต์สำหรับ AIX 7.1 คุณสามารถเลือกเพื่อใช้ JFS หรือ JFS ที่ปรับปรุงแล้วได้ สำหรับข้อมูลเพิ่มเติมเกี่ยวกับ JFS ที่ปรับปรุงแล้ว โปรดดู ผลการทำงานของระบบไฟล์

แอ็พพลิเคชันแบบ 64 บิตบนเคอร์เนลแบบ 32 บิต

ผลการทำงานของแอ็พพลิเคชันแบบ 64 บิตที่ทำงานอยู่บนเคอร์เนลแบบ 64 บิต บนระบบ POWER4 ควรมีผลการทำงานที่มากกว่าหรือเท่ากับแอ็พพลิเคชันเดียวกันที่รันอยู่บนฮาร์ดแวร์เดียวกัน ด้วยเคอร์เนลแบบ 32 บิต

เคอร์เนลแบบ 64 บิตอนุญาตให้แอ็พพลิเคชันแบบ 64 บิตที่ต้องการให้การสนับสนุน โดยไม่จำเป็นต้องให้ระบบเรียกพารามิเตอร์เพื่อทำการแม็พใหม่หรือเปลี่ยนรูปแบบใหม่ เคอร์เนลแอ็พพลิเคชันแบบ 64 บิต จะใช้คอมไพลเลอร์ระดับสูงมากขึ้น ซึ่งจะออปติไมซ์โดยเฉพาะกับระบบ POWER4

แอ็พพลิเคชันแบบ 32 บิตบนเคอร์เนลแบบ 64 บิต

ในอินสแตนซ์ส่วนใหญ่ แอ็พพลิเคชันแบบ 32 บิตสามารถรันบนเคอร์เนลแบบ 64 บิต ได้โดยไม่มีการลดระดับผลการทำงานลง

แอ็พพลิเคชันแบบ 32 บิตบนเคอร์เนลแบบ 64 บิตจะมีผลการทำงานที่ต่ำกว่าการเรียกบน 32 บิต เนื่องจากมีการเปลี่ยนรูปแบบพารามิเตอร์ใหม่ การลดระดับผลการทำงานนี้ จะไม่มากเกินกว่า 5% ตัวอย่างเช่น การเรียกคำสั่ง **fork()** อาจส่งผลทำให้มีการใช้งานที่มากเกินไป

แอ็พพลิเคชันแบบ 64 บิตบนเคอร์เนลแบบ 64 บิตสำหรับระบบที่ไม่ใช่ POWER4

ผลการทำงานของแอ็พพลิเคชันแบบ 64 บิตภายใต้เคอร์เนลแบบ 64 บิต บนระบบที่ไม่ใช่ POWER4 อาจมีผลการทำงานที่ต่ำกว่าแอ็พพลิเคชันเดียวกันบนฮาร์ดแวร์เดียวกันภายใต้เคอร์เนลแบบ 32 บิต

ระบบที่ไม่ใช่ POWER4 จะเป็นบริดจ์ให้กับระบบ POWER4 และจะขาดการสนับสนุนบางอย่างที่จำเป็นสำหรับผลการทำงานของเคอร์เนลแบบ 64 บิต

ส่วนขยายเคอร์เนลแบบ 64 บิตบนระบบที่ไม่ใช่ POWER4

ผลการทำงานของส่วนขยายเคอร์เนลแบบ 64 บิตบนระบบ POWER4 ควรเป็นสำเนาเดียวกันหรือดีกว่าสำเนาแบบ 32 บิตบนฮาร์ดแวร์เดียวกัน

อย่างไรก็ตาม ผลการทำงานของส่วนขยายเคอร์เนลแบบ 64 บิตบนเครื่องที่ไม่ใช่ POWER4 อาจต่ำกว่าส่วนขยายเคอร์เนลแบบ 32 บิตบนฮาร์ดแวร์เดียวกัน เนื่องจากขาดการออปติไมซ์สำหรับผลการทำงานของเคอร์เนลแบบ 64 บิตบนระบบที่ไม่ใช่ POWER4

ระบบไฟล์ที่เจอร์นัลแล้วซึ่งได้รับการปรับปรุง

JFS ที่ปรับปรุงแล้ว หรือ JFS2 คือระบบไฟล์การทำเจอร์นัล AIX เนทีฟอื่น นี้คือระบบไฟล์ดีฟอลต์สำหรับ AIX 6.1 และใหม่กว่า

สำหรับข้อมูลเพิ่มเติมเกี่ยวกับ FS ที่ปรับปรุงแล้ว โปรดดู “ผลการทำงานของระบบไฟล์” ในหน้า 257

ผลการทำงานของไมโครโพรเซสเซอร์

หัวข้อนี้จะสอดแทรกข้อมูลเกี่ยวกับเทคนิคสำหรับการตรวจพบการหนี หรือโปรแกรมที่ใช้ตัวประมวลผล และลดผลกระทบที่มีต่อผลการทำงานของระบบลง

ถ้าคุณไม่คุ้นเคยกับการกำหนดตารางเวลาไมโครโพรเซสเซอร์ คุณอาจต้องการอ้างถึงหัวข้อ “ประสิทธิภาพตัวจัดตารางเวลาตัวประมวลผล” ในหน้า 43 ก่อนที่จะดำเนินการต่อ

การมอนิเตอร์ผลการทำงานของไมโครโพรเซสเซอร์

หน่วยการประมวลผลคือหนึ่งในคอมโพเนนต์ของระบบที่มีความเร็วสูงสุด

การเปรียบเทียบกันแบบนานๆ ครั้งสำหรับโปรแกรมเดี่ยวเพื่อทำให้ไมโครโพรเซสเซอร์มี 100 เปอร์เซ็นต์ของเวลาทำงาน (นั่นคือ 0 เปอร์เซ็นต์ของเวลาสูญเปล่า และ 0 เปอร์เซ็นต์ของเวลารอ) เพื่อให้มีเวลาเพิ่มขึ้นมากกว่าสองถึงสามวินาทีในแต่ละครั้ง แม้ว่าในระบบหลายผู้ใช้ที่มีการโหลดค่อนข้างมาก จะมีช่วงเวลา 10 มิลลิวินาทีที่จะสิ้นสุดเธรดทั้งหมดที่อยู่ในสถานะรอเป็นครั้งคราว ถ้ามอนิเตอร์แสดงไมโครโพรเซสเซอร์ 100 เปอร์เซ็นต์ของเวลาทำงาน สำหรับช่วงเวลาที่ขยายเพิ่ม นั่นเป็นโอกาสที่ดีสำหรับโปรแกรมบางตัวที่อยู่ใน การวนรอบไม่รู้จบ แม้ว่าโปรแกรมจะเป็น ″เพียง″ การสิ้นเปลื้องเท่านั้น แทนที่จะหยุดการทำงานที่ต้องการระบุหรือทำงานด้วย

คำสั่ง vmstat

เครื่องมือแรกที่จะใช้คือคำสั่ง **vmstat** ซึ่งให้ข้อมูลย่อเกี่ยวกับรีซอร์สระบบต่างๆ และ ปัญหาประสิทธิภาพที่เกี่ยวข้องกับรีซอร์สนั้น

คำสั่ง **vmstat** รายงานสถิติเกี่ยวกับเคอร์นัลเธรด ในรันคิวและคิวการรอ หน่วยความจำ การเพจ ดิสก์ การขัดจังหวะ การเรียกระบบ การสลับบริบท และกิจกรรม CPU กิจกรรม CPU ที่รายงานคือ รายละเอียดเปอร์เซ็นต์ของโหมดผู้ใช้ โหมดระบบ เวลา idle, และการรอดิสก์ I/O

หมายเหตุ: ถ้าใช้คำสั่ง **vmstat** โดยไม่กำหนดช่วงเวลา ดังนั้นคำสั่งจะสร้างรายงานเดียว รายงานเดียวเป็นรายงานค่าเฉลี่ยจากเวลาที่ระบบถูกสตาร์ท คุณสามารถระบุเฉพาะพารามิเตอร์ Count กับพารามิเตอร์ Interval ถ้าระบุพารามิเตอร์ Interval โดยไม่ระบุพารามิเตอร์ Count ดังนั้นรายงานจะถูกสร้างอย่างต่อเนื่อง

เนื่องจากเป็นตัวมอนิเตอร์ CPU คำสั่ง **vmstat** จะมีผลเหนือคำสั่ง **iostat** ที่เอาต์พุตแบบหนึ่งบรรทัดต่อรายงานจะสามารถดูได้ง่ายเมื่อรายงานเลื่อนขึ้น และมีโอเวอร์เฮดที่เกี่ยวข้องน้อยกว่า ในกรณีที่มีดิสก์จำนวนมากต่ออยู่กับระบบ ตัวอย่างต่อไปนี้สามารถช่วยคุณ ระบุสถานการณ์ซึ่งโปรแกรม run away หรือเน้น CPU มากเกินกว่า จะรันในสภาพแวดล้อมหลายผู้ใช้

```
# vmstat 2
kthr     memory             page              faults        cpu
----- ----------- ------------------------ ------------ -----------
 r  b   avm   fre  re  pi  po  fr   sr  cy  in   sy  cs us sy id wa
 1  0 22478  1677   0   0   0   0    0   0 188 1380 157 57 32  0 10
 1  0 22506  1609   0   0   0   0    0   0 214 1476 186 48 37  0 16
 0  0 22498  1582   0   0   0   0    0   0 248 1470 226 55 36  0  9

 2  0 22534  1465   0   0   0   0    0   0 238  903 239 77 23  0  0
 2  0 22534  1445   0   0   0   0    0   0 209 1142 205 72 28  0  0
```

```
 2  0 22534  1426    0    0    0    0     0    0 189 1220 212 74 26  0  0
 3  0 22534  1410    0    0    0    0     0    0 255 1704 268 70 30  0  0
 2  1 22557  1365    0    0    0    0     0    0 383  977 216 72 28  0  0

 2  0 22541  1356    0    0    0    0     0    0 237 1418 209 63 33  0  4
 1  0 22524  1350    0    0    0    0     0    0 241 1348 179 52 32  0 16
 1  0 22546  1293    0    0    0    0     0    0 217 1473 180 51 35  0 14
```

เอาต์พุตนี้แสดงผลกระทบของการเริ่มใช้โปรแกรมใน tight loop บน ระบบหลายผู้ใช้ที่ยุ่ง รายงานสามฉบับแรก (สรุปถูกลบออก) แสดงระบบซึ่งสมดุลที่ 50–55 เปอร์เซ็นต์ผู้ใช้, 30–35 เปอร์เซ็นต์ระบบ, และ 10–15 เปอร์เซ็นต์รอ I/O เมื่อโปรแกรม looping เริ่มขึ้น จะมีการใช้รอบ CPU ที่มีอยู่ทั้งหมด เนื่องจากโปรแกรม looping ไม่ได้ทำ I/O จึงสามารถรับ รอบทั้งหมดที่ไม่ได้ใช้ก่อนหน้านี้เนื่องจากรอ I/O สิ่งที่เลวร้ายที่สุดคือ แสดงถึง กระบวนการที่พร้อมจะครอบครอง CPU เสมอเมื่อโปรแกรมที่มีประโยชน์ ออกจาก CPU ไป เนื่องจากโปรแกรม looping มีระดับความสำคัญเท่ากับโปรแกรมของ กระบวนการพื้นหน้าอื่นทั้งหมด โปรแกรมจึงไม่จำเป็นต้องยอมให้ CPU เมื่อกระบวนการอื่นสามารถจัดส่งได้ โปรแกรมรันประมาณ 10 วินาที (ห้ารายงาน) จากนั้น กิจกรรมที่รายงานโดยคำสั่ง **vmstat** มีรูปร่างเป็นปกติมากขึ้น

การใช้งานที่สมบูรณ์ที่สุดคือมีการใช้ CPU 100 เปอร์เซ็นต์ของเวลา ซึ่งเป็นจริง ในกรณีของระบบผู้ใช้รายเดียวที่ไม่ต้องแบ่งใช้ CPU โดยทั่วไป ถ้าเวลา `us` + `sy` ต่ำกว่า 90 เปอร์เซ็นต์ ไม่ถือว่าระบบผู้ใช้รายเดียว มีข้อจำกัด CPU อย่างไรก็ตาม ถ้าเวลา `us` + `sy` บนระบบหลายผู้ใช้เกินกว่า 80 เปอร์เซ็นต์ กระบวนการอาจต้องรอ ในรันคิว เวลาการตอบกลับและผลผลิตอาจได้รับผลกระทบ

เพื่อตรวจสอบว่า CPU เป็นปัญหาคอขวดหรือไม่ให้พิจารณาคอลัมน์ที่สี่ `cpu` และคอลัมน์ที่สอง `kthr` (เคอร์นัลเธรด) ในรายงาน **vmstat** อาจมีประโยชน์ที่จะดูคอลัมน์ `faults` ด้วย:

- **cpu**

  รายละเอียดเปอร์เซ็นต์ของการใช้เวลา CPU ในระหว่างช่วงเวลา คอลัมน์ `cpu` มีดังนี้:

  - **us**

    คอลัมน์ `us` แสดงเปอร์เซ็นต์ของเวลา CPU ที่ใช้ ในโหมดผู้ใช้ กระบวนการ UNIX สามารถดำเนินการในโหมดผู้ใช้หรือโหมดระบบ (เคอร์นัล) เมื่ออยู่ในโหมดผู้ใช้ กระบวนการดำเนินการภายในแอ็พพลิเคชันโค้ดของตนเองและไม่ต้องการ รีซอร์สเคอร์นัลเพื่อทำการคำนวณ จัดการหน่วยความจำ หรือตั้งค่าตัวแปร

  - **sy**

    คอลัมน์ `sy` แสดงรายละเอียดเปอร์เซ็นต์ของเวลาที่ CPU ดำเนินกระบวนการในโหมดผู้ใช้ ซึ่งรวมถึงรีซอร์ส CPU ที่ใช้โดย กระบวนการเคอร์นัล (**kprocs**) และอื่นๆ ที่ต้องการการเข้าถึงรีซอร์ส เคอร์นัล ถ้ากระบวนการต้องการรีซอร์สเคอร์นัล ต้องทำการเรียกระบบ และสลับไปยังโหมดระบบเพื่อทำให้รีซอร์สนั้นมีอยู่ ตัวอย่างเช่น การอ่านหรือการบันทึกไฟล์ต้องการรีซอร์สเคอร์นัลเพื่อเปิดไฟล์ ค้นหาที่ตั้งเฉพาะ และอ่านหรือบันทึกข้อมูล ยกเว้นว่ามีการใช้ไฟล์ที่แม็พหน่วยความจำ

  - **id**

    คอลัมน์ `id` จะแสดงเปอร์เซ็นต์ของเวลาที่ CPU ไม่ทำงาน หรือรอ โดยไม่มี I/O ของโลคัลดิสก์ที่ค้างอยู่ ถ้าไม่มีเธรดที่พร้อมใช้งานสำหรับการดำเนินการ (คิวการรันว่างเปล่า) ระบบจะจัดการกับเธรดที่ชื่อ **wait** หรือที่เรียกว่า **idle kproc** บนระบบ SMP สามารถจัดส่งได้หนึ่งเธรด **wait** ต่อตัวประมวลผล รายงานที่สร้างขึ้นโดยคำสั่ง **ps** (ที่มีอ็อพชัน **–k** หรือ **–g 0**) ระบุข้อมูลนี้เป็น `kproc` หรือ `wait` ถ้ารายงาน **ps** แสดงเวลารวมสูงสำหรับเธรดนี้ หมายความว่ามีเวลายาวนานที่ไม่มีเธรดพร้อมจะรัน หรือกำลังรอที่จะดำเนินการบน CPU ส่วนใหญ่แล้ว ระบบ *idle* และ *กำลังรอ* ภารกิจใหม่

  - **wa**

คอลัมน์ wa แสดงรายละเอียดเปอร์เซ็นต์ของเวลาที่ CPU *idle* โดยมีโลคัลดิสก์ I/O ที่ค้างอยู่และดิสก์ที่ติดตั้ง NFS ถ้า มี I/O ค้างอยู่อย่างน้อยหนึ่งรายการที่ดิสก์เมื่อ **wait** กำลังรัน เวลาจะถูกจัดประเภทเป็นการรอ I/O ยกเว้นว่ากระบวนการใช้อะซิงโครนัส I/O การร้องขอ I/O ไปยังดิสก์ส่งผลให้กระบวนการที่เรียกถูกบล็อก (หรือพัก) จนกว่าการร้องขอเสร็จสมบูรณ์ หลังจากการร้องขอ I/O สำหรับกระบวนการเสร็จสมบูรณ์แล้ว จะมีการวางกระบวนการบนรันคิว ถ้า I/Os เสร็จได้เร็วขึ้น จะมีเวลา CPU ที่สามารถใช้มากขึ้น

ค่า wa มากกว่า 25 เปอร์เซ็นต์ อาจบ่งชี้ว่าระบบย่อยดิสก์อาจไม่สมดุล หรือ อาจเป็นผลมาจากเวิร์กโหลดที่เน้นดิสก์ หากต้องการข้อมูลเพิ่มเติม เกี่ยวกับการเปลี่ยนแปลงที่ทำใน wa ให้ดู “การรายงานเวลารอ I/O” ในหน้า 196

- **kthr**

  จำนวนของเคอร์นัลเธรดในคิวต่างๆ โดยเฉลี่ยต่อวินาที ในช่วงเวลาตัวอย่าง คอลัมน์ kthr มีดังนี้:

  - **r**

    จำนวนเฉลี่ยของเคอร์นัลเธรดที่รันได้ ซึ่งรวมถึง เธรดที่กำลังรันอยู่และเธรดที่กำลังรอ CPU ถ้าจำนวนนี้มากกว่าจำนวนของ CPUs แสดงว่าอาจมีอย่างน้อยหนึ่งเธรด กำลังรอ CPU และเธรดที่กำลังรอ CPUs ยิ่งมาก ผลกระทบ ต่อประสิทธิภาพยิ่งมาก

  - **b**

    จำนวนเฉลี่ยของเคอร์นัลเธรดในคิวการรอ VMM ต่อ วินาที จำนวนนี้รวมเธรดที่กำลังรอ I/O ระบบไฟล์หรือเธรด ที่ถูกพักไว้เนื่องจากการควบคุมโหลดหน่วยความจำ

    ถ้ากระบวนการถูกพักไว้ เนื่องจากการควบคุมโหลดหน่วยความจำ คอลัมน์ที่บล็อก (b) ในรายงาน **vmstat** บ่งชี้การเพิ่มขึ้นในจำนวนของเธรด แทนรันคิว

  - **p**

    สำหรับ **vmstat -I** จำนวนเธรด ที่กำลังรอ I/Os ที่อุปกรณ์ raw ต่อวินาที เธรดที่กำลังรอ I/Os ที่ระบบไฟล์ จะไม่รวมในค่านี้

- **faults**

  ข้อมูลเกี่ยวกับการควบคุมกระบวนการ เช่น trap และ อัตราการขัดจังหวะ คอลัมน์ faults มีดังนี้:

  - **in**

    จำนวนของการขัดจังหวะอุปกรณ์ต่อวินาทีที่สังเกตพบในช่วงเวลา สามารถดูข้อมูลเพิ่มเติมได้ใน “การประเมินค่าผลการทำงานดิสก์ด้วยคำสั่ง vmstat” ในหน้า 199

  - **sy**

    จำนวนของการเรียกระบบต่อวินาทีที่สังเกตพบในช่วงเวลา รีซอร์สมีอยู่สำหรับกระบวนการผู้ใช้ผ่านทางการเรียกระบบที่กำหนดไว้เป็นอย่างดี การเรียกเหล่านี้บอกให้เคอร์นัลทำการดำเนินงานสำหรับกระบวนการเรียก และแลกเปลี่ยนข้อมูลระหว่างเคอร์นัลและกระบวนการ เนื่องจากเวิร์กโหลดและ แอ็พพลิเคชันมีความแตกต่างกันมาก และการเรียกที่ต่างกันทำฟังก์ชันที่ต่างกัน จึงไม่สามารถกำหนดได้ว่าการเรียกระบบจำนวนเท่าไรต่อวินาทีถือเป็นการเรียกที่มากเกินไป แต่โดยปกติ เมื่อคอลัมน์ sy เพิ่มขึ้นมากกว่า 10000 การเรียกต่อวินาที บนตัวประมวลผลเดียว จะมีการเรียกการสืบสวนเพิ่มเติม (บนระบบ SMP จำนวนคือ 10000 เรียกต่อวินาทีต่อตัวประมวลผล) เหตุผลหนึ่งอาจเป็นรูทีนย่อย ″polling″ เช่น รูทีนย่อย **select( )** สำหรับคอลัมน์นี้ แนะนำให้มีค่าประเมินพื้นฐาน เพื่อให้จำนวนสำหรับค่า sy ปกติ

  - **cs**

    จำนวนของการสลับบริบทต่อวินาทีที่สังเกตพบในช่วงเวลา รีซอร์ส CPU ฟิสิคัลมีการแบ่งย่อยเป็นส่วนเวลาโลจิคัลส่วนละ 10 มิลลิวินาที สมมุติว่าเธรดมีการจัดตารางเวลาสำหรับการดำเนินการ เธรดจะรันจนกว่าส่วนเวลา ของตนจะหมดเวลา จนกว่าได้รับยกเว้น หรือจนกว่าเธรดสมัครใจจะยกเลิก การควบคุม CPU เมื่อเธรดอื่นได้รับการควบคุม CPU

สภาพแวดล้อมบริบท หรือการทำงานของเธรดก่อนหน้านี้ต้องถูกบันทึก และต้องโหลดบริบท ของเธรดปัจจุบัน ระบบปฏิบัติการมีโพรซีเดอร์การสลับบริบทที่มีประสิทธิภาพ มาก ดังนั้นการสลับแต่ละครั้งจึงไม่เสียรีซอร์สมากนัก การเพิ่มขึ้นอย่างมากในการสลับบริบท เช่น เมื่อ cs สูงกว่า ดิสก์ I/O และอัตราแพ็กเก็ตเครือข่ายมาก จำเป็นต้องมีการสืบสวนเพิ่มเติม

คำสั่ง iostat

คำสั่ง **iostat** จะเป็นวิธีที่เร็วที่สุด ในการขอรับความพึงพอใจในครั้งแรก ไม่ว่าระบบจะมีปัญหาเกี่ยวกับผลการทำงานในขอบเขตของดิสก์ I/O หรือไม่ก็ตาม

โปรดดู “การประเมินค่าผลการทำงานดิสก์ด้วยคำสั่ง iostat” ในหน้า 196 เครื่องมือยังรายงานข้อมูลสถิติ CPU

ตัวอย่างต่อไปนี้แสดงส่วนหนึ่งของเอาต์พุตคำสั่ง **iostat** stanza กลุ่มแรกจะแสดงข้อมูลสถิติโดยสรุปตั้งแต่เริ่มต้นทำงานกับระบบ

```
# iostat -t 2 6
tty:      tin         tout   avg-cpu:  % user    % sys     % idle    % iowait
          0.0          0.8               8.4      2.6       88.5       0.5
          0.0         80.2               4.5      3.0       92.1       0.5
          0.0         40.5               7.0      4.0       89.0       0.0
          0.0         40.5               9.0      2.5       88.5       0.0
          0.0         40.5               7.5      1.0       91.5       0.0
          0.0         40.5              10.0      3.5       80.5       6.0
```

คอลัมน์ข้อมูลสถิติ CPU (*% user*, *% sys*, *% idle* และ *% iowait*) จัดเตรียมการแตกของการใช้ CPU ข้อมูลนี้ยังรายงานอยู่ในเอาต์พุตคำสั่ง **vmstat** ในคอลัมน์ที่มีป้าย *us*, *sy*, *id* และ *wa* สำหรับคำอธิบายโดยละเอียดสำหรับค่า โปรดดู “คำสั่ง vmstat” ในหน้า 112 และให้จดบันทึกการเปลี่ยนแปลงที่ทำไว้ *%iowait* ซึ่งกล่าวไว้ใน “การรายงานเวลารอ I/O” ในหน้า 196

งานที่เกี่ยวข้อง:

“การประเมินค่าผลการทำงานดิสก์ด้วยคำสั่ง iostat” ในหน้า 196
เริ่มต้นการประเมินผลโดยรันคำสั่ง **iostat** ด้วยพารามิเตอร์ช่วงเวลาในระหว่างที่มีเวิร์กโหลดสูงสุดของระบบ หรือขณะที่รันแอ็พพลิเคชันที่สำคัญซึ่งคุณต้องการลดเวลาหน่วงของ I/O

สิ่งอ้างอิงที่เกี่ยวข้อง:

“การมอนิเตอร์ผลการทำงานอย่างต่อเนื่องด้วยคำสั่ง iostat” ในหน้า 16
คำสั่ง **iostat** มีประโยชน์สำหรับการพิจารณาดิสก์ และการใช้ CPU

คำสั่ง sar

คำสั่ง **sar** รวบรวมข้อมูลสถิติเกี่ยวกับ ระบบ

แม้ว่าสามารถใช้คำสั่งนี้เพื่อรวบรวมข้อมูลที่เป็นประโยชน์บางอย่างเกี่ยวกับประสิทธิภาพระบบได้ก็ตาม แต่คำสั่ง **sar** อาจทำให้โหลดระบบสูงขึ้นซึ่งอาจซ้ำเติมให้ปัญหา ประสิทธิภาพที่มีอยู่แล้วแย่ลงอีก ถ้าความถี่ในการสุ่มตัวอย่างสูง แต่ เมื่อเปรียบเทียบระหว่างข้อดีและข้อเสียแล้ว คำสั่ง **sar** มีข้อดี มากกว่า ระบบเก็บรักษาชุดของตัวนับกิจกรรมระบบ ซึ่งจัดทำเร็กคอร์ดกิจกรรมต่างๆ และให้ข้อมูลที่คำสั่ง **sar** รายงาน คำสั่ง **sar** ไม่ทำให้เกิดการอัพเดตหรือการใช้ตัวนับเหล่านี้ การอัพเดตหรือการใช้มีการทำโดยอัตโนมัติ ไม่ว่าคำสั่ง **sar** รันหรือไม่ก็ตาม คำสั่งเพียงแต่แยกข้อมูลในตัวนับ และบันทึก ตามข้อมูลอัตราการสุ่มตัวอย่างและจำนวนของตัวอย่างที่ระบุให้กับ คำสั่ง **sar** เท่านั้น

ด้วยอ็อพชันที่มีเป็นจำนวนมาก คำสั่ง **sar** นำเสนอสถิติการจัดคิว การเพจ, TTY, และสถิติอื่นอีกมาก คุณลักษณะที่สำคัญหนึ่งของคำสั่ง **sar** คือการรายงานสถิติ CPU ของทั้งระบบ (สากล ของตัวประมวลผลทั้งหมด) (ซึ่งคำนวณเป็นค่าเฉลี่ยสำหรับค่าที่ระบุ เป็นเปอร์เซ็นต์ หรือเป็นผลรวม) หรือรายงานสถิติสำหรับตัวประมวลผล แต่ละตัว ดังนั้น คำสั่งนี้จึงมีประโยชน์มากเป็นพิเศษบน ระบบ SMP

มีการใช้คำสั่ง **sar** ในสามกรณีดังนี้:

การสุ่มตัวอย่างแบบเรียลไทม์และการแสดงผล:

เมื่อต้องการรวบรวมและแสดงรายงานสถิติระบบในทันที ให้รันคำสั่ง **sar**

ใช้คำสั่งต่อไปนี้:

```
# sar -u 2 5

AIX ses12 1 6 000126C5D600    04/08/08

System configuration: lcpu=2  mode=Capped

19:42:43    %usr    %sys    %wio   %idle   physc
19:42:45       0       2       1      97    0.98
19:42:47       0       0       0     100    1.02
19:42:49       0       0       0     100    1.00
19:42:51       0       0       0     100    1.00
19:42:53       0       0       0     100    1.00

Average        0       1       0      99    1.00
```

ตัวอย่างนี้ได้มาจากเวิร์กสเตชันที่มีผู้ใช้รายเดียว และแสดงการใช้ประโยชน์ CPU

การแสดงข้อมูลที่ดักจับไว้ก่อนหน้านี้:

อ็อพชัน **-o** และ **-f** (เขียนและอ่านไปยัง/จาก ผู้ใช้ที่กำหนดไฟล์ข้อมูล) อนุญาตให้คุณมองเห็นลักษณะการทำงานของเครื่องในสองขั้นตอน ซึ่งจะใช้รีซอร์สน้อยลง ในช่วงเวลาของการสร้างแก้ปัญหา

คุณสามารถใช้เครื่องที่แยกออกเพื่อวิเคราะห์ข้อมูล โดยการถ่ายโอนไฟล์เนื่องจากไฟล์ไบนารีที่สะสมไว้จะเก็บข้อมูลทั้งหมดที่คำสั่ง **sar** ต้องการ

```
# sar -o /tmp/sar.out 2 5 > /dev/null
```

คำสั่งข้างต้นจะรันคำสั่ง **sar** ในส่วนหลัง เก็บรวบรวมข้อมูลกิจกรรมระบบในเวลา 2 วินาทีทั้งหมด 5 ช่วงเวลา และเก็บข้อมูล (ที่ไม่ได้จัดรูปแบบ) **sar** ลงในไฟล์ `/tmp/sar.out` การเปลี่ยนทิศทางของเอาต์พุตมาตรฐาน จะถูกนำมาใช้เพื่อหลีกเลี่ยนเอาต์พุตบนหน้าจอ

คำสั่งต่อไปนี้จะแตกข้อมูล CPU จากไฟล์ และเอาต์พุตรายงานที่จัดรูปแบบแล้วไปยังเอาต์พุตมาตรฐาน:

```
# sar -f/tmp/sar.out

AIX ses12 1 6 000126C5D600    04/08/08

System configuration: lcpu=2  mode=Capped
```

```
20:17:00    %usr    %sys    %wio   %idle   physc
20:18:00       0       1       0      99    1.00
20:19:00       0       1       0      99    1.00
20:20:00       0       1       0      99    1.00
20:21:01       0       1       0      99    1.00
20:22:00       0       0       0      99    1.00

Average        0       1       0      99    1.00
```

ไฟล์ข้อมูลแบบไบนารีที่ดักจัดจะเก็บข้อมูลทั้งหมดที่ต้องการสำหรับ รายงาน รายงาน **sar** ทุกรายงานสามารถตรวจสอบได้ ซึ่งอนุญาตให้แสดงข้อมูลที่ระบุเฉพาะตัวประมวลผลของระบบ SMP บนระบบตัวประมวลผลเดี่ยว

การลงบัญชีกิจกรรมระบบผ่านทาง cron daemon:

สอง shell scripts (`/usr/lib/sa/sa1` และ `/usr/lib/sa/sa2`) มีการสร้างขึ้นเพื่อรันโดย **cron** daemon และให้สถิติและรายงาน ประจำวัน

คำสั่ง **sar** เรียกกระบวนการที่มีชื่อว่า **sadc** เพื่อ เข้าถึงข้อมูลระบบ ตัวอย่าง stanzas มีการรวมไว้ (แต่ commented out) ในไฟล์ `/var/spool/cron/crontabs/adm` crontab ที่จะระบุเมื่อ **cron** daemon ควรจะ รัน shell scripts

บรรทัดต่อไปนี้แสดง crontab ที่แก้ไขสำหรับผู้ใช้ adm มีการลบ เฉพาะอักขระข้อคิดเห็นสำหรับการรวบรวมข้อมูลเท่านั้น:

```
#==========================================================
#      SYSTEM ACTIVITY REPORTS
#  8am-5pm activity reports every 20 mins during weekdays.
#  activity reports every an hour on Saturday and Sunday.
#  6pm-7am activity reports every an hour during weekdays.
#  Daily summary prepared at 18:05.
#==========================================================
0 8-17 * * 1-5 /usr/lib/sa/sa1 1200 3 &
0 * * * 0,6 /usr/lib/sa/sa1 &
0 18-7 * * 1-5 /usr/lib/sa/sa1 &
5 18 * * 1-5 /usr/lib/sa/sa2 -s 8:00 -e 18:01 -i 3600 -ubcwyaqvm &
#==========================================================
```

การรวบรวมข้อมูลในลักษณะนี้มีประโยชน์ในการกำหนดลักษณะการใช้งานระบบ ในรอบระยะเวลาหนึ่ง และเพื่อกำหนดชั่วโมงการใช้งานสูงสุด

อ็อพชัน microprocessor ที่มีประโยชน์:

มีอ็อพชันที่เกี่ยวข้องกับ microprocessor ซึ่งมีประโยชน์จำนวนมาก สำหรับคำสั่ง **sar**

อ็อพชันที่มีประโยชน์มากที่สุดมีดังนี้:

- **sar -P**

  อ็อพชัน **-P** รายงาน สถิติสำหรับแต่ละตัวประมวลผลสำหรับตัวประมวลผลที่ระบุ โดยการระบุคีย์เวิร์ด **ALL** จะมีการรายงานสถิติเกี่ยวกับตัวประมวลผล แต่ละตัวและค่าเฉลี่ยสำหรับตัวประมวลผลทั้งหมด เมื่อใช้ **-P ALL** ภายในสภาพแวดล้อมเวิร์กโหลดพาร์ติชัน สถิติ RSET-wide จะแสดงขึ้นเพิ่มเติมจากสถิติทั้งระบบ และตัวประมวลผลที่เป็นสมาชิกของ

RSET จะมีสัญลักษณ์เครื่องหมายดอกจัน (*) อยู่ข้างหน้า สถิติ RSET-wide จะแสดงขึ้นเฉพาะถ้า สภาพแวดล้อมเวิร์กโหลดพาร์ติชันเชื่อมโยงกับ RSET เท่านั้น ในบรรดาแฟล็กซึ่งระบุสถิติที่จะรายงาน เฉพาะแฟล็ก **-a**, **-c**, **-m**, **-u**, และ **-w** เท่านั้นที่มีความหมายกับแฟล็ก **-P**

ตัวอย่าง ต่อไปนี้แสดงสถิติสำหรับแต่ละตัวประมวลผลในขณะที่โปรแกรมที่ยึดไว้กับ microprocessor กำลังรันอยู่บนตัวประมวลผลหมายเลข 0:

```
# sar -P ALL 2 2

AIX tooltime2 1 6 00CA52594C00    04/02/08
System configuration: lcpu=4  mode=Capped
05:23:08 cpu    %usr    %sys    %wio   %idle   physc
05:23:11  0        8      92       0       0    1.00
          1        0      51       0      49    0.00
          2        0       0       0     100    0.51
          3        0       0       0     100    0.48
          -        4      46       0      50    1.99
05:23:13  0       10      89       0       0    1.00
          1        0       7       0      93    0.00
          2        0       3       2      95    0.51
          3        0       0       0     100    0.49
          -        5      45       0      49    2.00

Average   0        9      91       0       0    1.00
          1        0      12       0      88    0.00
          2        0       2       1      98    0.51
          3        0       0       0     100    0.48
          -        5      46       0      49    1.99
```

บรรทัดสุดท้ายของทุก stanza ซึ่งขึ้นต้นด้วย เครื่องหมายขีด (-) ในคอลัมน์ cpu คือค่าเฉลี่ยสำหรับตัวประมวลผล ทั้งหมด บรรทัดค่าเฉลี่ย (-) แสดงขึ้นเฉพาะถ้ามีการใช้อ็อพชัน **-P ALL** เท่านั้น บรรทัดนี้จะถูกลบออกถ้ามีการระบุตัวประมวลผล Stanza สุดท้ายที่มีชื่อว่า Average แทนเวลาประทับ เก็บรักษาค่าเฉลี่ยสำหรับแถวเฉพาะตัวประมวลผลบน stanzas ทั้งหมด

ตัวอย่างต่อไปนี้แสดงเอาต์พุต **vmstat** ในระหว่างเวลานี้:

```
System configuration: lcpu=4 mem=4457OMB
kthr    memory              page              faults        cpu
----- ----------- ------------------------ ------------ -----------
 r  b   avm   fr       re  pi  po  fr   sr  cy  in   sy  cs us sy id wa
 2  0 860494 6020610   0   0   0   0    0   0  16 14061 409  5 45 49  0
 2  0 860564 6020540   0   0   0   0    0   0   4 14125 400  5 45 50  0
 1  0 860669 6020435   0   0   0   0    0   0   3 14042 388  5 46 49  0
 2  0 860769 6020335   0   0   0   0    0   0   3 13912 398  5 45 50  0
```

บรรทัดแรกที่มีหมายเลขคือสรุปตั้งแต่ เริ่มต้นระบบ บรรทัดที่สองแสดงการเริ่มต้นของคำสั่ง **sar** และเมื่อใช้ร่วมกับแถวที่สาม สามารถเปรียบเทียบค่าในรายงานได้ คำสั่ง **vmstat** สามารถแสดง การใช้ประโยชน์ microprocessor เฉลี่ยบนตัวประมวลผลทั้งหมดเท่านั้น ซึ่งเปรียบเทียบได้กับแถวที่มีเครื่องหมายขีด (-) จากเอาต์พุตแสดงการใช้ ประโยชน์ microprocessor จากคำสั่ง **sar**

เมื่อรัน ภายในสภาพแวดล้อม WPAR คำสั่งเดียวกันจะให้เอาต์พุต ดังต่อไปนี้:

```
AIX wpar1 1 6 00CBA6FE4C00    04/01/08
wpar1 configuration: lcpu=2  memlim=204MB  cpulim=0.06  rset=Regular
05:23:08 cpu    %usr    %sys    %wio   %idle   physc
05:23:11 *0        8      92       0       0    1.00
         *1        0      51       0      49    0.00
```

```
          2        0        0        0     100    0.51
          3        0        0        0     100    0.48
          R        4       71        0      24    1.00
          -        4       46        0      50    1.99
05:23:13 *0       10       89        0       0    1.00
         *1        0        7        0      93    0.00
          2        0        3        2      95    0.51
          3        0        0        0     100    0.49
          R        5       48        0      46    1.00
          -        5       45        0      49    2.00

Average  *0        9       91        0       0    1.00
         *1        0       12        0      88    0.00
          2        0        2        1      98    0.51
          3        0        0        0     100    0.48
          R        4       51        0      44    1.00
          -        5       46        0      49    1.99
```

WPAR มีทะเบียน RSET ที่เชื่อมโยง ตัวประมวลผล 0 และ 1 มีการแนบเข้ากับ RSET บรรทัด R แสดงการใช้โดย RSET ที่เชื่อมโยงกับ WPAR ตัวประมวลผลที่แสดงใน RSET มีสัญลักษณ์เครื่องหมายดอกจัน (*) อยู่ข้างหน้า

- **sar −P** RST ใช้เพื่อแสดงเมตริกซ์การใช้งาน ของตัวประมวลผลที่แสดงใน RSET ถ้าไม่มี RSET ที่เชื่อมโยงกับสภาพแวดล้อม WPAR เมตริกซ์ของตัวประมวลผลทั้งหมด จะแสดงขึ้น

  ตัวอย่างต่อไปนี้แสดงการรัน **sar −P** RST ภายในสภาพแวดล้อม WPAR:

```
AIX wpar1 1 6 00CBA6FE4C00    04/01/08

wpar1 configuration: lcpu=2  memlim=204MB  cpulim=0.06  rset=Regular

05:02:57 cpu    %usr    %sys    %wio   %idle   physc
05:02:59  0       20      80       0       0    1.00
          1       10       0       0      90    0.00
          R       15      40       0      45    1.00
05:03:01  0       20      80       0       0    1.00
          1        8       0       0      92    0.00
          R       14      40       0      46    1.00

Average   0       20      80       0       0    1.00
          1        9       0       0      91    0.00
          R       14      40       0      46    1.00
```

- **sar −u**

  แสดงการใช้ประโยชน์ microprocessor แฟล็กนี้เป็นค่าดีฟอลต์ถ้าไม่มีการระบุแฟล็กอื่น แฟล็กนี้แสดง ข้อมูลเดียวกันกับสถิติ microprocessor ของคำสั่ง **vmstat** หรือ **iostat**

  ในระหว่าง ตัวอย่างต่อไปนี้ คำสั่ง copy เริ่มต้นขึ้น:

```
# sar -u 3 3
AIX wpar1 1 6 00CBA6FE4C00    04/01/08
wpar1 configuration: lcpu=2  memlim=204MB  cpulim=0.06  rset=Regular
05:02:57 cpu    %usr    %sys    %wio   %idle   physc
05:02:59  0       20      80       0       0    1.00
          1       10       0       0      90    0.00
          R       15      40       0      45    1.00
05:03:01  0       20      80       0       0    1.00
          1        8       0       0      92    0.00
```

```
          R       14      40       0      46    1.00

Average   0       20      80       0       0    1.00
          1        9       0       0      91    0.00
          R       14      40       0      46    1.00
```

เมื่อรันภายในเวิร์กโหลดพาร์ติชัน คำสั่งเดียวกันจะให้เอาต์พุตดังต่อไปนี้:

```
AIX wpar1 1 6 00CBA6FE4C00    04/01/08

wpar1 configuration: lcpu=2  memlim=204MB  cpulim=0.06  rset=Regular

05:07:16    %usr    %sys    %wio   %idle   physc   %resc
05:07:19      17      83       -       -    0.11   181.6
05:07:22      19      81       -       -    0.08   133.5
05:07:26      16      84       -       -    0.10   173.4

Average       17      83       -       -    0.10   164.3
```

เอาต์พุตนี้แสดงข้อมูล %resc สำหรับเวิร์กโหลด พาร์ติชันที่มีการบังคับใช้ขีดจำกัดรีซอร์สตัวประมวลผล เมตริกซ์นี้ บ่งชี้เปอร์เซ็นต์ของรีซอร์สตัวประมวลผลที่ใช้โดยเวิร์กโหลด พาร์ติชัน

- **sar -c**

  อ็อพชัน **-c** แสดง อัตราการเรียกระบบ

```
# sar -c 1 3
19:28:25 scall/s sread/s swrit/s  fork/s  exec/s rchar/s wchar/s
19:28:26     134      36       1    0.00    0.00 2691306    1517
19:28:27      46      34       1    0.00    0.00 2716922    1531
19:28:28      46      34       1    0.00    0.00 2716922    1531

Average       75      35       1    0.00    0.00 2708329    1527
```

  ในขณะที่ คำสั่ง **vmstat** แสดงอัตราการเรียกระบบเช่นเดียวกัน แต่คำสั่ง **sar** ยังสามารถแสดงด้วยว่าการเรียกระบบเหล่านี้เป็น **read( ), write( ), fork( ), exec( )**, และ อื่นๆ ให้ความสนใจเป็นพิเศษกับคอลัมน์ fork/s ถ้าค่า นี้สูง อาจต้องสืบสวนเพิ่มเติมโดยใช้ฟังก์ชันการลงบัญชี คำสั่ง **trace** หรือคำสั่ง **tprof**

- **sar -q**

  อ็อพชัน **-q** แสดง ขนาดรันคิวและขนาด swap-queue

```
# sar -q 5 3

19:31:42 runq-sz %runocc swpq-sz %swpocc
19:31:47     1.0     100     1.0     100
19:31:52     2.0     100     1.0     100
19:31:57     1.0     100     1.0     100

Average      1.3      95     1.0      95
```

  runq-sz

  จำนวนเธรดเฉลี่ยที่สามารถรันได้ต่อวินาทีและ เปอร์เซ็นต์ของเวลาที่เป็นของรันคิว (% ที่ฟิลด์ อาจมีข้อผิดพลาด)

  swpq-sz

  จำนวนเธรดเฉลี่ยในคิวการรอ VMM และ % ของเวลาที่เป็นของ swap queue (% ที่ฟิลด์อาจมี ข้อผิดพลาด)

อ็อพชัน **-q** สามารถบ่งชี้ว่าคุณมีงานที่กำลังรันมากเกินไป หรือไม่ (runq-sz) หรือมีปัญหาคอขวดการเพจที่อาจเกิดขึ้นได้หรือไม่ ในระบบที่มีธุรกรรมสูง ตัวอย่างเช่น Enterprise Resource Planning (ERP) รันคิวอาจมีจำนวนหลายร้อย เนื่องจากแต่ละธุรกรรมใช้เวลา microprocessor จำนวนเล็กน้อย ถ้าการเพจ เป็นปัญหา ให้รันคำสั่ง **vmstat** การรอ I/O สูงบ่งชี้ว่ามีกิจกรรมการช่วงชิงดิสก์สูงหรือ มีการเพจมากเกินไปเนื่องจากหน่วยความจำไม่เพียงพอ

การใช้ชุดรีซอร์ส:

ปฏิบัติตามคำแนะนำเกี่ยวกับแนวปฏิบัติที่เหมาะสมที่สุดเหล่านี้สำหรับการใช้ชุดรีซอร์ส

ข้อแนะนำ

1. ถ้าชุดรีซอร์สเป็นแกนย่อยแต่มีตัวประมวลผลมากกว่าหนึ่งตัว จากแต่ละแกน เพื่อให้ได้ผลลัพธ์ที่ดีที่สุด ควรรวมเธรดหลัก และเธรด simultaneous multithreading (SMT) ต่อเนื่องในลำดับต่อมา (ตัวประมวลผลโลจิคัล) เพื่อให้ไม่มีช่องว่างในตัวประมวลผล ซึ่งรวมไว้ในแกน
2. ชุดรีซอร์สมีตัวประมวลผลจำนวนเดียวกันกับตัวประมวลผลที่แต่ละ แกนแสดงถึง

คำแนะนำ

ข้อความสั่งต่อไปนี้เป็นจริงเมื่อใช้ชุดรีซอร์ส:

- ชุดรีซอร์สสามารถมีแกนจากหลาย scheduler resource allocation domains (SRAD) และไม่จำเป็นต้องมีจำนวนแกนเท่ากันกับแกน จากแต่ละ SRAD
- ชุดรีซอร์สสามารถมีหนึ่งตัวประมวลผลจากแต่ละแกนที่ แสดงถึง และไม่จำเป็นต้องเป็นเธรด SMT หลัก
- เมื่อมีการเปิดใช้งานการปรับสมดุลโหลดสำหรับชุดรีซอร์ส ระบบย่อย folding ตัวประมวลผล เช่น VPM จะทำการตัดสินใจ folding โดยพิจารณาจาก แกนที่ชุดรีซอร์สต้องการ แกน ที่มีงานซึ่งใช้ชุดรีซอร์สมากที่สุดจะได้รับลำดับความสำคัญก่อนเมื่อตัดสินใจ fold หรือ unfolded แกน

  หมายเหตุ: ถ้ามีการเปิดใช้งานโหมด ประหยัดพลังงาน VPM จะทำการเลือกแกนโดยคำนึงถึงพลังงานแม้ว่า มีการเปิดใช้งานการปรับสมดุลโหลดสำหรับชุดรีซอร์ส

โปรแกรม xmperf

การใช้โปรแกรม **xmperf** แสดงการใช้ CPU เป็น แผนภูมิ skyline แบบเคลื่อนไหว

การใช้คำสั่ง time เพื่อประเมินการใช้ microprocessor

ใช้คำสั่ง **time** เพื่อทำความเข้าใจกับลักษณะประสิทธิภาพ ของโปรแกรมเดียวและ synchronous children ของโปรแกรมนั้น

คำสั่ง **time** รายงานเวลา จริง ซึ่งเป็นเวลาที่ผ่านไปตั้งแต่เริ่มต้นจนสิ้นสุดโปรแกรม คำสั่งนี้ยังรายงาน จำนวนของเวลา microprocessor ที่ใช้โดยโปรแกรมด้วย เวลา microprocessor มีการแบ่งเป็น user และ sys ค่า user คือ เวลาที่ใช้โดยตัวโปรแกรมเองและไลบรารีรูทีนย่อยใดๆ ที่โปรแกรมเรียก ค่า sys คือเวลาที่ใช้โดยการเรียกระบบซึ่งเรียกใช้โดยโปรแกรม (ทางตรงหรือทางอ้อม)

ผลรวมของ user + sys คือ ต้นทุน microprocessor โดยตรงทั้งหมดของการดำเนินการโปรแกรม ค่านี้ไม่ได้รวมต้นทุน microprocessor ในส่วนของเคอร์นัลที่อาจกล่าวได้ว่า รันในนามของโปรแกรม แต่ไม่ได้รันจริงบนเธรด ตัวอย่างเช่น ต้นทุนของการขโมย page frames เพื่อแทนที่ page frames ซึ่ง นำมาจากรายการที่ว่าง เมื่อโปรแกรมที่เริ่มต้นไม่ได้มีการรายงานเป็นส่วนประกอบ ของการใช้ microprocessor ของโปรแกรม

บนระบบตัวประมวลผลเดียว ผลต่างระหว่าง เวลา `real` และเวลา microprocessor ทั้งหมด นั่นคือ:

```
real - (user + sys)
```

คือ ผลรวมของปัจจัยทั้งหมดที่อาจทำให้โปรแกรมล่าช้า บวกกับต้นทุนที่ไม่ใช่ลักษณะ ประจำของโปรแกรมเอง บน SMP การประมาณค่าจะเป็นดังนี้:

```
real * number_of_processors - (user + sys)
```

ในลำดับของขนาดที่ลดลงโดยประมาณ ปัจจัยอาจเป็น:

- I/O ที่ต้องใช้ในการนำเข้าข้อความและข้อมูลของโปรแกรม
- I/O ที่ต้องใช้ในการจัดหาหน่วยความจำจริงสำหรับการใช้งานของโปรแกรม
- เวลา microprocessor ที่ใช้โดยโปรแกรมอื่น
- เวลา microprocessor ที่ใช้โดยระบบปฏิบัติการ

ในตัวอย่างต่อไปนี้ โปรแกรมที่ใช้ในส่วนก่อนหน้านี้ ถูกคอมไพล์เข้ากับ **-O3** แล้วเพื่อทำให้รันได้เร็วขึ้น มีความแตกต่าง เพียงเล็กน้อยเท่านั้นระหว่างเวลาจริง (นาฬิกาแขวนผนัง) ที่ต้องใช้ในการรัน โปรแกรมและผลรวมของเวลา microprocessor ของผู้ใช้และระบบ โปรแกรม ได้รับเวลาทั้งหมดที่ต้องการ ซึ่งอาจทำให้โปรแกรมอื่นในระบบต้องสูญเสียเวลา นั้นไป

```
# time looper
real    0m3.58s
user    0m3.16s
sys     0m0.04s
```

ในตัวอย่างถัดไป เรารันโปรแกรมในระดับความสำคัญน้อยลง โดยการเพิ่ม 10 ที่ค่า nice ของโปรแกรม ใช้เวลาเกือบสองเท่าในการรัน แต่โปรแกรมอื่นยังได้รับโอกาสทำงานของ ตัวเอง:

```
# time nice -n 10 looper
real    0m6.54s
user    0m3.17s
sys     0m0.03s
```

หมายเหตุว่าเราวางคำสั่ง **nice** ภายใน คำสั่ง **time** แทนที่จะเป็นคำสั่ง reverse ถ้าเรา ป้อน

```
# nice -n 10 time looper
```

เราจะได้รับ คำสั่ง **time** ที่แตกต่างไป (**/usr/bin/time**) ที่มีรายงานความแม่นยำน้อยลง แทนเวอร์ชันของคำสั่ง **time** ที่เราเคยใช้ ซึ่งมีการสร้างขึ้นใน **ksh** shell ถ้าคำสั่ง **time** มาก่อน คุณจะได้รับเวอร์ชันในตัว ยกเว้นว่าคุณระบุชื่อแบบเต็มที่ถูกต้องของ **/usr/bin/time** ถ้าคำสั่ง **time** มีการเรียกใช้จากคำสั่งอื่น คุณจะได้รับ **/usr/bin/time**

ข้อควรพิจารณาเกี่ยวกับคำสั่ง time และ timex

ควรพิจารณาข้อเท็จจริงต่างๆ เมื่อคุณใช้คำสั่ง **time** หรือ **timex** อย่างใดอย่างหนึ่ง

ปัจจัยเหล่านี้มีดังนี้:

- ไม่แนะนำให้ใช้คำสั่ง `/usr/bin/time` และ `/usr/bin/timex` ถ้าเป็นไปได้ ให้ใช้คำสั่งย่อย time ของ Korn หรือ C shell
- คำสั่ง **timex -s** ใช้คำสั่ง **sar** เพื่อจัดหาสถิติเพิ่มเติม เนื่องจากคำสั่ง **sar** เป็นแบบแทรกซ้อน คำสั่ง **timex -s** จึงเป็นไปด้วยโดยเฉพาะ สำหรับการรันอย่างคร่าวๆ ข้อมูลที่รายงานโดยคำสั่ง **timex -s** อาจไม่ได้สะท้อนถึงพฤติกรรมของโปรแกรมในระบบที่ไม่มีการมอนิเตอร์อย่างถูกต้อง

- เนื่องจากความยาวของ clock tick ระบบ (10 มิลลิวินาที) และกฎที่ใช้โดย ตัวจัดตารางเวลาในการกำหนดการใช้เวลา CPU ให้กับเธรด ผลของคำสั่ง **time** จึงไม่สามารถกำหนดได้โดยสมบูรณ์ เนื่องจากเวลา ได้มาจากการสุ่มตัวอย่าง จึงมีความแตกต่างที่หลีกเลี่ยงไม่ได้จำนวนมาก ระหว่างการรันที่ต่อเนื่องกัน ความแตกต่างนี้คือในแง่ของ clock ticks เวลารันของโปรแกรม ยิ่งสั้น ความผันแปรเป็นเปอร์เซ็นต์ของผลลัพธ์ที่รายงานยิ่งมาก (โปรดดู “การเข้าถึงตัวจับเวลาโพรเซสเซอร์” ในหน้า 454)
- การใช้คำสั่ง **time** หรือ **timex** (ไม่ว่าจาก `/usr/bin` หรือผ่านทางฟังก์ชัน shell **time** ในตัว) เพื่อวัดเวลาของผู้ใช้หรือระบบของลำดับของคำสั่งที่เชื่อมต่อโดยไปป์ ที่ป้อนบนบรรทัดคำสั่ง ไม่แนะนำให้ทำ ปัญหาหนึ่งที่อาจเกิดขึ้นได้คือ การควบคุมไวยากรณ์อาจทำให้คำสั่ง **time** ประเมินเพียง คำสั่งเดียว โดยไม่มีการบ่งชี้ข้อผิดพลาดของผู้ใช้ ไวยากรณ์ถูกต้องทางเทคนิคเพียงแต่ไวยากรณ์ไม่ได้ให้คำตอบที่ ผู้ใช้ต้องการ
- แม้ว่าไวยากรณ์คำสั่ง **time** ไม่ได้เปลี่ยน แต่เอาต์พุตของไวยากรณ์ มีความหมายเป็นอีกแบบหนึ่งในสภาพแวดล้อม SMP: บน SMP เวลาจริง หรือเวลาที่ผ่านไปอาจน้อยกว่าเวลาผู้ใช้ของกระบวนการ ขณะนี้ เวลาผู้ใช้ คือผลรวมของเวลาทั้งหมดที่ใช้โดยเธรดหรือกระบวนการบนตัวประมวลผล ทั้งหมด

  ถ้ากระบวนการหนึ่งมีเธรดสี่รายการ การรันกระบวนการนั้นบนระบบ ตัวประมวลผลเดียว (UP) แสดงว่าเวลาจริงมากกว่าเวลาผู้ใช้:

  ```
  # time 4threadedprog
  real    0m11.70s
  user    0m11.09s
  sys     0m0.08s
  ```

  การรันบนระบบ 4-way SMP อาจแสดงว่า เวลาจริงมีเพียงประมาณ 1/4 ของเวลาผู้ใช้ เอาต์พุตต่อไปนี้ แสดงว่ากระบวนการมัลติเธรดแจกจ่ายเวิร์กโหลดบนหลายตัวประมวลผล และพัฒนาเวลาการดำเนินการจริง ดังนั้น ผลผลิตของระบบ จึงเพิ่มขึ้น

  ```
  # time 4threadedprog
  real    0m3.40s
  user    0m9.81s
  sys     0m0.09s
  ```

identification ของโปรแกรมที่ใช้ไมโครโพรเซสเซอร์

เพื่อหาตำแหน่งของการประมวลผลที่มีอิทธิพลต่อการใช้ไมโครโพรเซสเซอร์ จึงมีเครื่องมือมาตรฐานอยู่สองแบบ คือ คำสั่ง **ps** และคำสั่ง **acctcom**

เครื่องมืออื่นๆที่ใช้คือ มอนิเตอร์ **topas** ซึ่งกล่าวถึงใน “การมอนิเตอร์ผลการทำงานของระบบอย่างต่อเนื่องด้วยคำสั่ง topas” ในหน้า 18

การใช้คำสั่ง ps

คำสั่ง **ps** เป็นเครื่องมือที่ยืดหยุ่นสำหรับการระบุโปรแกรม ที่กำลังรันอยู่บนระบบ และรีซอร์สที่โปรแกรมใช้อยู่ คำสั่งแสดงข้อมูลสถิติและสถานะเกี่ยวกับ กระบวนการบนระบบ เช่น ID กระบวนการหรือเธรด, กิจกรรม I/O, CPU และการใช้งานหน่วยความจำ

ในส่วนนี้ อ็อพชันและฟิลด์เอาต์พุตที่สัมพันธ์ กับ CPU จะถูกอภิปราย

สามคอลัมน์ของเอาต์พุตคอลัมน์ **ps** ที่เป็นไปได้รายงาน การใช้งาน CPU แต่ละค่าด้วยวิธีต่างกัน

คอลัมน์

ค่าคือ:

C เวลา CPU ที่ใช้ล่าสุดสำหรับกระบวนการ (ในหน่วยของนาฬิกา)

TIME เวลา CPU ทั้งหมดที่ใช้โดยกระบวนการตั้งแต่กระบวนการเริ่มต้นขึ้น (ในหน่วย นาทีและวินาที)

%CPU เวลา CPU ทั้งหมดที่ใช้โดยกระบวนการตั้งแต่กระบวนการเริ่มต้นขึ้น หารด้วย เวลาที่ผ่านไปตั้งแต่กระบวนการเริ่มต้นขึ้น เวลานี้เป็นการวัด การขึ้นต่อกันของ CPU ของโปรแกรม

การเน้น CPU

Shell สคริปต์ดังต่อไปนี้:

```
# ps -ef | egrep -v "STIME|$LOGNAME" | sort +3 -r | head -n 15
```

คือ เครื่องมือสำหรับการเน้นที่กระบวนการผู้ใช้ที่เน้น CPU ที่ใช้ขณะนี้ สูงสุดในระบบ (บรรทัดส่วนหัวถูกแทรกใหม่เพื่อความชัดเจน):

```
     UID   PID  PPID   C    STIME    TTY  TIME CMD
    mary 45742 54702 120 15:19:05 pts/29  0:02 ./looper
    root 52122     1  11 15:32:33 pts/31 58:39 xhogger
    root  4250     1   3 15:32:33 pts/31 26:03 xmconsole allcon
    root 38812  4250   1 15:32:34 pts/31  8:58 xmconstats 0 3 30
    root 27036  6864   1 15:18:35      -  0:00 rlogind
    root 47418 25926   0 17:04:26      -  0:00 coelogin <d29dbms:0>
    bick 37652 43538   0 16:58:40  pts/4  0:00 /bin/ksh
    bick 43538     1   0 16:58:38      -  0:07 aixterm
     luc 60062 27036   0 15:18:35 pts/18  0:00 -ksh
```

คอลัมน์ (C) ระบุ CPU ที่ใช้ขณะนี้ กระบวนการของ โปรแกรมการวนซ้ำนำหน้าในรายการ ค่า C สามารถลด การใช้งาน CPU ของกระบวนการวนลูปเนื่องจากตัวกำหนดตารางเวลาหยุดการนับ ที่ 120 สำหรับกระบวนการแบบมัลติเธรด ฟิลด์นี้ระบุผลรวม ของ CP ที่แสดงรายการสำหรับเธรดทั้งหมดภายในกระบวนการนั้น

ตัวอย่าง ต่อไปนี้แสดงโปรแกรมห้าเธรดแบบง่ายที่มีเธรด ทั้งหมดในโปรแกรมการวนรอบไม่รู้จบ:

```
ps -lmo THREAD -p 8060956
    USER PID   PPID  TID   ST CP PRI SC WCHAN F      TT    BND COMMAND
    root 8060956 6815882         - A  720 120  0        -   200001  pts/0   - ./a.out
       -       -       -   8716483 R  120 120  0        -   400000      -   - -
       -       -       -  17105017 R  120 120  0        -   400000      -   - -
       -       -       -  24182849 R  120 120  0        -   400000      -   - -
       -       -       -  24510589 R  120 120  0        -   400000      -   - -
       -       -       -  30277829 R  120 120  0        -   400000      -   - -
       -       -       -  35913767 R  120 120  0        -   400000      -   - -
```

ใน คอลัมน์ CP ค่า 720 ระบุผลรวมของแต่ละเธรด ที่แสดงรายการด้านล่างค่านี้ นั่นคือ: (5 * 120) + (120)

อัตราส่วนเวลา CPU

คำสั่ง **ps** ที่รันเป็นครั้งคราว แสดงเวลา CPU ภายใต้คอลัมน์ TIME และอัตราส่วน ของเวลา CPU ต่อเวลาจริงภายใต้คอลัมน์ %CPU ให้ดูกระบวนการ ที่มีการใช้สูงสุด อ็อพชัน **au** และ **v** ให้ข้อมูล ที่คล้ายกันเกี่ยวกับกระบวนการผู้ใช้ อ็อพชัน **aux** และ **vg** แสดง ทั้งกระบวนการผู้ใช้และระบบ

ตัวอย่างต่อไปนี้นำมาจากระบบ four-way SMP:

```
# ps au
USER       PID %CPU %MEM   SZ  RSS    TTY STAT    STIME TIME COMMAND
root     19048 24.6  0.0   28   44  pts/1 A    13:53:00 2:16 /tmp/cpubound
root     19388  0.0  0.0  372  460  pts/1 A      Feb 20 0:02 -ksh
root     15348  0.0  0.0  372  460  pts/4 A      Feb 20 0:01 -ksh
root     20418  0.0  0.0  368  452  pts/3 A      Feb 20 0:01 -ksh
root     16178  0.0  0.0  292  364      0 A      Feb 19 0:00 /usr/sbin/getty
root     16780  0.0  0.0  364  392  pts/2 A      Feb 19 0:00 -ksh
root     18516  0.0  0.0  360  412  pts/0 A      Feb 20 0:00 -ksh
root     15746  0.0  0.0  212  268  pts/1 A    13:55:18 0:00 ps au
```

%CPU คือ เปอร์เซ็นต์ของเวลา CPU ที่จัดสรรให้กับกระบวนการนั้นตั้งแต่ กระบวนการเริ่มต้นขึ้น มีการคำนวณดังนี้:

```
(เวลา CPU กระบวนการ / ช่วงเวลากระบวนการ) * 100
```

ลองคิดว่า มีสองกระบวนการ: กระบวนการแรกเริ่มทำงาน และรันห้าวินาที แต่ยังไม่ เสร็จ จากนั้นกระบวนการที่สองเริ่มทำงาน และรันห้าวินาที และไม่ เสร็จ ขณะนี้คำสั่ง **ps** จะแสดง %CPU 50% สำหรับ กระบวนการแรก (CPU ห้าวินาทีสำหรับ 10 วินาที ของเวลาที่ผ่านไป) และ 100 สำหรับกระบวนการที่สอง (CPU ห้าวินาทีสำหรับห้าวินาที ของเวลาที่ผ่านไป)

บน SMP ค่านี้มีการหาร ด้วยจำนวนของ CPUs ที่มีอยู่บนระบบ ลองกลับไปดูตัวอย่างก่อนหน้านี้ นี้คือเหตุผลว่าทำไมค่า %CPU สำหรับกระบวนการ **cpubound** จึงไม่เกิน 25 เนื่องจากตัวอย่างถูกรับนระบบตัวประมวลผล สี่วิธี กระบวนการ **cpubound** ใช้ 100 เปอร์เซ็นต์ของตัวประมวลผลหนึ่ง แต่ค่า %CPU ถูกหารโดยจำนวนของ CPUs ที่มีอยู่

THREAD อ็อพชัน

คำสั่ง **ps** สามารถแสดง เธรดและ CPUs ที่เธรดหรือกระบวนการถูกยึดไว้โดย การใช้คำสั่ง **ps -mo THREAD** ข้อมูลต่อไปนี้เป็นตัวอย่าง:

```
# ps -mo THREAD
USER PID   PPID  TID   ST CP PRI SC WCHAN F      TT    BND COMMAND
root 20918 20660 -     A  0  60  1  -     240001 pts/1 -   -ksh
-    -     -     20005 S  0  60  1  -     400    -     -   -
```

คอลัมน์ TID แสดง เธรด ID, คอลัมน์ BND แสดงกระบวนการและเธรดที่ยึดไว้กับ ตัวประมวลผล

โดยปกติ จะเห็นกระบวนการที่มีชื่อว่า kproc (PID จำนวน 516 ในระบบปฏิบัติการเวอร์ชัน 4) ที่ใช้เวลา CPU เมื่อไม่มีเธรดที่สามารถรันในระหว่างช่วงเวลา ตัวจัดตารางเวลาจะกำหนดเวลา CPU สำหรับ ช่วงเวลานั้นให้กับกระบวนการเคอร์นัลนี้ (kproc) ซึ่งรู้จักกันว่าเป็น *idle* หรือ *wait* kproc ระบบ SMP มี *idle* kproc สำหรับแต่ละตัวประมวลผล

สำหรับรายละเอียดแบบสมบูรณ์ เกี่ยวกับคำสั่ง **ps** ดูที่ in *Commands Reference*

การใช้คำสั่ง acctcom

คำสั่ง **acctcom** จะแสดงข้อมูลประวัติเกี่ยวกับการใช้งาน CPU หากเรียกใช้งานระบบบัญชีผู้ใช้

เรียกทำงานบัญชีผู้ใช้หากมีความต้องการที่แน่นอน เนื่องจากจุดเริ่มต้นของระบบบัญชีผู้ใช้จะวางการใช้งานที่สามารถวัดค่าได้บนระบบ หากต้องการเรียกใช้งานระบบบัญชีผู้ใช้ ให้ทำดังต่อไปนี้:

1. สร้างไฟล์บัญชีผู้ใช้ที่ว่างเปล่า:

   ```
   # touch acctfile
   ```

2. เปิดใช้บัญชีผู้ใช้:

   ```
   # /usr/sbin/acct/accton acctfile
   ```

3. อนุญาตให้บัญชีผู้ใช้รันชั่วขณะ จากนั้นปิดบัญชีผู้ใช้:

   ```
   # /usr/sbin/acct/accton
   ```

4. แสดงการดักจับของบัญชีผู้ใช้ ดังต่อไปนี้:

   ```
   # /usr/sbin/acct/acctcom acctfile
   COMMAND                     START    END          REAL      CPU     MEAN
   NAME       USER     TTYNAME  TIME     TIME       (SECS)   (SECS)  SIZE(K)
   #accton    root     pts/2   19:57:18 19:57:18     0.02     0.02   184.00
   #ps        root     pts/2   19:57:19 19:57:19     0.19     0.17    35.00
   #ls        root     pts/2   19:57:20 19:57:20     0.09     0.03   109.00
   #ps        root     pts/2   19:57:22 19:57:22     0.19     0.17    34.00
   #accton    root     pts/2   20:04:17 20:04:17     0.00     0.00     0.00
   #who       root     pts/2   20:04:19 20:04:19     0.02     0.02     0.00
   ```

ถ้าคุณนำไฟล์เดิมกลับมาใช้ใหม่ คุณสามารถมองเห็น เมื่อการประมวลผลใหม่เริ่มต้นขึ้น โดยการมองหาการประมวลผล **accton** (ซึ่งเป็นการประมวลผล ที่ใช้เพื่อปิดบัญชีผู้ใช้ในครั้งแรก)

การใช้คำสั่ง pprof เพื่อประเมินการใช้ microprocessor ของเคอร์นัลเธรด

คำสั่ง **pprof** รายงานการใช้ microprocessor บนเคอร์นัลเธรดทั้งหมดที่กำลังรันภายในช่วงเวลาโดยใช้ยูทิลิตีการติดตาม

ข้อมูลกระบวนการ raw มีการบันทึกลงใน `pprof.flow` และ สร้างรายงานขึ้นห้าฉบับ ถ้าไม่มีการระบุแฟล็ก จะมีการสร้างรายงานทั้งหมด

เมื่อต้องการกำหนดว่าโปรแกรม **pprof** มีการติดตั้งและมีอยู่ หรือไม่ ให้รันคำสั่งต่อไปนี้:

```
# lslpp -lI bos.perf.tools
```

ชนิด ของรายงานมีดังนี้:

pprof.cpu

แสดงรายการเธรดระดับเคอร์นัลทั้งหมดที่เรียงลำดับตามเวลา microprocessor จริง มี: ชื่อกระบวนการ, ID กระบวนการ, ID กระบวนการพาเรนต์, สถานะกระบวนการเมื่อ เริ่มต้นและสิ้นสุด, Thread ID, Parent Thread ID, เวลา CPU จริง, เวลาเริ่มต้น, เวลาหยุด, หยุด – เริ่มต้น

pprof.famcpu

แสดงข้อมูลเกี่ยวกับตระกูลทั้งหมด (กระบวนการที่มี ancestor เดียวกัน) ชื่อกระบวนการและ ID กระบวนการสำหรับตระกูลไม่จำเป็นต้องเป็น ancestor มี: เวลาเริ่มต้น, ชื่อกระบวนการ, ID กระบวนการ, จำนวนของเธรด, เวลา CPU ทั้งหมด

pprof.famind

แสดงรายการกระบวนการทั้งหมดที่จัดกลุ่มตามตระกูล (กระบวนการที่มี ancestor เดียวกัน) กระบวนการชายน์มีการย่อหน้าตามพาเรนต์ มี: เวลาเริ่มต้น, เวลาหยุด, เวลา CPU จริง, ID กระบวนการ, ID กระบวนการพาเรนต์, Thread ID, Parent Thread ID, สถานะกระบวนการเมื่อเริ่มต้นและสิ้นสุด, ระดับ, ชื่อกระบวนการ

pprof.namecpu

แสดงรายการข้อมูลเกี่ยวกับเคอร์นัลเธรดแต่ละชนิด (ทั้งหมดที่ดำเนินการได้ด้วยชื่อเดียวกัน) มี: ชื่อกระบวนการ, จำนวนของเธรด, เวลา CPU, % ของ เวลา CPU ทั้งหมด

pprof.start

แสดงรายการเคอร์นัลเธรดทั้งหมดที่เรียงลำดับตามเวลาเริ่มต้นซึ่งถูกจัดส่งใน ระหว่างช่วงเวลาคำสั่ง **pprof** มี: ชื่อกระบวนการ, ID กระบวนการ, ID กระบวนการพาเรนต์, สถานะกระบวนการเมื่อเริ่มต้นและสิ้นสุด, Thread ID, Parent Thread ID, เวลา CPU จริง, เวลาเริ่มต้น, เวลาหยุด, หยุด – เริ่มต้น

ข้อมูลต่อไปนี้เป็นไฟล์ pprof.namecpu ตัวอย่าง ที่สร้างขึ้นโดยการรันโปรแกรม **tthreads32** ซึ่ง forks off สี่เธรด ซึ่งแต่ละเธรด fork off กระบวนการของตัวเอง จากนั้นกระบวนการเหล่านี้ ดำเนินการหลายโปรแกรม **ksh** และ **sleep**:

```
                Pprof  PROCESS  NAME  Report

                Sorted  by  CPU  Time

                From: Thu Oct 19 17:53:07 2000
                To:   Thu Oct 19 17:53:22 2000


      Pname #ofThreads  CPU_Time        %
   ======== ==========  ========  ========
 tthreads32         13     0.116    37.935
         sh          8     0.092    30.087
       Idle          2     0.055    17.987
        ksh         12     0.026     8.503
      trace          3     0.007     2.289
       java          3     0.006     1.962
      kproc          5     0.004     1.308
    xmservd          1     0.000     0.000
    trcstop          1     0.000     0.000
    swapper          1     0.000     0.000
        gil          1     0.000     0.000
         ls          4     0.000     0.000
      sleep          9     0.000     0.000
         ps          4     0.000     0.000
    syslogd          1     0.000     0.000
       nfsd          2     0.000     0.000
            ==========  ========  ========
                   70     0.306   100.000
```

pprof.cpu ที่ตรงกันเป็นดังนี้:

```
                Pprof CPU Report

                Sorted  by  Actual CPU  Time

                From: Thu Oct 19 17:53:07 2000
                To:   Thu Oct 19 17:53:22 2000


E = Exec'd      F = Forked
X = Exited      A = Alive (when traced started or stopped)
C = Thread Created
```

```
      Pname      PID     PPID  BE      TID     PTID  ACC_time  STT_time  STP_time   STP-STT
      =====    =====    ===== ===    =====    =====  ========  ========  ========  ========
       Idle      774        0  AA      775        0     0.052     0.000     0.154     0.154
tthreads32     5490    11982  EX    18161    22435     0.040     0.027     0.154     0.126
         sh    11396     5490  EE    21917     5093     0.035     0.082     0.154     0.072
         sh    14106     5490  EE    16999    18867     0.028     0.111     0.154     0.043
         sh    13792     5490  EE    20777    18179     0.028     0.086     0.154     0.068
        ksh     5490    11982  FE    18161    22435     0.016     0.010     0.027     0.017
tthreads32     5490    11982  CX     5093    18161     0.011     0.056     0.154     0.098
tthreads32     5490    11982  CX    18179    18161     0.010     0.054     0.154     0.099
tthreads32    14506     5490  FE    17239    10133     0.010     0.128     0.143     0.015
       ksh    11982    13258  AA    22435        0     0.010     0.005     0.154     0.149
tthreads32    13792     5490  FE    20777    18179     0.010     0.059     0.086     0.027
tthreads32     5490    11982  CX    18867    18161     0.010     0.057     0.154     0.097
tthreads32    11396     5490  FE    21917     5093     0.009     0.069     0.082     0.013
tthreads32     5490    11982  CX    10133    18161     0.008     0.123     0.154     0.030
tthreads32    14106     5490  FE    16999    18867     0.008     0.088     0.111     0.023
     trace     5488    11982  AX    18159        0     0.006     0.001     0.005     0.003
     kproc     1548        0  AA     2065        0     0.004     0.071     0.154     0.082
      Idle      516        0  AA      517        0     0.003     0.059     0.154     0.095
      java    11612    11106  AA    14965        0     0.003     0.010     0.154     0.144
      java    11612    11106  AA    14707        0     0.003     0.010     0.154     0.144
     trace    12544     5488  AA    20507        0     0.001     0.000     0.001     0.001
        sh    14506     5490  EE    17239    10133     0.001     0.143     0.154     0.011
     trace    12544     5488  CA    19297    20507     0.000     0.001     0.154     0.153
       ksh     4930     2678  AA     5963        0     0.000     0.154     0.154     0.000
     kproc     6478        0  AA     3133        0     0.000     0.154     0.154     0.000
        ps    14108     5490  EX    17001    18867     0.000     0.154     0.154     0.000
tthreads32    13794     5490  FE    20779    18179     0.000     0.154     0.154     0.000
        sh    13794     5490  EE    20779    18179     0.000     0.154     0.154     0.000
        ps    13794     5490  EX    20779    18179     0.000     0.154     0.154     0.000
        sh    14108     5490  EE    17001    18867     0.000     0.154     0.154     0.000
tthreads32    14108     5490  FE    17001    18867     0.000     0.154     0.154     0.000
        ls    13792     5490  EX    20777    18179     0.000     0.154     0.154     0.000
:
:
:
```

การตรวจพบอีมูเลชันของคำสั่งด้วยเครื่องมือ emstat

หากต้องการรักษาความเข้ากันได้กับไบนารีเก่า เคอร์เนล AIX ต้องสอดแทรกรูทีนอีมูเลชันที่จัดเตรียมส่วนสนับสนุนสำหรับคำสั่งที่อาจไม่ได้สอดแทรกไว้ในสถาปัตยกรรมของชิปโดยเฉพาะ ความพยายามในการเรียกใช้งานคำสั่งที่ไม่ได้รับการสนับสนุน จะส่งผลทำให้เกิดข้อยกเว้นของคำสั่งที่ผิดกฎเกณฑ์ เคอร์เนลจะถอดรหัสคำสั่งแบบผิดกฎเกณฑ์ และถ้าคำสั่งนั้นไม่ได้รับการสนับสนุน เคอร์เนลจะรันรูทีนอีมูเลชันที่เลียนแบบคำสั่ง

ขึ้นอยู่กับความถี่ในการประมวลผลของคำสั่งที่ไม่ได้รับการสนับสนุน และความยาวพาธของอีมูเลชัน อีมูเลชันสามารถทำให้เกิดการผันแปรในระดับของผลการทำงานที่ลดลง เนื่องจากการใช้ context switch ของเคอร์เนล อีมูเลชันของคำสั่ง แม้ว่าเปอร์เซ็นต์ของอีมูเลชันที่น้อยมากอาจส่งผลทำให้เกิดความแตกต่างในเรื่องของ ผลการทำงานที่เป็นเรื่องใหญ่ก็ตาม ตารางต่อไปนี้แสดงความยาวพาธของคำสั่งที่ประมาณการ สำหรับคำสั่งที่ไม่ได้รับการสนับสนุนทั้งหลาย:

คำสั่ง	เลียนแบบใน	ความยาวพาธที่ประมาณการ (คำสั่ง)
abs	assembler	117
doz	assembler	120
mul	assembler	127
rlmi	C	425
sle	C	447
clf	C	542
div	C	1079

คำสั่งที่ไม่ได้อยู่บนแพล็ตฟอร์มทั่วไปทั้งหมด ต้องถูกลบออกจากโค้ดที่เขียนในแอสเซมเบลอร์ เนื่องจากการคอมไพล์ใหม่จะมีผลต่อซอร์สโค้ด ในระดับสูง รูทีนที่โค้ดในแอสเซมเบลอร์ต้องถูกเปลี่ยนแปลง เพื่อไม่ใช้ใช้คำสั่งที่ไม่มีอยู่ เนื่องจากการคอมไพล์ใหม่ อาจไม่ส่งผลในกรณีนี้

ขั้นตอนแรกคือ การพิจารณาว่า อีมูเลชันของคำสั่งกำลังเกิดขึ้น โดยใช้เครื่องมือ **emstat**

หากต้องการพิจารณาว่า โปรแกรม **emstat** ได้ถูกติดตั้งและพร้อมใช้งานแล้ว ให้รันคำสั่งต่อไปนี้:

```
# lslpp -lI bos.perf.tools
```

คำสั่ง **emstat** จะทำงานคล้ายกับคำสั่ง **vmstat** ตรงที่ว่า คุณระบุช่วงเวลาในหน่วยวินาที และเลือกที่จะระบุจำนวนของช่วงเวลา ค่าที่อยู่ในคอลัมน์แรก คือจำนวนสะสมเนื่องจากระบบบูต ขณะที่ค่าในคอลัมน์ที่สองคือ จำนวนของคำสั่งที่ถูกเลียนแบบ ในช่วงเวลา อีมูเลชันตามลำดับของจำนวนพันต่อวินาที สามารถมีผลกระทบต่อผลการทำงาน

ต่อไปนี้คือตัวอย่างของเอาต์พุตจากการออกคำสั่ง **emstat 1** :

```
# emstat 1

Emulation   Emulation
SinceBoot       Delta
        0           0
        0           0
        0           0
```

อีมูเลชันได้ถูกตรวจพบ ขั้นตอนนี้มีไว้เพื่อพิจารณาถึงแอ็พพลิเคชันที่กำลังเลียนแบบคำสั่ง นี่คือส่วนที่ยากในการพิจารณา วิธีหนึ่งคือ การรันหนึ่งแอ็พพลิเคชันในหนึ่งครั้ง และมอนิเตอร์ด้วยโปรแกรม **emstat** บางครั้ง อีมูเลชันจะเป็นสาเหตุทำให้เกิด hook ของการติดตาม สิ่งนี้จะสามารถมองเห็นได้ในไฟล์รายงานการติดตาม ASCII พร้อมกับคำว่า PROGRAM CHECK การประมวลผล/เธรด ที่เชื่อมโยงกับเหตุการณ์การติดตามนี้กำลังเลียนแบบคำสั่ง เนื่องจากเป็นโค้ดการเลียนแบบคำสั่งของตนเอง หรือกำลังประมวลผลไลบรารี หรือโค้ดอยู่ในโมดูลอื่นที่กำลังเลียนแบบคำสั่ง

การตรวจบกข้อยกเว้นในการจัดตำแหน่งด้วยเครื่องมือ alstat

การจัดตำแหน่งของข้อมูลที่ผิดอาจเป็นสาเหตุทำให้ฮาร์ดแวร์สร้างข้อยกเว้น ในการจัดตำแหน่ง

คอมไพลเลอร์ AIX จะดำเนินการจัดตำแหน่งของชนิดข้อมูลตามลักษณะการทำงาน ตัวอย่างเช่น ข้อมูลที่มีชนิดสั้น ซึ่งมีความยาวขนาด 2 ไบต์ จะถูกเติมเต็มให้เป็น 4 ไบต์โดยอัตโนมัติโดยคอมไพลเลอร์ การเขียนโปรแกรมมิงทั่วไป เช่น การแปลงชนิด

และการใช้โปรแกรมการจัดตำแหน่งสามารถทำให้ข้อมูลแอ็พพลิเคชัน ถูกจัดตำแหน่งไว้อย่างไม่ถูกต้อง การ optimization POWER® processor-based จะยอมรับการจัดตำแหน่งข้อมูลที่ถูกต้อง ดังนั้น การดึงข้อมูลที่จัดตำแหน่งผิดอาจต้องการเข้าถึงหน่วยความจำจำนวนมาก ที่การเข้าถึงเดี่ยวควรจะมีเพียงพอ ข้อยกเว้นในการจัดตำแหน่ง ที่สร้างโดยการจัดตำแหน่งข้อมูลที่ไม่ถูกต้องจะบังคับให้เคอร์เนล จำลองการเข้าถึงหน่วยความจำที่ต้องการ เนื่องจากในกรณีของอีมูเลชันของคำสั่ง เหตุการณ์นี้จะลดระดับผลการทำงานของแอ็พพลิเคชันลง

เครื่องมือ **alstat** ที่ทำแพ็กเกจด้วย `bos.perf.tools` สามารถนำมาใช้เพื่อตรวจสอบ ข้อยกเว้นการจัดตำแหน่งที่เกิดขึ้น หากต้องการแสดงข้อยกเว้นในการจัดตำแหน่งบนพื้นฐานต่อ CPU ให้ใช้อ็อพชัน **-v**

เนื่องจาก **alstat** และ **emstat** คือไบนารีเดียวกัน หนึ่งในเครื่องมือเหล่านี้สามารถนำมาใช้เพื่อแสดงอีมูเลชันของคำสั่ง และข้อยกเว้นการจัดตำแหน่ง หากต้องการแสดงอีมูเลชันของคำสั่ง ให้ใช้อ็อพชัน **-e** บน **alstat** หากต้องการแสดงข้อยกเว้นในการจัดตำแหน่ง ให้ใช้อ็อพชัน **-a** บน **emstat**

เอาต์พุตสำหรับ **alstat** จะดูคล้ายกับที่แสดงดังต่อไปนี้:

```
# alstat -e 1
  Alignment  Alignment  Emulation  Emulation
  SinceBoot      Delta  SinceBoot      Delta
          0          0          0          0
          0          0          0          0
          0          0          0          0
```

การสร้างโปรแกรมที่ดำเนินการได้ขึ้นใหม่ด้วยโปรแกรม fdpr

โปรแกรม **fdpr** (การสร้างโปรแกรมตามข้อมูลข้อคิดเห็น ขึ้นใหม่) ช่วยให้โมดูลที่ดำเนินการได้สามารถดำเนินการได้เร็วขึ้นและการใช้หน่วยความจำจริงมีประสิทธิภาพมากขึ้น

เมื่อต้องการกำหนดว่าโปรแกรม **fdpr** มีการติดตั้งและมีอยู่ บนระบบของคุณหรือไม่ ให้รันคำสั่งต่อไปนี้:

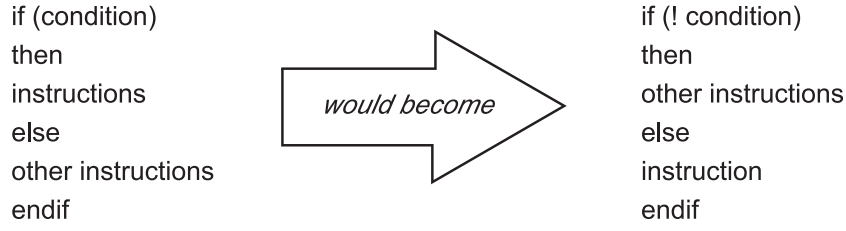
```
# lslpp -lI perfagent.tools
```

คำสั่ง **fdpr** คือ ยูทิลิตีการปรับประสิทธิภาพที่สามารถพัฒนาทั้งประสิทธิภาพและการใช้ประโยชน์ หน่วยความจำจริงของแอ็พพลิเคชันโปรแกรมระดับผู้ใช้ ไม่จำเป็น ต้องอินพุตซอร์สโค้ดในโปรแกรม **fdpr** อย่างไรก็ตาม โปรแกรมที่ดำเนินการได้แบบ stripped ไม่ได้รับการสนับสนุน ถ้ามีซอร์สโค้ด อยู่โปรแกรมที่สร้างขึ้นด้วยคอมไพเลอร์แฟล็ก **-qfdpr** จะมีข้อมูลที่ช่วยโปรแกรม **fdpr** ในการจัดทำ โปรแกรมที่จัดลำดับใหม่ซึ่งมีฟังก์ชันที่รับประกันได้ ถ้าใช้ แฟล็ก **-qfdpr** ควรจะใช้สำหรับอ็อบเจ็กต์โมดูลทั้งหมด ในโปรแกรม การลิงก์แบบสแตติกจะไม่ช่วยปรับปรุงประสิทธิภาพถ้ามีการใช้แฟล็ก **-qfdpr**

เครื่องมือ **fdpr** จัดลำดับคำสั่งในโปรแกรมที่ดำเนินการได้ใหม่ เพื่อปรับปรุงแคชคำสั่ง, Translation Lookaside Buffer (TLB), และการใช้ประโยชน์หน่วยความจำจริงโดยการทำดังต่อไปนี้:

- การรวมลำดับโค้ดที่ดำเนินการได้สูงไว้ด้วยกัน (ตามที่กำหนดผ่านทาง การจัดทำโปรไฟล์)
- การเข้ารหัส branches แบบมีเงื่อนไขอีกครั้งเพื่อพัฒนาการคาดการณ์ฮาร์ดแวร์ branch
- การย้ายโค้ดที่ดำเนินการไม่บ่อยออกจากสายงาน

ตัวอย่างเช่น ถ้ากำหนดข้อความ "if-then-else" โปรแกรม **fdpr** อาจสรุปว่าโปรแกรมจะใช้ else branch บ่อยกว่า if branch จากนั้น โปรแกรมจะกลับเงื่อนไขและสอง branches จะเป็นดังแสดงในรูปภาพต่อไปนี้



รูปที่ 16. ตัวอย่างของการเข้ารหัส Branch แบบมีเงื่อนไขอีกครั้ง. ภาพสาธิตแสดงให้เห็นว่าการเข้ารหัส branch แบบมีเงื่อนไขอีกครั้งเปลี่ยนโค้ดได้อย่างไร ตัวอย่างเช่น โค้ด If (เงื่อนไข) จะกลายเป็น If (! เงื่อนไข); โค้ดยังคงเป็นโค้ด; คำสั่ง กลายเป็นคำสั่งอื่น; else ยังคงเป็น else; คำสั่งอื่นกลายเป็นคำสั่ง ; และ endif ยังคงเป็น endif

แอ็พพลิเคชันขนาดใหญ่ (ใหญ่กว่า 5 MB) ที่ยึดไว้กับ CPU สามารถปรับปรุงเวลาการดำเนินการได้มากถึง 23 เปอร์เซ็นต์ แต่โดยปกติแล้ว ประสิทธิภาพจะพัฒนาขึ้นระหว่าง 5 ถึง 20 เปอร์เซ็นต์ การลดความต้องการหน่วยความจำจริงสำหรับหน้าข้อความ ของโปรแกรมชนิดนี้อาจสูงถึง 70 เปอร์เซ็นต์ ค่าเฉลี่ยคือระหว่าง 20 ถึง 50 เปอร์เซ็นต์ ตัวเลขนี้ขึ้นอยู่กับพฤติกรรมของแอ็พพลิเคชัน และอ็อพชันการใช้ประโยชน์ที่ออกใช้เมื่อใช้โปรแกรม **fdpr**

การประมวลผล **fdpr** เกิดขึ้นในสามระยะดังนี้:

1. ทดสอบโมดูลที่ดำเนินการได้ซึ่งจะใช้ประโยชน์สูงสุดเพื่อรวบรวมข้อมูลประสิทธิภาพโดยละเอียด
2. รันโมดูลที่ดำเนินการได้ที่ทดสอบในเวิร์กโหลด ที่ผู้ใช้ระบุ และบันทึกข้อมูลประสิทธิภาพจากการรันนั้น
3. ใช้ข้อมูลประสิทธิภาพเพื่อดำเนินกระบวนการใช้ประโยชน์ประสิทธิภาพสูงสุด ซึ่งส่งผลให้ได้โมดูลที่ดำเนินการได้ซึ่งสร้างขึ้นใหม่ ที่ควรจะทำเวิร์กโหลด ที่ใช้รวบรวมข้อมูลก่อนหน้านี้ได้อย่างมีประสิทธิภาพ มากขึ้น สิ่งสำคัญมากคือเวิร์กโหลดที่ใช้ในการดำเนินการ โปรแกรม **fdpr** ควรตรงกับการใช้งานจริงของโปรแกรม ให้มากที่สุด ประสิทธิภาพของโปรแกรมที่ดำเนินการได้ซึ่งสร้างขึ้นใหม่ โดยใช้เวิร์กโหลดที่แตกต่างจากที่ใช้ในการดำเนินการโปรแกรม **fdpr** โดยสิ้นเชิง เป็นสิ่งที่ไม่สามารถคาดการณ์ได้ แต่อาจ เลวร้ายกว่าประสิทธิภาพของโปรแกรมที่ดำเนินการได้ดั้งเดิม

ตามตัวอย่าง คำสั่ง **# fdpr -p ProgramName -R3 -x test.sh** จะใช้กรณีทดสอบ `test.sh` เพื่อรันรูปแบบทดสอบ ของโปรแกรม **ProgramName** เอาต์พุตของการรันนั้นจะใช้ในการจัดทำ การใช้ประโยชน์เชิงรุกสูงสุด (**-R3**) ของโปรแกรม ที่จะสร้างโมดูลใหม่ที่มีชื่อดีฟอลต์ว่า `ProgramName.fdpr` ระดับซึ่งโปรแกรมที่ดำเนินการได้และใช้ประโยชน์สูงสุดจะทำงานการผลิตได้ดีกว่า โปรแกรมที่ไม่ได้ใช้ประโยชน์สูงสุด จะขึ้นอยู่กับว่ากรณีทดสอบ `test.sh` สามารถเลียนแบบ เวิร์กโหลดการผลิตได้ดีมากเพียงใด

หมายเหตุ: โปรแกรม **fdpr** ใช้ขั้นตอนวิธีการใช้ประโยชน์สูงสุดขั้นสูง ซึ่งบางครั้งอาจส่งผลให้โปรแกรมที่ดำเนิน การได้และใช้ประโยชน์สูงสุดไม่ทำฟังก์ชันในวิธีเดียวกันกับโมดูล ดั้งเดิม ดังนั้นจึง *จำเป็นอย่างยิ่ง* ที่โปรแกรมที่ดำเนินการได้และใช้ประโยชน์สูงสุดต้องผ่านการทดสอบทั้งหมด ก่อนจะนำมาใช้ในสภาพการผลิตใดๆ นั่นคือก่อนที่เอาต์พุตจะเชื่อถือได้

โดย สรุป ผู้ใช้โปรแกรม **fdpr** ควรปฏิบัติดัง ต่อไปนี้:

- ใช้เวิร์กโหลดซึ่งแสดงถึงการใช้งานที่ตั้งใจไว้ในการดำเนินการโปรแกรม **fdpr**
- ทดสอบฟังก์ชันของโปรแกรมที่ดำเนินการได้ซึ่งสร้างขึ้นใหม่ที่ได้ โดยละเอียด
- ใช้โปรแกรมที่ดำเนินการได้ซึ่งสร้างขึ้นใหม่เฉพาะบนเวิร์กโหลดที่ใช้ในการ ปรับเท่านั้น

การควบคุม contention สำหรับไมโครโพรเซสเซอร์

แม้ว่าเคอร์เนล AIX จะจัดส่งเธรดไปยังตัวประมวลผลที่หลากหลาย เครื่องมือในการจัดการกับระบบส่วนใหญ่จะอ้างถึงการประมวลผลที่มีเธรดกำลังรันอยู่ แทนที่จะจัดการกับตัวเธรดเอง

การควบคุมระดับความสำคัญของการประมวลผลของผู้ใช้

ระดับความสำคัญของการประมวลผลของผู้ใช้สามารถจัดการได้โดยใช้คำสั่ง **nice** หรือ **renice** หรือรูทีนย่อย **setpri( )** และจะถูกแสดงด้วยคำสั่ง **ps**

ภาพรวมของระดับความสำคัญจะถูกจัดเตรียมไว้ใน “ระดับความสำคัญของกระบวนการและเธรด” ในหน้า 45

การคำนวณระดับความสำคัญจะนำมาใช้เพื่อให้บรรลุเป้าหมายต่อไปนี้:

- แบ่งใช้ CPU ระหว่างเธรด
- ป้องกันความอยากของเธรดใดๆ
- ปรับเธรดที่ผูกกับการคำนวณ
- เพิ่มการแบ่งแยกที่ต่อเนื่องกันระหว่างเธรดที่ล่วงเวลา

การรันคำสั่งโดยใช้คำสั่ง nice:

ผู้ใช้ใดๆ สามารถรันคำสั่งที่ระดับความสำคัญต่ำกว่าปกติได้โดยใช้คำสั่ง **nice**

เฉพาะผู้ใช้รากเท่านั้นที่สามารถใช้คำสั่ง **nice** เพื่อรัน คำสั่งต่างๆ ที่ระดับความสำคัญต่ำกว่าปกติ ในกรณีนี้ ช่วงค่าคำสั่ง **nice** อยู่ระหว่าง –20 ถึง 19

ด้วยคำสั่ง **nice** ผู้ใช้ระบุค่าที่จะเพิ่มหรือลบออกจากค่า `nice` มาตรฐาน ค่า `nice` ที่แก้ไขใช้สำหรับกระบวนการที่รัน คำสั่งที่ระบุระดับความสำคัญของกระบวนการยังคงไม่แน่นอน นั่นคือ ค่าระดับความสำคัญยังคงมีการคำนวณใหม่เป็นระยะๆ ตามข้อมูลการใช้ CPU, ค่า `nice`, และค่าระดับความสำคัญต่ำสุดของกระบวนการผู้ใช้

ค่า `nice` มาตรฐานของกระบวนการพื้นหน้าคือ 20 (24 สำหรับ กระบวนการพื้นหลัง ksh) คำสั่งต่อไปนี้จะทำให้คำสั่ง **vmstat** รัน ในพื้นหน้าโดยมีค่า `nice` เป็น 25 (แทนที่จะเป็นค่ามาตรฐาน 20) ส่งผลให้ระดับความสำคัญต่ำกว่าความต้องการ

```
# nice -n 5 vmstat 10 3 > vmstat.out
```

ถ้า คุณใช้ล็อกอินราก คำสั่ง **vmstat** สามารถรันที่ระดับความสำคัญ สูงขึ้นด้วยข้อมูลต่อไปนี้:

```
# nice -n -5 vmstat 10 3 > vmstat.out
```

ถ้า คุณไม่ได้ใช้ล็อกอินรากและออกใช้คำสั่ง **nice** เช่นในตัวอย่างก่อนหน้า นี้ คำสั่ง **vmstat** จะยังคงรันได้แต่รันด้วยค่า `nice` มาตรฐาน 20 และคำสั่ง **nice** จะไม่ออก ข้อความแสดงข้อผิดพลาด

การตั้งค่าระดับความสำคัญคงที่ด้วยรูทีนย่อย setpri:

แอ็พพลิเคชันที่รันภายใต้ ID ผู้ใช้รากสามารถใช้รูทีนย่อย **setpri( )** เพื่อตั้งค่าระดับความสำคัญของตนเองหรือของกระบวนการอื่น

ตัวอย่างเช่น:

```
retcode = setpri(0,59);
```

จะให้ระดับความสำคัญคงที่ 59 แก่กระบวนการปัจจุบัน ถ้ารูทีนย่อย **setpri( )** ล้มเหลว จะส่งคืน –1

โปรแกรมต่อไปนี้ยอมรับค่าระดับความสำคัญ และรายการของ IDs กระบวนการ และตั้งค่าระดับความสำคัญของกระบวนการทั้งหมดเป็นค่าที่ระบุ

```
/*
   fixprocpri.c
   Usage: fixprocpri priority PID . . .
*/

#include <sys/sched.h>
#include <stdio.h>
#include <sys/errno.h>

main(int argc,char **argv)
{
   pid_t ProcessID;
   int Priority,ReturnP;

   if( argc < 3 ) {
      printf(" usage - setpri priority pid(s) \n");
      exit(1);
   }

   argv++;
   Priority=atoi(*argv++);
   if ( Priority < 50 ) {
      printf(" Priority must be >= 50 \n");
      exit(1);
   }

   while (*argv) {
      ProcessID=atoi(*argv++);
      ReturnP = setpri(ProcessID, Priority);
      if ( ReturnP > 0 )
          printf("pid=%d new pri=%d  old pri=%d\n",
            (int)ProcessID,Priority,ReturnP);
      else {
          perror(" setpri failed ");
            exit(1);
      }
   }
}
```

การแสดงระดับความสำคัญของการประมวลผลด้วยคำสั่ง ps

แฟล็ก **–l** (L ที่เป็นตัวพิมพ์เล็ก) ของคำสั่ง **ps** จะแสดงค่า `nice` และค่าระดับความสำคัญของการประมวลผลที่ระบุ

ตัวอย่างเช่น คุณสามารถแสดงระดับความสำคัญของการประมวลผลทั้งหมด ที่เป็นเจ้าของโดยผู้ใช้ที่กำหนดไว้ด้วยคำสั่งต่อไปนี้:

```
# ps -lu user1
      F S UID  PID PPID   C PRI NI ADDR    SZ    WCHAN    TTY  TIME CMD
 241801 S 200 7032 7286   0  60 20 1b4c   108           pts/2  0:00 ksh
 200801 S 200 7568 7032   0  70 25 2310    88  5910a58  pts/2  0:00 vmstat
 241801 S 200 8544 6494   0  60 20 154b   108           pts/0  0:00 ksh
```

เอาต์พุต จะแสดงผลลัพธ์ของคำสั่ง **nice –n 5** ตามที่กล่าวไว้ก่อนหน้านี้ การประมวลผล 7568 จะมีระดับความสำคัญที่ต่ำกว่าระดับความสำคัญ 70 (คำสั่ง **ps** จะรันโดยเซสชันที่แยกต่างหากในโหมดผู้ใช้ระดับสูง ดังนั้นจึงเป็นการมีอยู่ของ TTY สองตัว)

ถ้าหนึ่งในการประมวลผลใช้รูทีนย่อย **setpri(10758, 59)** เพื่อกำหนดระดับความสำคัญที่ไม่เปลี่ยนแปลง ตัวอย่างของเอาต์พุต **ps –l** จะเป็นดังต่อไปนี้:

```
      F S UID   PID  PPID   C PRI NI ADDR    SZ    WCHAN    TTY  TIME CMD
 200903 S   0 10758 10500   0  59 -- 3438    40  4f91f98  pts/0  0:00 fixpri
```

การแก้ไขระดับความสำคัญด้วยคำสั่ง renice

คำสั่ง **renice** จะปรับเปลี่ยนค่า `nice` และระดับความสำคัญของการประมวลผลตั้งแต่หนึ่งกระบวนการขึ้นไปที่กำลังรันอยู่ การประมวลผลจะถูกระบุด้วย ID การประมวลผล ID กลุ่มการประมวลผล หรือชื่อของผู้ใช้ที่เป็นเจ้าของการประมวลผล อย่างใดอย่างหนึ่ง

คำสั่ง **renice** ไม่สามารถใช้สำหรับการประมวลผลที่มี ระดับความสำคัญที่ไม่เปลี่ยนแปลง ผู้ใช้ที่ไม่ใช่รากสามารถระบุค่าที่ต้องการเพิ่มได้ แต่ไม่สามารถลบออกจากค่า `nice` ของการประมวลผลที่รันอยู่ตั้งแต่หนึ่งกระบวนการขึ้นไป การแก้ไข จะถูกทำกับค่า `nice` ของการประมวลผล ระดับความสำคัญของการประมวลผลเหล่านี้จะยังคงไม่ใช่ค่าที่ไม่เปลี่ยนแปลง เฉพาะผู้ใช้รากเท่านั้นที่สามารถใช้คำสั่ง **renice** เพื่อปรับเปลี่ยนค่าของระดับความสำคัญภายในช่วง –20 ถึง 20 หรือลบออกจากค่า `nice` ของการประมวลผลที่รันอยู่ตั้งแต่หนึ่งกระบวนการขึ้นไป

หากต้องการดำเนินการกับตัวอย่างต่อไป ให้ใช้คำสั่ง **renice** เพื่อปรับเปลี่ยนค่า `nice` ของการประมวลผล **vmstat** ที่คุณเริ่มต้นทำงานกับ **nice**

```
# renice -n -5 7568
# ps -lu user1
      F S UID  PID PPID   C PRI NI ADDR    SZ    WCHAN    TTY  TIME CMD
 241801 S 200 7032 7286   0  60 20 1b4c   108           pts/2  0:00 ksh
 200801 S 200 7568 7032   0  60 20 2310    92  5910a58  pts/2  0:00 vmstat
 241801 S 200 8544 6494   0  60 20 154b   108           pts/0  0:00 ksh
```

ณ ตอนนี้ การประมวลผลกำลังรันอยู่ที่ระดับความสำคัญที่คุ้นเคย ซึ่งเท่ากับการประมวลผลส่วนหน้า อื่นๆ หากต้องการเลิกทำผลกระทบที่เกิดขึ้นจากการทำเช่นนี้ คุณสามารถออกคำสั่งต่อไปนี้ได้:

```
# renice -n 5 7568
# ps -lu user1
      F S UID  PID PPID   C PRI NI ADDR    SZ    WCHAN    TTY  TIME CMD
 241801 S 200 7032 7286   0  60 20 1b4c   108           pts/2  0:00 ksh
 200801 S 200 7568 7032   1  70 25 2310    92  5910a58  pts/2  0:00 vmstat
 241801 S 200 8544 6494   0  60 20 154b   108           pts/0  0:00 ksh
```

ในตัวอย่างเหล่านี้ คำสั่ง **renice** จะถูกรันโดย ผู้ใช้ราก เมื่อรันโดย ID ผู้ใช้ธรรมดา จะมีข้อจำกัดหลักอยู่ด้วยกันสองข้อในการใช้คำสั่ง **renice** :

- เฉพาะการประมวลผลที่เป็นเจ้าของโดย ID ผู้ใช้นั้นเท่านั้นที่สามารถระบุได้

- ค่า nice ของการประมวลผลไม่สามารถลดลงได้ ไม่ว่าจะส่งคืนการประมวลผลกลับไปยังระดับความสำคัญที่เป็นค่าดีฟอลต์หลังจากทำระดับความสำคัญ ให้ต่ำกว่าคำสั่ง **renice**

การอธิบายไวยากรณ์คำสั่ง nice และ renice

คำสั่ง **nice** และ **renice** มีวิธี ระบุจำนวนที่จะเพิ่มในค่า nice มาตรฐานจำนวน 20 แตกต่างกัน

คำสั่ง Nice	คำสั่ง Renice	ให้ผลลัพธ์ค่า nice	ค่าลำดับความสำคัญที่ดีที่สุด
nice -n 5	**renice -n 5**	25	70
nice -n +5	**renice -n +5**	25	70
nice -n -5	**renice -n -5**	15	55

การคำนวณค่าระดับความสำคัญของเธรด

ส่วนนี้อธิบายการปรับโดยใช้การคำนวณระดับความสำคัญและคำสั่ง **schedo**

คำสั่ง "คำสั่ง schedo" ในหน้า 136 ช่วยให้คุณสามารถ เปลี่ยนพารามิเตอร์ตัวจัดตารางเวลา CPU บางอย่างที่ใช้ในการคำนวณค่า ระดับความสำคัญของเธรดแต่ละรายการ โปรดดู "ระดับความสำคัญของกระบวนการและเธรด" ในหน้า 45 สำหรับข้อมูลพื้นหลังเกี่ยวกับระดับความสำคัญ

เมื่อต้องการกำหนดว่าโปรแกรม **schedo** มีการติดตั้งและมีอยู่ หรือไม่ ให้รันคำสั่งต่อไปนี้:

```
# lslpp -lI bos.perf.tune
```

การคำนวณระดับความสำคัญ:

เคอร์นัลเก็บรักษาค่าระดับความสำคัญ (บางครั้งเรียกว่าระดับความสำคัญ การจัดตารางเวลา) สำหรับแต่ละเธรด ค่าระดับความสำคัญเป็นจำนวนเต็มบวก และแตกต่างกันไปอย่างมาก ตามความสำคัญของเธรดที่เชื่อมโยง นั่นคือ ค่าระดับความสำคัญที่น้อยกว่า บ่งชี้ถึงเธรดที่มีความสำคัญมากกว่า เมื่อตัวจัดตารางเวลาค้นหาเธรดที่จะจัดส่ง ตัวจัดตารางเวลาจะเลือกเธรดที่จัดส่งได้ซึ่งมีค่าระดับความสำคัญ ต่ำที่สุด

สูตรสำหรับการคำนวณค่าระดับความสำคัญคือ:

```
ค่าระดับความสำคัญ = ระดับความสำคัญพื้นฐาน + nice penalty + (CPU penalty ตามข้อมูลการใช้ CPU ล่าสุด)
```

ค่าการใช้ CPU ล่าสุดของเธรดที่กำหนดจะเพิ่มขึ้น 1 ในแต่ละครั้ง ที่เธรดอยู่ในการควบคุมของ CPU เมื่อการขัดจังหวะไทม์เมอร์เกิดขึ้น (ทุก 10 มิลลิวินาที) ค่าการใช้ CPU ล่าสุดแสดงขึ้นเป็นคอลัมน์ *C* ในเอาต์พุต คำสั่ง **ps** ค่าสูงสุดของการใช้ CPU ล่าสุดคือ 120

ขั้นตอนวิธีดีฟอลต์คำนวณ CPU penalty โดยการหารการใช้ CPU ล่าสุดด้วย 2 ดังนั้นอัตราส่วน CPU-penalty-ต่อ-การใช้-CPU-ล่าสุดจึงเป็น 0.5 อัตราส่วนนี้ มีการควบคุมโดยค่าที่เรียกว่า *R* (ค่าดีฟอลต์ คือ 16) สูตรเป็นดังนี้:

```
CPU_penalty = C * R/32
```

ทุกหนึ่งวินาที ขั้นตอนวิธีดีฟอลต์จะหารค่าการใช้ CPU ล่าสุดของทุกเธรด ด้วย 2 ดังนั้นปัจจัย recent-CPU-usage-decay จึงเป็น 0.5 ปัจจัยนี้มีการควบคุมโดยค่าที่เรียกว่า *D* (ค่า ดีฟอลต์คือ 16) สูตรเป็นดังนี้:

```
C = C * D/32
```

ขั้นตอนวิธีสำหรับการคำนวณค่าระดับความสำคัญใช้ค่า **nice** ของ กระบวนการเพื่อกำหนดระดับความสำคัญของเธรดใน กระบวนการ เมื่อ หน่วยของเวลา CPU เพิ่มขึ้น ระดับความสำคัญจะลดลงด้วยผลกระทบ nice การใช้ schedo -r -d ช่วยให้การควบคุมเพิ่มเติมในการคำนวณระดับ ความสำคัญโดยการตั้งค่าใหม่สำหรับ R และ D โปรดดู "คำสั่ง schedo"สำหรับข้อมูลเพิ่มเติม

เริ่มต้นด้วยสมการต่อไปนี้:

```
p_nice = ระดับความสำคัญพื้นฐาน + ค่า nice
```

ตอนนี้ใช้สูตรต่อไปนี้:

```
If p_nice > 60,
   then x_nice = (p_nice * 2) - 60,
   else x_nice = p_nice.
```

ถ้าค่า nice มากกว่า 20 แสดงว่ามีผลกระทบต่อค่าระดับความสำคัญ เป็นสองเท่าเมื่อเทียบกับถ้าค่า nice น้อยกว่าหรือเท่ากับ 20 การคำนวณระดับความสำคัญใหม่ (การละเว้นการตัดเลขจำนวนเต็ม) เป็นดังนี้:

```
ค่าระดับความสำคัญ = x_nice + [(x_nice + 4)/64 * C*(R/32)]
```

คำสั่ง schedo:

สามารถปรับได้โดยใช้อ็อพชันสองรายการของคำสั่ง **schedo** คือ: **sched_R** และ **sched_D**

แต่ละอ็อพชันระบุพารามิเตอร์ที่เป็นเลขจำนวนเต็มตั้งแต่ 0 ถึง 32 วิธีการใช้พารามิเตอร์คือคูณด้วยค่าของพารามิเตอร์ แล้วหารด้วย 32 ค่าดีฟอลต์ของ *R* และ *D* คือ 16 ซึ่งให้พฤติกรรม เหมือนกันกับขั้นตอนวิธีดั้งเดิม [(D=R=16)/32=0.5] ช่วงค่าใหม่ ทำให้สามารถตรวจสอบพฤติกรรมได้มากขึ้น ตัวอย่างเช่น:

```
# schedo -o sched_R=0
```

[(R=0)/32=0, (D=16)/32=0.5] อาจหมายความว่า CPU penalty เป็น 0 เสมอ ทำให้ระดับความสำคัญทำหน้าที่เป็นค่า nice อย่างเดียวเท่านั้น ไม่มีกระบวนการพื้นหลังใด จะได้รับเวลา CPU ยกเว้นว่าไม่มีกระบวนการพื้นหน้าที่จัดส่งได้ อยู่เลย ค่าระดับความสำคัญของเธรดอาจจะเป็นค่าคงที่ แม้ว่าทางด้านเทคนิคแล้ว จะไม่ใช่เธรดที่มีระดับความสำคัญคงที่ก็ตาม

```
# schedo -o sched_R=5
```

[(R=5)/32=0.15625, (D=16)/32=0.5] อาจหมายความว่ากระบวนการ พื้นหน้าจะไม่มีทางแข่งขันกับกระบวนการพื้นหลังที่เริ่มต้นด้วยคำสั่ง **nice -n 10** ขีดจำกัดของส่วนเวลา CPU ที่สะสม 120 อาจหมายความว่า CPU penalty สูงสุดสำหรับกระบวนการพื้นหน้าจะเป็น 18

```
# schedo -o sched_R=6 -o sched_D=16
```

[(R=6)/32=0.1875, (D=16)/32=0.5] อาจหมายความว่า ถ้ากระบวนการ พื้นหลังมีการเริ่มต้นด้วยคำสั่ง **nice -n 10** จะมีอย่างน้อย หนึ่งวินาทีก่อนกระบวนการพื้นหลังจะเริ่มต้นได้รับเวลา CPU ใดๆ อย่างไรก็ตาม กระบวนการพื้นหน้าจะยังคงสามารถแยกความแตกต่างได้ โดยดูจากข้อมูลพื้นฐานของการใช้ CPU กระบวนการพื้นหน้าที่ใช้เวลารันนานที่ควรจะ รันอยู่ในพื้นหลังจะสะสมการใช้ CPU ให้เพียงพอที่จะไม่ต้อง ถูกรบกวนจากกระบวนการพื้นหน้าจริงในท้ายที่สุด

```
# schedo -o sched_R=32 -o sched_D=32
```

[(R=32)/32=1, (D=32)/32=1] อาจหมายความว่า เธรดที่ใช้เวลารันนานอาจ มีค่า C ถึง 120 และคงอยู่เช่นนั้น แข่งขันกันด้วยค่า nice เธรดใหม่จะมีระดับความสำคัญโดยไม่คำนึงถึงค่า nice ของเธรดนั้น จนกว่าเธรดสะสมส่วนเวลาจนมากเพียงพอที่จะนำเธรดไปภายในช่วงค่าระดับความ สำคัญของเธรดที่มีอยู่

คำแนะนำสำหรับ R และ D มีดังนี้:

- ค่า R น้อยลงทำให้ช่วงระดับความสำคัญแคบลงและทำให้ค่า nice มีผลกระทบต่อระดับความสำคัญมากขึ้น
- ค่า R มากขึ้นทำให้ช่วงระดับความสำคัญกว้างขึ้นและทำให้ค่า nice มีผลกระทบต่อระดับความสำคัญน้อยลง
- ค่า D น้อยลงทำให้การใช้ CPU มีอัตราเร็วขึ้น และส่งผลให้เธรดที่เน้น CPU จะถูกจัดตารางเวลาเร็วขึ้น
- ค่า D มากขึ้นทำให้การใช้ CPU มีอัตราช้าขึ้น และทำลายเธรดที่เน้น CPU มากขึ้น (ดังนั้นจึงเหมาะสำหรับเธรดชนิดโต้ตอบ)

ตัวอย่างการคำนวณระดับความสำคัญ:

ตัวอย่างแสดง R=4 และ D=31 และไม่มีเธรดที่สามารถรันได้

```
current_effective_priority
                  | base process priority
                  |    | nice value
                  |    |    | count (time slices consumed)
                  |    |    |    | (schedo -o sched_R)
                  |    |    |    |     |
  time 0          p = 40 + 20 + (0   * 4/32) =   60
  time 10 ms      p = 40 + 20 + (1   * 4/32) =   60
  time 20 ms      p = 40 + 20 + (2   * 4/32) =   60
  time 30 ms      p = 40 + 20 + (3   * 4/32) =   60
  time 40 ms      p = 40 + 20 + (4   * 4/32) =   60
  time 50 ms      p = 40 + 20 + (5   * 4/32) =   60
  time 60 ms      p = 40 + 20 + (6   * 4/32) =   60
  time 70 ms      p = 40 + 20 + (7   * 4/32) =   60
  time 80 ms      p = 40 + 20 + (8   * 4/32) =   61
  time 90 ms      p = 40 + 20 + (9   * 4/32) =   61
  time 100ms      p = 40 + 20 + (10  * 4/32) =   61
        .
  (skipping forward to 1000msec or 1 second)
        .
  time 1000ms     p = 40 + 20 + (100 * 4/32) =   72
  time 1000ms     swapper recalculates the accumulated CPU usage counts of
                  all processes. For the above process:
                  new_CPU_usage = 100 * 31/32 = 96  (if d=31)
                  after decaying by the swapper: p = 40 + 20 + ( 96 * 4/32) = 72
                  (if d=16, then p = 40 + 20 + (100/2 * 4/32) = 66)
  time 1010ms     p = 40 + 20 + ( 97 * 4/32) =   72
  time 1020ms     p = 40 + 20 + ( 98 * 4/32) =   72
  time 1030ms     p = 40 + 20 + ( 99 * 4/32) =   72
                 ..
  time 1230ms     p = 40 + 20 + (119 * 4/32) =   74
  time 1240ms     p = 40 + 20 + (120 * 4/32) =   75     count <= 120
  time 1250ms     p = 40 + 20 + (120 * 4/32) =   75
  time 1260ms     p = 40 + 20 + (120 * 4/32) =   75
                 ..
```

```
time 2000ms     p = 40 + 20 + (120 * 4/32) =   75
time 2000ms     swapper recalculates the counts of all processes.
                For above process 120 * 31/32 = 116
time 2010ms     p = 40 + 20 + (117 * 4/32) =   74
```

การแก้ไขการแบ่งเวลาของตัวกำหนดตารางเวลาด้วยคำสั่ง schedo

ความยาวของการแบ่งเวลาของตัวกำหนดตารางเวลาสามารถแก้ไขได้ด้วยคำสั่ง **schedo** หากต้องการเปลี่ยนการแบ่งเวลา ให้ใช้อ็อพชัน **schedo -o timeslice=***value*

ค่าของ **-t** คือจำนวนของการทำเครื่องหมายสำหรับการแบ่งเวลา และเฉพาะเธรด SCHED_RR เท่านั้นที่จะใช้ค่าการแบ่งเวลาที่ไม่ใช่ค่าดีฟอลต์ (โปรดดู "นโยบายการจัดตารางเวลาสำหรับเธรด" ในหน้า 47 สำหรับคำอธิายของ เธรดที่มีระดับความสำคัญที่ไม่เปลี่ยนแปลง)

การเปลี่ยนการแบ่งเวลาจะมีผลบังคับใช้โดยทันที และไม่ต้องการรีบูต

เธรดที่รันด้วยนโยบายการกำหนดตารางเวลา SCHED_OTHER หรือ SCHED_RR สามารถใช้ CPU สำหรับการแบ่งเวลาแบบเต็ม (การแบ่งเวลาตามค่าดีฟอลต์คือ เครื่องหมาย 1 ขีด ของนาฬิกา) และขีดของนาฬิกาหนึ่งขีดคือ 10 มิลลิวินาที

ในบางสถานการณ์ มี context switching มากเกินไปที่กำลังเกิดขึ้น และการใช้ของการจัดส่งเธรดอาจใช้เวลานานกว่าการอนุญาตให้เธรดเหล่านี้ รันสำหรับการแบ่งเวลาให้ยาวนานขึ้น ในกรณีเหล่านี้ การเพิ่มการแบ่งเวลาอาจมีผลกระทบในทางบวกสำหรับผลการทำงานของเธรดที่มีระดับความสำคัญที่ไม่เปลี่ยนแปลง ให้ใช้คำสั่ง **vmstat** และ **sar** เพื่อกำหนดจำนวนของ context switch ต่อวินาที

ในสภาวะแวดล้อมที่มีความยาวของการแบ่งเวลาที่เพิ่มสูงขึ้น บางแอ็พพลิคเชันอาจไม่ต้องการหรือไม่ควรมีการแบ่งเวลาแบบเต็ม แอ็พพลิเคชันเหล่านี้ สามารถยกเลิกตัวประมวลผลอย่างชัดเจนด้วยการเรียกของระบบ **yield( )** (ซึ่งสามารถโปรแกรมได้ในสภาวะแวดล้อมที่ไม่มีการเปลี่ยนแปลง) หลังจากที่เรียก **yield( )** แล้ว การเรียกเธรดจะถูกย้ายไปยังส่วนท้ายของคิวการจัดส่ง สำหรับระดับความสำคัญ

การจัดการ ID ผู้ใช้ microprocessor-efficient ด้วยคำสั่ง mkpasswd

เพื่อปรับปรุงเวลาตอบกลับล็อกอินและสงวนเวลา microprocessor ในระบบที่มีผู้ใช้จำนวนมาก ระบบปฏิบัติการสามารถใช้เวอร์ชันที่จัดทำดัชนี ของไฟล์ /etc/passwd เพื่อค้นหา IDs ผู้ใช้ เมื่อใช้ฟังก์ชัน นี้ ไฟล์ /etc/passwd ยังคงมีอยู่ แต่ไม่ได้ใช้ในการประมวลผลปกติ

เวอร์ชันที่จัดทำดัชนีของไฟล์มีการสร้างขึ้นโดยคำสั่ง **mkpasswd** ถ้าเวอร์ชันที่จัดทำดัชนีไม่ใช่กระบวนการล็อกอินปัจจุบัน ให้แปลงเป็นการค้นหา ตามลำดับ microprocessor-intensive แบบช้าผ่านทาง /etc/passwd

คำสั่งที่สร้างไฟล์รหัสผ่านที่จัดทำดัชนีคือ **mkpasswd -f** คำสั่งนี้สร้างเวอร์ชันที่จัดทำดัชนีของ /etc/passwd, /etc/security/passwd, และ /etc/security/lastlog ไฟล์ที่สร้างขึ้นคือ /etc/passwd.nm.idx, /etc/passwd.id.idx, /etc/security/passwd.idx, และ /etc/security/lastlog.idx หมายเหตุว่าคำสั่งนี้จะปรับปรุงประสิทธิภาพของแอ็พพลิเคชันที่ยังต้องการรหัสผ่าน ที่เข้ารหัส (เช่น ล็อกอินและโปรแกรมอื่นใดที่ต้องทำการพิสูจน์ตัวตน รหัสผ่าน) ได้เป็นอย่างมาก

แอ็พพลิเคชันยังสามารถเปลี่ยนไปใช้รูทีนอื่น เช่น **_getpwent( )** แทน **getpwent( )**, **_getpwnam_shadow(name,0)** แทน **getpwnam(name)**, หรือ **_getpwuid_shadow(uid,0)** แทน **getpwuid(uid)** เพื่อทำการ แก้ไขชื่อ/ID ในกรณีที่ไม่ต้องการรหัสผ่านที่เข้ารหัส ซึ่งช่วยป้องกัน การค้นหา /etc/security/passwd

อย่าแก้ไขไฟล์รหัสผ่านด้วยมือเนื่องจากเวลาประทับของไฟล์ฐานข้อมูล (.idx) จะไม่ซิงค์กันและจะใช้วิธีการค้นหาดีฟอลต์ (เชิงเส้น) ถ้าใช้คำสั่ง **passwd, mkuser, chuser, rmuser** (หรือคำสั่ง SMIT ที่เทียบเท่า ซึ่งมีพาธด่วนของชื่อเดียวกัน) เพื่อจัดการ IDs ผู้ใช้ ไฟล์ที่จัดทำดัชนีจะมีการอัพเดตโดยอัตโนมัติ ถ้าไฟล์ /etc/passwd มีการเปลี่ยนแปลงโดยผู้แก้ไขหรือด้วย คำสั่ง **pwdadm** ไฟล์ดัชนีต้องถูกสร้างขึ้นใหม่

หมายเหตุ: คำสั่ง **mkpasswd** ไม่มีผลกระทบต่อฐานข้อมูลผู้ใช้ NIS, DCE, หรือ LDAP

ผลการทำงานของหน่วยความจำ

ส่วนนี้อธิบายถึงจำนวนหน่วยความจำที่ใช้ซึ่งสามารถวัดได้และแก้ไขได้

หน่วยความจำของระบบจะมีความจุที่เกือบเต็มอยู่ตลอดเวลา แม้ว่า โปรแกรมที่รันจะไม่ได้ใช้งานหน่วยความจำที่มีอยู่ทั้งหมด ระบบปฏิบัติการยังคงเก็บรักษาเพจข้อความของโปรแกรมที่รันก่อนหน้านี้ และไฟล์ที่โปรแกรมใช้ลงในหน่วยความจำ และไม่มีค่าใช้จ่ายที่เชื่อมโยงกับการเก็บรักษานี้ เนื่องจากหน่วยความจำที่ไม่ได้นำมาใช้ในหลายๆ กรณี โปรแกรมหรือไฟล์จะถูกใช้อีกครั้ง ซึ่งจะลดดิสก์ I/O

ผู้อ่านซึ่งไม่คุ้นเคยกับการจัดการกับหน่วยความจำเสมือน อาจต้องการค้นหา “ประสิทธิภาพผู้จัดการหน่วยความจำเสมือน” ในหน้า 51 ก่อนที่จะดำเนินการต่อ

การใช้หน่วยความจำ

ทูลด้านประสิทธิภาพการทำงานที่หลากหลายได้จัดเตรียมรายงานการใช้หน่วยความจำไว้

รายงานที่สำคัญส่วนใหญ่มาจากคำสั่ง **vmstat, ps** และ **svmon**

การพิจารณาการใช้หน่วยความจำด้วยคำสั่ง vmstat

คำสั่ง **vmstat** จะสรุปหน่วยความจำเสมือนที่ *แอ็คทีฟ* ทั้งหมดโดยใช้การประมวลผลทั้งหมดในระบบ และจำนวนของกรอบของเพจหน่วยความจำที่เกิดขึ้นจริง บนรายการว่าง

หน่วยความจำเสมือนที่แอ็คทีฟจะถูกนิยามเป็นจำนวนของหน่วยความจำเสมือนที่ทำงานกับเพจเซ็กเมนต์ ซึ่งได้ถูกสัมผัสแล้ว หมายเลขนี้อาจมีค่ามากกว่าจำนวนของกรอบเพจที่เกิดขึ้นจริงในเครื่อง เนื่องจากเพจหน่วยความจำเสมือนที่แอ็คทีฟบางตัวอาจถูกเขียนออกไปยัง พื้นที่การเพจ

ขณะที่พิจารณาว่า ระบบอาจมีหน่วยความจำแบบสั้น หรือการปรับหน่วยความจำบางส่วนตามจำนวนที่ต้องการ ให้รันคำสั่ง **vmstat** ผ่านการตั้งค่าช่วงเวลา และตรวจสอบคอลัมน์ *pi* และ *po* สำหรับรายงานผลลัพธ์ คอลัมน์เหล่านี้บ่งชี้ถึงจำนวนของพื้นที่การเพจ page-ins ต่อวินาที และจำนวนของพื้นที่การเพจ page-outs ต่อวินาที ถ้าค่านั้นเป็นค่าคงที่ที่ไม่ใช่ศูนย์ ก็อาจทำให้หน่วยความจำเกิดปัญหาคอขวด (bottleneck) ได้ การมีค่าที่ไม่ใช่ศูนย์เป็นครั้งคราว ไม่ใช่ปัญหาที่ต้องระวัง เนื่องจากการเพจคือ ตัวการสำคัญของหน่วยความจำเสมือน

```
# vmstat 2 10
kthr     memory             page              faults        cpu
----- ----------- ------------------------ ------------ -----------
 r  b   avm   fre  re  pi  po  fr   sr  cy  in   sy  cs us sy id wa
 1  3 113726   124   0  14   6 151  600   0 521 5533 816 23 13  7 57
 0  3 113643   346   0   2  14 208  690   0 585 2201 866 16  9  2 73
 0  3 113659   135   0   2   2 108  323   0 516 1563 797 25  7  2 66
 0  2 113661   122   0   3   2 120  375   0 527 1622 871 13  7  2 79
```

```
 0  3 113662   128   0  10   3 134  432   0 644 1434 948 22  7  4 67
 1  5 113858   238   0  35   1 146  422   0 599 5103 903 40 16  0 44
 0  3 113969   127   0   5  10 153  529   0 565 2006 823 19  8  3 70
 0  3 113983   125   0  33   5 153  424   0 559 2165 921 25  8  4 63
 0  3 113682   121   0  20   9 154  470   0 608 1569 1007 15  8  0 77
 0  4 113701   124   0   3  29 228  635   0 674 1730 1086 18  9  0 73
```

สำหรับเอาต์พุตตัวอย่างข้างต้น ให้สังเกต I/O ที่มีค่าสูงซึ่งรออยู่ในเอาต์พุต และจำนวนของเธรดบนคิวที่บล็อก กิจกรรม I/O อื่นๆ อาจทำให้ I/O รอ แต่ในกรณีนี้ I/O ที่รอมีสาเหตุมาจาก การเพจเข้าและออกจากพื้นที่การเพจ

หากต้องการดูว่า ระบบมีปัญหาในเรื่องผลการทำงานกับ VMM ให้ตรวจสอบคอลัมน์ที่อยู่ใต้ *หน่วยความจำ* และ *เพจ*:

- **หน่วยความจำ**
  แสดงข้อมูลเกี่ยวกับหน่วยความจำที่ใช้จริงและหน่วยความจำเสมือน
  - **avm**
    หน่วยความเสมือนที่แอ็คทีฟ (Active Virtual Memory) คอลัมน์ `avm` จะแสดงถึงจำนวนของเพจหน่วยความจำเสมือนที่แอ็คทีฟซึ่งแสดงอยู่ในเวลาที่ตัวอย่าง **vmstat** ถูกเก็บรวบรวม นโยบายพื้นที่เพจที่เลื่อนออกไปคือนโยบายดีฟอลต์ภายใต้นโยบายนี้ ค่าสำหรับ `avm` อาจสูงกว่าจำนวนของเพจพื้นที่การเพจ ที่ใช้ ข้อมูลสถิติ `avm` จะไม่สอดแทรกเพจไฟล์
  - **fre**
    คอลัมน์ `fre` จะแสดงจำนวนเฉลี่ย ของเพจหน่วยความจำที่เป็นอิสระ เพจจะมีพื้นที่ขนาด 4 KB ของหน่วยความจำที่เกิดขึ้นจริง ระบบจะรักษาบัฟเฟอร์ของเพจหน่วยความจำไว้ ซึ่งเรียกว่า รายการอิสระ โดยจะมีความสามารถในการเข้าถึงได้ เมื่อ VMM ต้องการพื้นที่ จำนวนต่ำสุดของเพจที่ VMM เก็บไว้บนรายการอิสระ จะถูกพิจารณาโดยพารามิเตอร์ *minfree* ของคำสั่ง **vmo** สำหรับรายละเอียดเพิ่มเติม โปรดดู “การปรับการเปลี่ยนหน้า VMM” ในหน้า 166
    เมื่อแอ็พพลิเคชันถูกยกเลิก เพจการทำงานทั้งหมดจะส่งกลับคืนรายการอิสระโดยทันที อย่างไรก็ตาม เพจหรือไฟล์ที่ยังคงอยู่จะยังคงอยู่ใน RAM และไม่ได้เพิ่มกลับไปยังรายการอิสระ จนกว่าจะถูกนำไปใช้โดย VMM สำหรับโปรแกรมอื่น เพจที่ยังคงอยู่ ยังคงเป็นอิสระหากไฟล์ที่สอดคล้องกันถูกลบทิ้ง
    ด้วยเหตุผลนี้ ค่า *fre* อาจไม่ได้บ่งชี้ถึงหน่วยความจำที่ใช้จริงทั้งหมด ซึ่งสามารถทำให้พร้อมใช้งานได้กับการประมวลผลอื่นๆ ถ้าจำเป็นต้องมีกรอบของเพจ เพจที่ยังคงอยู่ซึ่งเกี่ยวข้องกับแอ็พพลิเคชันที่ถูกยกเลิกจะอยู่ท่ามกลางการส่งต่อไปยังโปรแกรมอื่น ในส่วนแรก
    ถ้าค่า *fre* เป็นส่วนหนึ่งของ *maxfree* ที่อยู่ด้านบน จึงมีความเป็นไปได้น้อยที่ระบบจะ *แกว่งไปแกว่งมา* การแกว่งไปแกว่งมาหมายความว่า ระบบยังคงเพจเข้าและเพจออกอยู่ อย่างไรก็ตาม ถ้าระบบกำลังพบกับการแกว่งไปแกว่งมา คุณสามารถเชื่อมั่นได้ว่า ค่า *fre* จะมีขนาดเล็ก
- **เพจ**
  ข้อมูลเกี่ยวกับข้อบกพร่องของเพจและกิจกรรมการเพจ ซึ่งข้อมูลเหล่านี้จะเป็นค่าเฉลี่ย ตามช่วงเวลา และกำหนดไว้ในหน่วยต่อวินาที
  - **re**

    หมายเหตุ: คอลัมน์นี้ยังคงไม่ได้รับการสนับสนุน
  - **pi**
    คอลัมน์ `pi` จะแสดงรายละเอียดของจำนวนเพจที่เพจเข้า จากพื้นที่การเพจ พื้นที่การเพจคือส่วนหนึ่งของหน่วยความจำเสมือน ที่อยู่บนดิสก์ ซึ่งจะใช้เป็นโอเวอร์โฟลว์ เมื่อหน่วยความจำถูก commit มากเกินไป พื้นที่การเพจประกอบ

ด้วยโลจิคัลวอลุ่ม ที่ใช้เฉพาะกับหน่วยเก็บของเพจชุดการทำงาน ที่ได้มาจากหน่วยความจำที่ใช้จริง เมื่อเพจที่ถูกนำมาใช้อ้างอิงโดยการประมวลผล ข้อบกพร่องของเพจจะเกิดขึ้น และเพจนั้นต้องถูกอ่านในหน่วยความจำ จากพื้นที่การเพจ เนื่องจากมีคอนฟิกูเรชันของฮาร์ดแวร์ ซอฟต์แวร์ และแอ็พพลิเคชันที่หลากหลาย จึงไม่มีจำนวนที่แน่นอนในการค้นหา ฟิลด์นี้เป็นฟิลด์ที่สำคัญ เนื่องจากเป็นตัวบ่งชี้หลักของกิจกรรมของพื้นที่การเพจ ถ้าการเพจขาเข้าเกิดขึ้น จำเป็นต้องมีการเพจขาออกก่อนหน้านี้สำหรับเพจนั้น และยังเหมือนกับในสภาวะแวดล้อมแบบจำกัดหน่วยความจำ ซึ่งแต่ละเพจขาเข้าจะบังคับเพจอื่นๆ ให้ถูกนำมาใช้ และรวมถึงเพจขาออกด้วย

- **po**

  คอลัมน์ *po* จะแสดงจำนวน (อัตรา) ของเพจ ที่เพจออกไปยังพื้นที่การเพจ เมื่อใดก็ตามที่เพจของหน่วยเก็บใช้งานถูกนำมาใช้ เพจนั้นจะถูกเขียนลงในพื้นที่การเพจ ถ้าเพจนั้นยังไม่ได้อยู่ในพื้นที่การเพจ หรือถ้าเพจนั้นถูกแก้ไข ถ้าไม่มีการอ้างถึงอีกครั้ง เพจนั้นจะยังคงอยู่บนอุปกรณ์การเพจ จนกว่าการประมวลผลจะยกเลิกหรือไม่ยอมรับพื้นที่ การอ้างอิงกับแอดเดรสในลำดับถัดมา จะมีอยู่ภายในเพจที่อยู่นอกความบกพร่องซึ่งส่งผลทำให้เพจเกิดความบกพร่อง และเพจนั้น จะถูกเพจเข้าโดยระบบ เมื่อการประมวลผลยกเลิกแบบปกติ พื้นที่การเพจใดๆ ที่จัดสรรให้กับการประมวลผลนั้นจะถูกล้างข้อมูล ถ้าระบบกำลังอ่านจำนวนของเพจที่ยังคงอยู่ ซึ่งเป็นข้อมูลที่สำคัญ คุณอาจมองเห็นการเพิ่มขึ้นใน *po* โดยไม่มีการเพิ่มขึ้นใน *pi* ที่สอดคล้องกัน เหตุการณ์นี้ไม่ได้บ่งชี้ถึงการแกว่งไปแกว่งมา แต่อาจเป็นการรับประกันการตรวจสอบในรูปแบบการเข้าถึงข้อมูลของแอ็พพลิเคชัน

- **fr**

  จำนวนของเพจที่ถูกล้างข้อมูลต่อวินาทีโดยอัลกอริธึมการแทนที่เพจ ในระหว่างช่วงเวลา เนื่องจากรูทีนการแทนที่เพจ VMM จะสแกนตาราง กรอบของเพจ หรือ PFT ซึ่งรูทีนจะใช้เงื่อนไขเพื่อเลือกเพจที่ต้องการนำมาใช้ เพื่อเติมเต็มรายการอิสระของกรอบหน่วยความจำที่พร้อมใช้งาน เงื่อนไขจะสอดแทรกชนิดของเพจ การทำงาน (การคำนวณ) และเพจไฟล์ (การคงอยู่) เนื่องจากเพจได้ถูกล้างข้อมูลแล้ว จึงไม่ได้หมายความถึงว่า I/O ใดๆ จะเข้าแทนที่ ตัวอย่างเช่น ถ้าเพจของหน่วยเก็บ (ไฟล์) ที่ยังคงอยู่ ไม่ได้ถูกแก้ไข เพจนั้นจะไม่ถูกเขียนกลับไปยังดิสก์ ถ้า I/O ไม่มีความจำเป็น รีซอร์สของระบบจำนวนน้อยที่สุดจะมีความต้องการล้างข้อมูลเพจนั้น

- **sr**

  จำนวนของเพจที่ถูกตรวจสอบต่อวินาทีโดยอัลกอริธึมการแทนที่เพจ ในระหว่างช่วงเวลา อัลกอริธึมการแทนที่เพจอาจต้องสแกนกรอบของเพจจำนวนมาก ก่อนที่จะนำกรอบของเพจเหล่านั้นมาใช้เพื่อให้มีเพียงพอกับ threshold การแทนที่เพจ ค่า **sr** ที่สูงกว่าเมื่อเปรียบเทียบกับค่า **fr** คือความยากของอัลกอริธึมการแทนที่เพจเพื่อค้นหาเพจที่เหมาะสมสำหรับนำมาใช้

- **cy**

  จำนวนวงรอบต่อวินาทีของอัลกอริธึมนาฬิกา VMM จะใช้เทคนิคที่รู้จักกันในนามของอัลกอริธึมนาฬิกาเพื่อเลือกเพจที่ต้องการแทนที่ เทคนิคนี้จะใช้ประโยชน์ของบิตที่อ้างถึงสำหรับแต่ละเพจเป็นสิ่งบ่งชี้ถึง เพจที่มีการใช้ล่าสุด (อ้างอิงถึง) เมื่อรูทีนตัวเรียกใช้เพจ ถูกเรียก รูทีนจะวงรอบผ่าน PFT ซึ่งจะตรวจสอบบิตที่อ้างอิง ของเพจ

  คอลัมน์ *cy* จะแสดงจำนวนครั้งต่อวินาทีที่โค้ดการแทนที่เพจ ได้ถูกสแกน PFT เนื่องจากรายการอิสระสามารถเติมเต็มได้โดยไม่ต้องสแกน PFT ให้เสร็จสิ้น และเนื่องจากฟิลด์ **vmstat** ทั้งหมดจะถูกรายงานในรูปของตัวเลข ฟิลด์นี้จะมีค่าเป็นศูนย์ตามปกติ

วิธีหนึ่งในการพิจารณาจำนวนที่เหมาะสมของ RAM สำหรับระบบคือ การมองหาค่าที่ใหญ่ที่สุดสำหรับ *avm* ตามที่ได้แสดงด้วยคำสั่ง **vmstat** การคูณค่านั้นด้วย 4 K จะได้จำนวนไบต์ จากนั้นให้เปรียบเทียบ จำนวนไบต์ของ RAM บนระบบ ตามหลักการแล้ว *avm* ควรมีค่าน้อยกว่าจำนวนทั้งหมดของ RAM ถ้าไม่ใช่ จำนวนบางส่วนของการเพจหน่วยความจำเสมือนจะเกิดขึ้น จำนวนการเพจ ที่เกิดขึ้นจะขึ้นอยู่กับความแตกต่างระหว่างค่าสองค่า โปรดจำไว้ว่า แนวคิดของหน่วยความจำเสมือนคือ การให้ความสามารถในการกำหนดแอดเดรส ด้วยหน่วยความจำที่มากกว่าที่เรามี (หน่วยความจำบางส่วนอยู่ใน RAM และส่วนที่

เหลืออยู่ในพื้นที่การเพจ) แต่ถ้ามีหน่วยความจำเสมือนที่มากกว่าหน่วยความจำที่ใช้จริง สิ่งนี้อาจเป็นสาเหตุทำให้มีการเพจเกินปกติซึ่งส่งผลทำให้เกิดการหน่วงเวลา ถ้า *avm* มีขนาดต่ำกว่า RAM การเพจของพื้นที่การเพจอาจเป็นสาเหตุทำให้ RAM ถูกเติมด้วยเพจไฟล์จนเต็ม ในกรณีนี้ การปรับค่า *minperm*,*maxperm* และ *maxclient* สามารถลดจำนวนของการเพจของ พื้นที่การเพจลงได้ โปรดอ้างถึง "การปรับการเปลี่ยนหน้า VMM" ในหน้า 166 สำหรับข้อมูลเพิ่มเติม

คำสั่ง vmstat -I:

คำสั่ง **vmstat -I** แสดงข้อมูลเพิ่มเติม เช่น file pages in ต่อวินาที, file pages out ต่อวินาที ซึ่งหมายถึง VMM page-ins และ page-outs ใดๆ ที่ไม่ใช่ page-ins พื้นที่ว่างการเพจ หรือ page-outs พื้นที่ว่างการเพจ

คอลัมน์ `re` และ `cy` ไม่ได้รายงานด้วยแฟล็กนี้

คำสั่ง vmstat -s:

อ็อพชันสรุป, **-s**, ส่งรายงานสรุปไปยังเอาต์พุตมาตรฐาน โดยเริ่มต้นจากการเริ่มต้นระบบที่ระบุในจำนวนนับแทน ช่วงเวลา

วิธีที่แนะนำในการใช้สถิติเหล่านี้คือรันคำสั่งนี้ก่อนเวิร์กโหลด บันทึกเอาต์พุต แล้วรันคำสั่งอีกครั้งหลังจากเวิร์กโหลด และบันทึกเอาต์พุต ขั้นตอนถัดไปคือการกำหนดความแตกต่างระหว่างชุดเอาต์พุต สองชุด **awk** script ที่เรียกว่า **vmstatit** ซึ่งทำกระบวนงานนี้โดยอัตโนมัติมีการจัดเตรียมไว้ใน "ปัญหาเกี่ยวกับดิสก์หรือหน่วยความจำ" ในหน้า 40

```
# vmstat -s
  3231543 total address trans. faults
    63623 page ins
   383540 page outs
      149 paging space page ins
      832 paging space page outs
        0 total reclaims
   807729 zero filled pages faults
     4450 executable filled pages faults
   429258 pages examined by clock
        8 revolutions of the clock hand
   175846 pages freed by the clock
    18975 backtracks
        0 lock misses
       40 free frame waits
        0 extend XPT waits
    16984 pending I/O waits
   186443 start I/Os
   186443 iodones
141695229 cpu context switches
317690215 device interrupts
        0 software interrupts
        0 traps
 55102397 syscalls
```

จำนวน page-in และ page-out ในสรุปแสดงถึงกิจกรรมหน่วยความจำเสมือน ที่จะ page in หรือ out pages จากพื้นที่ว่างหน้าและพื้นที่ว่างไฟล์ ข้อมูล paging space ins และ outs แสดงถึงพื้นที่ว่างหน้าเท่านั้น

การกำหนดปัญหาการใช้หน่วยความจำด้วยคำสั่ง ps

คำสั่ง **ps** ยังสามารถใช้เพื่อมอนิเตอร์การใช้หน่วยความจำ ของการประมวลผล

คำสั่ง **ps v** *PID* จัดเตรียมรายงานที่ครอบคลุม ข้อมูลสถิติที่เกี่ยวข้องกับหน่วยความจำสำหรับการประมวลผล เช่น:

- ข้อบกพร่องของเพจ
- ขนาดของเซ็กเมนต์ใช้งานที่สัมผัส
- ขนาดของเซ็กเมนต์ใช้งานและเซ็กเมนต์โค้ดในหน่วยความจำ
- ขนาดของเซ็กเมนต์ข้อความ
- ขนาดของชุดที่ฝังตัว
- เปอร์เซ็นต์ของหน่วยความจำที่ใช้จริงซึ่งใช้โดยการประมวลผลนี้

ต่อไปนี้คือตัวอย่าง:

```
# ps v
   PID    TTY STAT  TIME PGIN  SIZE   RSS   LIM  TSIZ   TRS %CPU %MEM COMMAND
 36626  pts/3 A     0:00    0   316   408 32768    51    60  0.0  0.0 ps v
```

คอลัมน์ที่สำคัญบนรายงานที่แสดงผลลัพธ์ **ps** จะอธิบายไว้ดังต่อไปนี้:

PGIN จำนวนของเพจขาเข้าที่เป็นสาเหตุทำให้เกิดข้อบกพร่องของเพจ เนื่องจาก I/O ทั้งหมดจะถูกจัดกลุ่มเป็นข้อบกพร่องของเพจ นี่คือการวัดค่าวอลุ่ม I/O พื้นฐาน

SIZE ขนาดเสมือน (ในพื้นที่การเพจ) ในหน่วยกิโลไบต์ของส่วนของข้อมูลของ การประมวลผล (แสดงเป็น SZ โดยแฟล็กอื่นๆ) จำนวนนี้มีค่าเท่ากับจำนวนของเพจเซ็กเมนต์ใช้งาน ของการประมวลผลที่สัมผัสคูณด้วย 4 ถ้าเพจเซ็กเมนต์ใช้งานบางเพจ ถูกเพจออก จำนวนนี้จะมีค่ามากกว่าจำนวนของหน่วยความจำที่ใช้จริง ที่ถูกใช้ SIZE ประกอบด้วยเพจในเซ็กเมนต์ส่วนตัว และเซ็กเมนต์ข้อมูลไบบารีที่แบ่งใช้ของการประมวลผล

RSS ขนาดหน่วยความจำจริง (ชุดฝังตัว) ในหน่วยกิโลไบต์ของการประมวลผล จำนวนนี้ จะเท่ากับผลรวมของจำนวนของเซ็กเมนต์ที่ใช้งาน กับเพจเซ็กเมนต์โค้ดในหน่วยความจำคูณด้วย 4 โปรดจำไว้ว่า เพจเซ็กเมนต์โค้ด จะถูกแบ่งใช้ระหว่างอินสแตนซ์ของโปรแกรมที่กำลังรันอยู่ในปัจจุบันทั้งหมด ถ้าการประมวลผล **ksh** 26 กระบวนการกำลังรันอยู่เฉพาะหนึ่งสำเนาของเพจที่กำหนดไว้ของโปรแกรมเรียกทำงาน **ksh** จะอยู่ในหน่วยความจำ แต่คำสั่ง **ps** จะรายงานว่าขนาดเซ็กเมนต์โค้ด ที่เป็นส่วนหนึ่งของ RSS ของแต่ละอินสแตนซ์ของโปรแกรม **ksh**

TSIZ ขนาดของข้อความอิมเมจ (โปรแกรมที่แบ่งใช้) นี่คือขนาดของส่วนของข้อความ ของไฟล์เรียกทำงาน เพจของส่วนของข้อความของโปรแกรมเรียกทำงาน จะถูกนำมาไว้ในหน่วยความจำ เมื่อถูกสัมผัส นั่นคือ การแบรนช์ หรือการโหลด จำนวนนี้แสดงข้อจำกัดในส่วนสำหรับจำนวนของข้อความ ที่สามารถโหลดได้ ค่า TSIZ ไม่ได้สะท้อนถึงการใช้หน่วยความจำที่แท้จริง นี่คือค่า TSIZ ที่ยังสามารถมองเห็นได้โดยเรียกทำงานคำสั่ง **dump −ov** พร้อมกับโปรแกรมเรียกทำงาน (ตัวอย่างเช่น **dump −ov** `/usr/bin/ls`)

TRS ขนาดของชุดฝังตัว (หน่วยความจำจริง) ของข้อความ นี่คือจำนวนของเพจเซ็กเมนต์ของโค้ดคูณด้วย 4 จำนวนนี้จะเป็นจำนวนของการใช้หน่วยความจำที่เกินจริงสำหรับโปรแกรมที่มีอินสแตนซ์ จำนวนมากรันอยู่ ค่า TRS สามารถมีค่าสูงกว่าค่า TSIZ ได้ เนื่องจากเพจอื่นๆ อาจประกอบด้วยเซ็กเมนต์โค้ด เช่น ส่วนหัวของ XCOFF และส่วนของโหลดเดอร์

%MEM

คำนวณเป็นผลรวมของจำนวนของเซ็กเมนต์ใช้งานและเพจเซ็กเมนต์โค้ด ในหน่วยความจำคูณด้วย 4 (นั่นคือ ค่า

RSS) ซึ่งหารด้วยขนาดของหน่วยความจำจริงที่ใช้งานอยู่ ในเครื่องในหน่วย KB คูณ 100 แล้วจึงปัดเศษให้ใกล้เคียงกับจุดของเปอร์เซ็นต์เต็ม ค่านี้พยายามถ่ายทอดเปอร์เซ็นต์ของหน่วยความจำจริง ที่ถูกใช้โดยกระบวนการ แต่น่าเสียดาย RSS มีเจตนาเกินจริงสำหรับต้นทุนของการประมวลผล ที่แบ่งใช้ข้อความโปรแกรมกับการประมวลผลอื่น ยิ่งไปกว่านั้น การปัดเศษให้เป็นเปอร์เซ็นต์เต็มจะเป็นสาเหตุทำให้การประมวลผลทั้งหมดที่อยู่ในระบบ มีค่า RSS ต่ำกว่า 0.005 คูณขนาดของหน่วยความจำจริง เพื่อให้มี %MEM ของ 0.0

หมายเหตุ: คำสั่ง **ps** ไม่ได้บ่งชี้หน่วยความที่ใช้โดยเซ็กเมนต์หน่วยความจำที่แบ่งใช้หรือเซ็กเมนต์การแม็พหน่วยความจำ เนื่องจากแอ็พพลิเคชันจำนวนมาก ใช้หน่วยความจำที่แบ่งใช้หรือเซ็กเมนต์หน่วยความจำที่แม็พ คำสั่ง **svmon** จะเป็นเครื่องมือที่ดีกว่าที่ใช้ดูการใช้หน่วยความจำของเซ็กเมนต์เหล่านั้น

คำสั่ง svmon

คำสั่ง **svmon** นำเสนอการวิเคราะห์ที่ละเอียดมากขึ้นเกี่ยวกับ การใช้หน่วยความจำ คำสั่งนี้ให้ข้อมูลมากกว่าแต่ก็มีความเสี่ยงมากกว่าคำสั่ง **vmstat** และ **ps** คำสั่ง **svmon** ตรวจจับภาพถ่ายของสถานะปัจจุบัน ของหน่วยความจำ อย่างไรก็ตาม ภาพถ่ายที่ได้ไม่ใช่ภาพถ่ายที่แท้จริงเนื่องจากคำสั่งรันที่ระดับ ผู้ใช้ที่อาจมีการขัดจังหวะได้

เมื่อต้องการกำหนดว่าคำสั่ง **svmon** มีการติดตั้งและมีอยู่ หรือไม่ ให้รันคำสั่งต่อไปนี้:

```
# lslpp -lI bos.perf.tools
```

คำสั่ง **svmon** สามารถดำเนินการโดยผู้ใช้รากเท่านั้น

ถ้ามีการใช้ช่วงเวลา ซึ่งเป็นอ็อพชัน **–i** สถิติจะแสดงขึ้นจนกว่าคำสั่งถูก killed หรือจนกว่าจะถึงจำนวนของช่วงเวลา ซึ่งสามารถระบุได้ต่อจากช่วงเวลา

คุณสามารถใช้รายงานต่างๆ ต่อไปนี้เพื่อวิเคราะห์ข้อมูลที่แสดงขึ้น:

สากล (–G)
แสดงสถิติเกี่ยวกับหน่วยความจำจริงและพื้นที่ว่างการเพจที่ใช้อยู่ สำหรับทั้งระบบ

กระบวนการ (–P)
แสดงการใช้หน่วยความจำของกระบวนการที่ระบุซึ่งใช้งานอยู่ ถ้าไม่ได้ระบุ รายการของกระบวนการ สถิติการใช้หน่วยความจำจะแสดงกระบวนการทั้งหมดที่ใช้งานอยู่

เซกเมนต์ (–S)
แสดงการใช้หน่วยความจำของเซกเมนต์ที่ระบุ ถ้าไม่ได้ระบุ รายการของเซกเมนต์ สถิติการใช้หน่วยความจำจะแสดงเซกเมนต์ทั้งหมดที่กำหนด

เซกเมนต์โดยละเอียด (–D)
แสดงข้อมูลรายละเอียดเกี่ยวกับเซกเมนต์ที่ระบุ

ผู้ใช้ (–U)
แสดงสถิติการใช้หน่วยความจำของชื่อล็อกอินที่ระบุ ถ้าไม่ได้ระบุ รายการของชื่อล็อกอิน สถิติการใช้หน่วยความจำจะแสดงชื่อล็อกอินทั้งหมดที่กำหนด

คำสั่ง (–C)
แสดงสถิติการใช้หน่วยความจำของกระบวนการที่ระบุโดยเรียงตามชื่อ คำสั่ง

คลาสการจัดการเวิร์กโหลด (-W)

แสดงสถิติการใช้หน่วยความจำของคลาสการจัดการเวิร์กโหลดที่ ระบุ ถ้าไม่ได้ระบุคลาส สถิติการใช้หน่วยความจำจะแสดงคลาสทั้งหมดที่กำหนด

เฟรม (-F)

แสดงข้อมูลเกี่ยวกับเฟรม ถ้าไม่ได้ระบุหมายเลขเฟรม จะมีการรายงานเปอร์เซ็นต์ของหน่วยความจำที่ใช้ เฟรมเดียวที่จะพิจารณา คือเฟรมที่มีบิตการอ้างอิง ในระหว่างรอบเวลาการประมวลผล บิตการอ้างอิงทั้งหมดจะรีเซ็ต ดังนั้น เมื่อใช้อ็อพชัน **-f** ครั้งที่สอง คำสั่ง **svmon** จะรายงานเปอร์เซ็นต์ ของหน่วยความจำจริงที่มีการเข้าถึงตั้งแต่การใช้อ็อพชัน **-f** ครั้งก่อน หน้านี้ ถ้ามีการกำหนดพูลที่สำรองไว้บนระบบ จะมีการรายงานเปอร์เซ็นต์ของ หน่วยความจำที่ใช้ในแต่ละพูลที่กำหนดไว้

ระดับ (-T)

แสดงข้อมูลเกี่ยวกับระดับ เช่น หมายเลขระดับ ชื่อ superclass เมื่อใช้แฟล็ก **-a** และจำนวนหน้าทั้งหมดในหน่วยความจำจริงจากเซกเมนต์ ที่เป็นสมาชิกของระดับ

จำนวนหน่วยความจำที่ใช้งานอยู่:

คำสั่ง **svmon** สามารถแสดงข้อมูลเกี่ยวกับจำนวนของหน่วยความจำ ที่ใช้งานอยู่

หากต้องการพิมพ์ข้อมูลสถิติแบบโกลบอล ให้ใช้แฟล็ก **-G** ในตัวอย่างต่อไปนี้ จะทำซ้ำสองครั้งในช่วงเวลาหนึ่งวินาที

```
# svmon -G -i 1 2

               size      inuse       free        pin    virtual
memory      1048576     425275     623301      66521     159191
pg space     262144      31995

               work       pers       clnt
pin           46041          0          0
in use       129600     275195          0

PageSize   PoolSize      inuse       pgsp        pin    virtual
s   4 KB          -     404795      31995      46041     159191
L  16 MB          5          0          0          5          0
               size      inuse       free        pin    virtual
memory      1048576     425279     623297      66521     159195
pg space     262144      31995

               work       pers       clnt
pin           46041          0          0
in use       129604     275195          0

PageSize   PoolSize      inuse       pgsp        pin    virtual
s   4 KB          -     404799      31995      46041     159195
L  16 MB          5          0          0          5          0
```

โปรดสังเกตว่า หากเพจขนาด 4 KB พร้อมใช้งานบนระบบเท่านั้น ส่วนที่แตกข้อมูลต่อขนาดของเพจจะไม่ถูกแสดง

คอลัมน์บนรายงานที่แสดงผลลัพธ์ **svmon** จะอธิบายไว้ดังต่อไปนี้:

memory

ข้อมูลสถิติที่อธิบายถึงการใช้หน่วยความจำที่เกิดขึ้นจริง ซึ่งแสดงในเพจขนาด 4 KB

size ขนาดทั้งหมดของหน่วยความจำในเพจขนาด 4 KB

inuse จำนวนของเพจใน RAM ที่ใช้งานอยู่โดยการประมวลผลบวกกับจำนวนของเพจที่ยังคงอยู่ซึ่งเป็นเจ้าของการประมวลผลที่ยกเลิก และยังคงอยู่ใน RAM ค่านี้คือ ขนาดทั้งหมดของหน่วยความจำลบกับจำนวนของเพจ บนรายการอิสระ

free จำนวนของเพจบนรายการอิสระ

pin จำนวนของเพจที่ตรึงอยู่ใน RAM (เพจที่ตรึงอยู่คือเพจที่ยังใน RAM เสมอ และไม่สามารถส่งออกได้)

virtual จำนวนของเพจที่จัดสรรในการประมวลผลพื้นที่เสมือน

pg space

ข้อมูลสถิติที่อธิบายถึงการใช้พื้นที่การเพจ ที่แสดงในเพจที่มีขนาด 4 KB ค่าที่รายงาน คือจำนวนของเพจพื้นที่การเพจที่ใช้จริง ซึ่งบ่งชี้ว่า เพจเหล่านี้จะถูกส่งออกไปยังพื้นที่การเพจ และแตกต่างจากคำสั่ง **vmstat** ตรงที่ว่าง คอลัมน์ `avm` ของคำสั่ง **vmstat** จะแสดงหน่วยความจำเสมือนที่ประเมินผลแล้ว แต่ยังไม่ได้ถูกส่งออก

size ขนาดทั้งหมดของพื้นที่การเพจในเพจขนาด 4 KB

inuse จำนวนทั้งหมดของเพจที่ถูกจัดสรรไว้

pin ข้อมูลสถิติโดยละเอียดเกี่ยวกับเซ็ตย่อยของหน่วยความจำที่ใช้จริงซึ่งมีเพจที่ตรึงไว้ และแสดงอยู่ในกรอบขนาด 4 KB

work จำนวนของเพจการทำงานที่ตรึงไว้ใน RAM

pers จำนวนของเพจที่ยังคงอยู่ซึ่งตรึงไว้ใน RAM

clnt จำนวนของเพจไคลเอ็นต์ที่ตรึงไว้ใน RAM

in use ข้อมูลสถิติโดยละเอียดเกี่ยวกับเซ็ตย่อยของหน่วยความจำที่ใช้จริงซึ่งใช้งานอยู่ และแสดงอยู่ในกรอบขนาด 4 KB

work จำนวนของเพจการทำงานใน RAM

pers จำนวนของเพจที่ยังคงอยู่ใน RAM

clnt จำนวนของเพจไคลเอ็นต์ใน RAM (เพจไคลเอ็นต์คือเพจรีโมตไฟล์)

PageSize

แสดงให้เห็นหากเพจมีขนาดที่ไม่ใช่ 4 KB ซึ่งพร้อมใช้งานบนระบบ ให้ระบุข้อมูลสถิติแต่ละส่วนต่อขนาดเพจที่พร้อมใช้งานบนระบบ

PageSize

ขนาดของเพจ

PoolSize

จำนวนของเพจในพูลหน่วยความจำสงวน

inuse จำนวนของเพจที่ใช้

pgsp จำนวนของเพจที่จัดสรรไว้ในพื้นที่การเพจ

pin จำนวนของเพจที่ตรึงไว้

virtual จำนวนของเพจที่จัดสรรอยู่ในระบบพื้นที่เสมือน

ตัวอย่างเช่น มี 1 048 576 เพจของขนาดทั้งหมดของหน่วยความจำ คูณจำนวนนี้ด้วย 4096 เพื่อดูขนาดหน่วยความจริงที่ใช้จริงทั้งหมดในหน่วยไบต์ (4 GB) ขณะที่ 425 275 เพจกำลังใช้งานอยู่ ซึ่งมี 623 301 เพจบนรายการอิสระ และ 66 521 เพจถูกตรึงไว้ใน RAM จากเพจทั้งหมดที่ใช้งานอยู่ จะมี 129 600 เพจการทำงานใน RAM และมี 275 195 เพจที่ยังคงอยู่ใน RAM และ 0 เพจไคลเอ็นต์ใน RAM ผลรวมของทั้งสามส่วนนี้ บวกกับหน่วยความจำที่สงวนไว้แต่ไม่ได้ใช้โดยพูลที่สงวนไว้ จะเท่ากับคอลัมน์ *inuse* ของส่วนของ *memory* ส่วนของ *pin* หารด้วยขนาดหน่วยความจำที่ตรึงไว้ในหมวดหมู่การทำงาน หมวดหมู่การคงอยู่ และหมวดหมู่ไคลเอ็นต์ ผลรวมทั้งสามหมวดหมู่นี้ บวกกับหน่วยความจำที่สงวนไว้โดยพูลที่สงวนไว้ ซึ่งจะถูกตรึงไว้เสมอ จะเท่ากับคอลัมน์ *pin* ของส่วนของ *memory* มี 262 144 เพจ (1 GB) ของพื้นที่การเพจทั้งหมด และมี 31 995 เพจที่ใช้งานอยู่ คอลัมน์ *inuse* ของ *memory* จะมีขนาดใหญ่กว่าคอลัมน์ *inuse* ของ *pg space* เนื่องจากหน่วยความจำสำหรับเพจไฟล์ไม่ว่าง เมื่อโปรแกรมเสร็จสิ้นการทำงาน ขณะที่จัดสรรพื้นที่การเพจ

การใช้หน่วยความจำโดยการประมวลผล:

คำสั่ง **svmon -P** จะแสดงข้อมูลสถิติการใช้หน่วยความจำ สำหรับการประมวลผลทั้งหมดที่รันอยู่บนระบบ

ต่อไปนี้คือตัวอย่างของคำสั่ง **svmon -P** :

```
# svmon -P

------------------------------------------------------------------------------
     Pid Command          Inuse        Pin     Pgsp    Virtual 64-bit Mthrd 16MB
   16264 IBM.ServiceRM    10075       3345     3064      13310      N     Y    N

     PageSize      Inuse        Pin       Pgsp    Virtual
     s   4 KB      10075       3345       3064      13310
     L  16 MB          0          0          0          0

    Vsid      Esid Type Description              PSize  Inuse   Pin Pgsp Virtual
   f001e         d work shared library text          s   4857     0   36  6823
       0         0 work kernel seg                  s   4205  3335 2674  5197
   b83f7         2 work process private             s    898     2  242  1098
   503ea         f work shared library data         s     63     0   97   165
   c8439         1 pers code,/dev/hd2:149841        s     28     0    -     -
   883f1         - work                             s     21     8   14    26
   e83dd         - pers /dev/hd2:71733              s      2     0    -     -
   f043e         4 work shared memory segment       s      1     0    1     1
   c0438         - pers large file /dev/hd9var:243  s      0     0    -     -
   b8437         3 mmap mapped to sid a03f4         s      0     0    -     -
   583eb         - pers large file /dev/hd9var:247  s      0     0    -     -

------------------------------------------------------------------------------
     Pid Command          Inuse        Pin     Pgsp    Virtual 64-bit Mthrd 16MB
   17032 IBM.CSMAgentR     9791       3347     3167      12944      N     Y    N

     PageSize      Inuse        Pin       Pgsp    Virtual
     s   4 KB       9791       3347       3167      12944
     L  16 MB          0          0          0          0

    Vsid      Esid Type Description              PSize  Inuse   Pin Pgsp Virtual
   f001e         d work shared library text          s   4857     0   36  6823
       0         0 work kernel seg                  s   4205  3335 2674  5197
     400         2 work process private             s    479     2  303   674
   38407         f work shared library data         s    120     0  127   211
```

```
   a83f5         1 pers code,/dev/hd2:149840         s     99     0    -     -
   7840f         - work                              s     28    10   27    39
   e83dd         - pers /dev/hd2:71733               s      2     0    -     -
   babf7         - pers /dev/hd2:284985              s      1     0    -     -
   383e7         - pers large file /dev/hd9var:186   s      0     0    -     -
   e03fc         - pers large file /dev/hd9var:204   s      0     0    -     -
   f839f         3 mmap mapped to sid 5840b          s      0     0    -     -
[...]
```

เอาต์พุตของคำสั่งจะแสดงรายละเอียดทั้งหน่วยความจำแบบโกลบอลที่ใช้ต่อการประมวลผล และยังหน่วยความจำที่ใช้ต่อเซ็กเมนต์โดยละเอียดซึ่งจะถูกใช้โดยการประมวลผลแต่ละตัวที่รายงาน กฎการเรียงลำดับที่เป็นค่าดีฟอลต์ จะลดการเรียงลำดับลงตามจำนวนเพจ Inuse คุณสามารถเปลี่ยนกฎการเรียงลำดับได้โดยใช้คำสั่ง **svmon** พร้อมกับแฟล็ก **-u, -p, -g** หรือ **-v**

สำหรับสรุปของการประมวลผลสิบห้าอันดับแรกที่ใช้หน่วยความจำบนระบบ จะใช้คำสั่งต่อไปนี้:

```
# svmon -Pt15 | perl -e 'while(<>){print if($.==2||$&&&!$s++);$.=0 if(/^-+$/)}'
-------------------------------------------------------------------------------
     Pid Command          Inuse        Pin     Pgsp    Virtual 64-bit Mthrd 16MB
   16264 IBM.ServiceRM    10075       3345     3064      13310      N     Y    N
   17032 IBM.CSMAgentR     9791       3347     3167      12944      N     Y    N
   21980 zsh               9457       3337     2710      12214      N     N    N
   22522 zsh               9456       3337     2710      12213      N     N    N
   13684 getty             9413       3337     2710      12150      N     N    N
   26590 perl5.8.0         9147       3337     2710      12090      N     N    N
    7514 sendmail          9390       3337     2878      12258      N     N    N
   14968 rmcd              9299       3340     3224      12596      N     Y    N
   18940 ksh               9275       3337     2710      12172      N     N    N
   14424 ksh               9270       3337     2710      12169      N     N    N
    4164 errdemon          9248       3337     2916      12255      N     N    N
    3744 cron              9217       3337     2770      12125      N     N    N
   11424 rpc.mountd        9212       3339     2960      12290      N     Y    N
   21564 rlogind           9211       3337     2710      12181      N     N    N
   26704 rlogind           9211       3337     2710      12181      N     N    N
```

Pid 16 264 คือ ID การประมวลผลที่มีการใช้หน่วยความจำสูงมากที่สุด คำสั่ง บ่งชี้ถึงชื่อคำสั่ง ในกรณีนี้ IBM.ServiceRM คอลัมน์ Inuse ซึ่งคือจำนวนทั้งหมดของเพจในหน่วยความจำจริง จากเซ็กเมนต์ที่ถูกใช้โดยการประมวลผล ซึ่งแสดง 10 075 เพจ แต่ละเพจจะมีขนาด 4 KB คอลัมน์ Pin ซึ่งคือจำนวนทั้งหมดของเพจที่ตรึงจากเซ็กเมนต์ ที่ถูกใช้โดยการประมวลผล ซึ่งแสดง 3 345 เพจ คอลัมน์ Pgsp ซึ่งคือจำนวนของเพจพื้นที่การเพจที่ถูกใช้โดยการประมวลผล ซึ่งแสดง 3 064 เพจ คอลัมน์ Virtual (จำนวนทั้งหมดของเพจในการประมวลผลพื้นที่เสมือน) ซึ่งแสดง 13 310

ส่วนรายละเอียดจะแสดงข้อมูลเกี่ยวกับแต่ละเซ็กเมนต์สำหรับแต่ละการประมวลผล ที่แสดงอยู่ในส่วนของสรุป ซึ่งประกอบด้วย segment identifier Vsid เสมือน และ Esid ที่มีประสิทธิภาพ Esid จะสะท้อนถึงการลงทะเบียนเซ็กเมนต์ ซึ่งถูกใช้เพื่อเข้าถึงเพจที่สอดคล้องกัน ชนิดของเซ็กเมนต์นี้ ยังแสดงข้อมูลพร้อมกับคำอธิบายที่มีอยู่ในคำอธิบายเชิงข้อความของเซ็กเมนต์ ซึ่งประกอบด้วยชื่อวอลุ่มและ i-node ของไฟล์สำหรับเซ็กเมนต์ที่ยังคงอยู่ รายงานยังแสดงรายละเอียดเกี่ยวกับขนาดของเพจที่เซ็กเมนต์นั้นรองรับ โดยที่ s หมายความถึงเพจที่มีขนาด 4 KB และ L หมายความถึงเพจที่มีขนาด 16 MB จำนวนของเพจใน RAM นั่นคือ Inuse จำนวนของเพจที่ตรึงไว้ใน RAM นั่นคือ Pin จำนวนของเพจในพื้นที่การเพจ นั่นคือ Pgsp และจำนวนของเพจเสมือน นั่นคือ Virtual

คุณสามารถใช้อ็อพชันเพิ่มเติมได้เพื่อขอรับรายละเอียดเพิ่มเติม อ็อพชัน **–j** จะแสดงพาธของไฟล์สำหรับเซ็กเมนต์ที่ยังคงอยู่ อ็อพชัน **–l** จัดเตรียมรายละเอียดเพิ่มเติมสำหรับเซ็กเมนต์ และอ็อพชัน **–r** จะแสดงช่วงของหน่วยความจำที่ใช้โดยเซ็กเมนต์แต่ละส่วน ต่อไปนี้คือตัวอย่างของคำสั่ง **svmon** พร้อมกับอ็อพชัน **–l, –r** และ **–j** :

```
# svmon -S f001e 400 e83dd -l -r -j

    Vsid      Esid Type Description              PSize  Inuse   Pin Pgsp Virtual
   f001e         d work shared library text          s   4857     0   36  6823
                   Addr Range: 0..60123
                   Shared library text segment
     400         2 work process private             s    480     2  303   675
                   Addr Range: 0..969 : 65305..65535
                   pid(s)=17032
   e83dd         - pers /dev/hd2:71733              s      2     0    -     -
                        /usr/lib/nls/loc/uconvTable/ISO8859-1
                   Addr Range: 0..1
                   pid(s)=17552, 17290, 17032, 16264, 14968, 9620
```

Address Range ระบุหนึ่งช่วงสำหรับเซ็กเมนต์ที่ยังคงอยู่ หรือไคลเอ็นต์เซ็กเมนต์ และสองช่วงสำหรับเซ็กเมนต์การทำงาน ช่วงสำหรับเซ็กเมนต์ที่ยังคงอยู่ หรือไคลเอ็นต์เซ็กเมนต์จะใช้รูปแบบ '0..x' โดยที่ x คือจำนวนสูงสุดของเพจเสมือน ที่ได้นำมาใช้แล้ว ฟิลด์ช่วงสำหรับเซ็กเมนต์การทำงานสามารถอยู่ในรูปแบบ '0..x : y..65535' โดยที่ 0..x มีข้อมูลโกลบอลและจะโตขึ้นเรื่อยๆ และ y..65535 จะมีพื้นที่สแต็กและจะโตลงเรื่อยๆ สำหรับช่วงแอดเดรสในเซ็กเมนต์การทำงาน พื้นที่จะถูกจัดสรรโดยเริ่มต้นจากปลายสุดทั้งสองด้าน และจะทำงานไปจนถึง กึ่งกลาง ถ้าเซ็กเมนต์การทำงานไม่ใช่แบบส่วนตัว (เคอร์เนลหรือไลบรารีที่แบ่งใช้) พื้นที่จะถูกจัดสรรแตกต่างกัน

จากตัวอย่างข้างต้น ID เซ็กเมนต์ 400 คือเซ็กเมนต์การทำงานแบบส่วนตัว ช่วงของแอดเดรสคือ 0..969 : 65305..65535 ID เซ็กเมนต์ f001e คือข้อความไลบรารีที่แบ่งใช้เซ็กเมนต์การทำงาน ช่วงของแอดเดรสคือ 0..60123

เซ็กเมนต์สามารถนำมาใช้ได้โดยการประมวลผลจำนวนมาก เพจแต่ละเพจในหน่วยความจำที่ใช้จริงจากเซ็กเมนต์ จะแสดงอยู่ในฟิลด์ Inuse สำหรับแต่ละการประมวลผล ที่ใช้เซ็กเมนต์นั้น ดังนั้น จำนวนทั้งหมดสำหรับ Inuse อาจมีค่าเกินจำนวนทั้งหมดของเพจ ในหน่วยความจำที่ใช้จริง และจะเป็นจริงสำหรับฟิลด์ Pgsp และ Pin ด้วยเช่นกัน ค่าที่แสดงอยู่ในส่วนของสรุปประกอบด้วยผลรวมของ Inuse, Pin และ Pgsp และจำนวน Virtual ของเซ็กเมนต์ทั้งหมดที่ใช้โดยการประมวลผล

จากตัวอย่างข้างต้น เซ็กเมนต์ e83dd จะถูกใช้โดยการประมวลผลทั้งหลาย ซึ่งมี PID ที่เป็น 17552, 17290, 17032, 16264, 14968 และ 9620

ข้อมูลโดยละเอียดเกี่ยวกับ ID เซ็กเมนต์ที่ระบุเฉพาะ:

อ็อพชัน **–D** แสดงข้อมูลสถิติการใช้หน่วยความจำโดยละเอียด สำหรับเซ็กเมนต์

ต่อไปนี้คือตัวอย่าง:

```
# svmon -D 38287 -b
Segid: 38287
Type:  working
PSize: s (4 KB)
Address Range: 0..484
Size of page space allocation: 2 pages (  0,0 MB)
Virtual: 18 frames ( 0,1 MB)
Inuse: 16 frames ( 0,1 MB)
```

```
Page   Psize    Frame      Pin        Ref        Mod  ExtSegid    ExtPage
 341     s       527720     N          N          N        -           -
 342     s       996079     N          N          N        -           -
 343     s       524936     N          N          N        -           -
 344     s       985024     N          N          N        -           -
 347     s       658735     N          N          N        -           -
 348     s        78158     N          N          N        -           -
 349     s       174728     N          N          N        -           -
 350     s       758694     N          N          N        -           -
 404     s       516554     N          N          N        -           -
 406     s       740622     N          Y          N        -           -
 411     s       528313     N          Y          Y        -           -
 412     s      1005599     N          Y          N        -           -
 416     s       509936     N          N          Y        -           -
 440     s       836295     N          N          Y        -           -
 443     s        60204     N          N          Y        -           -
 446     s       655288     N          N          Y        -           -
```

คำอธิบายของคอลัมน์มีดังต่อไปนี้:

Page ระบุดัชนีของเพจภายในเซ็กเมนต์

Psize ระบุขนาดของหน้า (**s** สำหรับ 4 KB **m** สำหรับ 64 KB **L** สำหรับ 16 MB และ **S** สำหรับ 16 GB)

Frame ระบุดัชนีของกรอบหน่วยความจำจริงที่เพจนั้นตั้งอยู่

Pin ระบุแฟล็กที่บ่งชี้ถึงเพจที่ตรึงไว้

Ref ระบุด้วยแฟล็ก **-b** เท่านั้น ระบุแฟล็กที่บ่งชี้ถึงบิตการอ้างอิงของเพจ ว่าเปิดอยู่

Mod ระบุด้วยแฟล็ก **-b** เท่านั้น ระบุแฟล็กที่บ่งชี้ถึงเพจที่แก้ไข

ExtSegid
ในกรณีของเพจที่เป็นของเซ็กเมนต์ที่ขยายเพิ่มซึ่งลิงก์กับเซ็กเมนต์ที่ตรวจสอบ ตัวระบุเซ็กเมนต์เสมือนของเซ็กเมนต์นี้จะถูกแสดง

ExtPage
ในกรณีที่เพจเป็นของเซ็กเมนต์ที่ขยายเพิ่มซึ่งลิงก์กับเซ็กเมนต์ที่ตรวจสอบ ดัชนีของเพจภายในเซ็กเมนต์ที่ขยายเพิ่มจะถูกแสดง

เมื่อเซ็กเมนต์ที่ขยายเพิ่มถูกลิงก์กับเซ็กเมนต์ที่ตรวจสอบ รายงานจะดูคล้ายกับที่แสดงดังต่อไปนี้:

```
 Page     Psize   Frame     Pin        Ref        Mod  ExtSegid    ExtPage
65574     s       345324     N          N          N    288071         38
65575     s       707166     N          N          N    288071         39
65576     s       617193     N          N          N    288071         40
```

แฟล็ก **-b** จะแสดงสถานะของการอ้างอิง และบิตที่แก้ไขของกรอบที่แสดงทั้งหมด หลังจากที่แสดงแล้ว บิตที่อ้างอิงของกรอบจะถูกตั้งค่า เมื่อใช้กับแฟล็ก **-i** ระบบจะตรวจสอบกรอบที่สามารถเข้าถึงได้ ระหว่างช่วงเวลาแต่ละช่วง

หมายเหตุ: เนื่องจากผลกระทบต่อผลการทำงาน ให้ใช้แฟล็ก **-b** ด้วยความระมัดระวัง

รายการของการใช้หน่วยความจำบนสุดของเซ็กเมนต์:

อ็อพชัน **–S** จะถูกใช้เพื่อเรียงลำดับเซ็กเมนต์ตามการใช้หน่วยความจำ และแสดงข้อมูลสถิติการใช้หน่วยความจำสำหรับเซ็กเมนต์ที่ระบุ ถ้าไม่มีรายการของเซ็กเมนต์ที่จัดหาไว้ ข้อมูลสถิติการใช้หน่วยความจำจะแสดงเซ็กเมนต์ที่กำหนดไว้ทั้งหมด

คำสั่งต่อไปนี้จะเรียงลำดับระบบและเซ็กเมนต์ที่ไม่ใช่ของระบบด้วยจำนวนของหน้า ในหน่วยความจำที่ใช้จริง อ็อพชัน **–t** สามารถนำมาใช้เพื่อจำกัดจำนวนของเซ็กเมนต์ที่แสดง กับจำนวนที่ระบุไว้ แฟล็ก **–u** จะเรียงลำดับเอาต์พุตตามลำดับจากมากไปหาน้อยตามจำนวนทั้งหมดของเพจในหน่วยความจำที่ใช้จริง

ต่อไปนี้คือตัวอย่างเอาต์พุตของคำสั่ง **svmon** ด้วยอ็อพชัน **–S, –t** และ **–u** :

```
# svmon -Sut 10

    Vsid      Esid Type Description              PSize  Inuse   Pin Pgsp Virtual
   70c4e         - pers large file /dev/lv01:26      s  84625     0    -     -
   22ec4         - work                              s  29576     0    0 29586
   8b091         - pers /dev/hd3:123                 s  24403     0    -     -
   7800f         - work kernel heap                  s  22050  3199 19690 22903
   a2db4         - pers /dev/hd3:105                 s  15833     0    -     -
   80010         - work page frame table             s  15120 15120    0 15120
   7000e         - work misc kernel tables           s  13991     0 2388 14104
   dc09b         - pers /dev/hd1:28703               s   9496     0    -     -
   730ee         - pers /dev/hd3:111                 s   8568     0    -     -
   f001e         - work                              s   4857     0   36  6823
```

สหสัมพันธ์ระหว่างเอาต์พุตคำสั่ง svmon และ vmstat

นี่คือสหสัมพันธ์ระหว่างเอาต์พุต **svmon** และ **vmstat**

ต่อไปนี้คือตัวอย่างเอาต์พุตจากคำสั่ง **svmon** :

```
# svmon -G
               size      inuse       free        pin    virtual
memory      1048576     417374     631202      66533     151468
pg space     262144      31993

              work       pers       clnt
pin           46053          0          0
in use       121948     274946          0

PageSize   PoolSize      inuse       pgsp        pin    virtual
s   4 KB          -     397194     262144      46053     151468
L  16 MB          5          0          0          5          0
```

คำสั่ง **vmstat** จะถูกรันอยู่ที่อีกหน้าต่างหนึ่ง ขณะที่คำสั่ง **svmon** จะถูกรัน รายงาน **vmstat** จะแสดงดังนี้:

```
# vmstat 3
kthr     memory             page              faults        cpu
----- ----------- ------------------------ ------------ -----------
 r  b   avm   fre  re  pi  po  fr   sr  cy  in   sy  cs us sy id wa
 1  5 205031 749504   0   0   0   0    0   0 1240  248 318  0  0 99  0
 2  2 151360 631310   0   0   3   3   32   0 1187 1718 641  1  1 98  0
```

```
 1  0 151366 631304   0   0   0   0    0   0 1335 2240 535  0  1 99  0
 1  0 151366 631304   0   0   0   0    0   0 1303 2434 528  1  4 95  0
 1  0 151367 631303   0   0   0   0    0   0 1331 2202 528  0  0 99  0
```

รายงาน **svmon** แบบโกลบอลจะแสดงจำนวนที่เกี่ยวข้อง คอลัมน์ `fre` ของคำสั่ง **vmstat** จะเกี่ยวข้องกับคอลัมน์ `memory free` ของคำสั่ง **svmon** ค่า Active Virtual Memory, `avm` ของรายงานคำสั่ง **vmstat** จะคล้ายกับค่าหน่วยความจำเสมือน ที่คำสั่ง **svmon** รายงาน

ความสัมพันธ์ระหว่างเอาต์พุตของคำสั่ง svmon และ ps

มีความสัมพันธ์บางอย่างระหว่างเอาต์พุตของคำสั่ง **svmon** และ **ps**

ตัวอย่างที่ 1

ต่อไปนี้เป็นตัวอย่างสำหรับเอาต์พุตของคำสั่ง **svmon** และ **ps**:

```
# # ps v 405528
    PID    TTY STAT  TIME PGIN  SIZE   RSS   LIM  TSIZ   TRS %CPU %MEM COMMAND
 405528  pts/0 A    43:11    1   168   172 32768     1     4 99.5  0.0 yes

(0) root @ clock16: 6.1.2.0: /
# svmon -O unit=KB,segment=category,filtercat=exclusive -P 405528
Unit: KB
--------------------------------------------------------------------------------
     Pid Command          Inuse      Pin     Pgsp  Virtual
  405528 yes                172       16        0      168

................................................................................
EXCLUSIVE segments                  Inuse      Pin     Pgsp  Virtual
                                      172       16        0      168

    Vsid      Esid Type Description             PSize  Inuse   Pin Pgsp Virtual
   554f1         f work shared library data         s     92     0    0      92
   49416         2 work process private             s     76    16    0      76
   6d49f         1 clnt code,/dev/hd2:338           s      4     0    -       -
```

เอาต์พุตของคำสั่ง **ps** ด้านบนจะแสดง `SIZE` เป็น 168 และ `RSS` เป็น 172 การใช้คำสั่ง **svmon** ด้านบนจะให้ทั้งสองค่า

คุณสามารถใช้ค่าเอาต์พุตจากคำสั่ง **svmon** ที่แสดงด้านลบพร้อมกับสมการต่อไปนี้เพื่อคำนวณ `SIZE` และ `RSS`:

```
SIZE = Work Process Private Memory Usage in KB + Work Shared Library Data Memory Usage in KB
RSS = SIZE + Text Code Size (Type=clnt, Description=code,)
```

การใช้ค่าในตัวอย่างด้านบนคุณจะได้รับค่าต่อไปนี้:

```
SIZE = 92 + 76 = 168
RSS = 168 + 4 = 172
```

ตัวอย่างที่ 2

ต่อไปนี้เป็นตัวอย่างสำหรับเอาต์พุตของคำสั่ง **svmon** และ **ps**:

```
# ps v 282844
    PID    TTY STAT  TIME PGIN  SIZE   RSS   LIM  TSIZ   TRS %CPU %MEM COMMAND
 282844      - A    15:49  322 24604 25280    xx   787   676  0.0  3.0 /usr/sbin/rsct/b

(0) root @ clock16: 6.1.2.0: /
# svmon -O unit=KB,segment=category,filtercat=exclusive -P 282844
Unit: KB
--------------------------------------------------------------------------------
     Pid Command          Inuse      Pin     Pgsp  Virtual
  282844 IBM.CSMAgentR    25308       16        0    24604

................................................................................
EXCLUSIVE segments                  Inuse      Pin     Pgsp  Virtual
                                    25308       16        0    24604

    Vsid      Esid Type Description              PSize  Inuse   Pin Pgsp Virtual
   2936e         2 work process private              s  23532    16    0   23532
   2d36f         f work shared library data          s   1072     0    0    1072
    1364         1 clnt code,/dev/hd2:81988          s    676     0    -       -
   154c1         - clnt /dev/hd9var:353              s     16     0    -       -
   41494         - clnt /dev/hd2:82114               s      8     0    -       -
   4d3d7         - clnt /dev/hd9var:357              s      4     0    -       -
   7935a         - clnt /dev/hd9var:307              s      0     0    -       -
   4d377         3 mmap maps 2 source(s)             s      0     0    -       -
   3934a         - clnt /dev/hd9var:300              s      0     0    -       -
```

เอาต์พุตของคำสั่ง **ps** ด้านบนจะแสดง SIZE เป็น 24604 และ RSS เป็น 25280

คุณสามารถใช้ค่าเอาต์พุตจากคำสั่ง **svmon** ที่แสดงด้านลบพร้อมกับสมการต่อไปนี้เพื่อคำนวณ SIZE และ RSS:

```
SIZE = Work Process Private Memory Usage in KB + Work Shared Library Data Memory Usage in KB
RSS = SIZE + Text Code Size (Type=clnt, Description=code,)
```

การใช้ค่าในตัวอย่างด้านบนคุณจะได้รับค่าต่อไปนี้:

```
SIZE = 23532 + 1072 = 24604
RSS = 24604 + 676 = 25280
```

การคำนวณข้อกำหนดหน่วยความจำต่ำที่สุด

ข้อกำหนดด้านหน่วยความจำต่ำสุดของโปรแกรมสามารถคำนวณได้ง่ายขึ้น

```
เพจหน่วยความจำทั้งหมด (หน่วยขนาด 4 KB) = T + ( N * ( PD + LD ) ) + F
```

โดยที่:

T = จำนวนของเพจสำหรับข้อความ (แบ่งใช้โดยผู้ใช้ทั้งหมด)

N = จำนวนสำเนาของโปรแกรมนี้รันอยู่อย่างพร้อมเพียงพอ

PD = จำนวนของเพจเซ็กเมนต์ใช้งานในการประมวลผลเซ็กเมนต์ส่วนบุคคล

LD = จำนวนของเพจข้อมูลไลบรารีที่แบ่งใช้โดยใช้การประมวลผล

F = จำนวนของเพจไฟล์ (แบ่งใช้โดยผู้ใช้ทั้งหมด)

คูณผลลัพธ์ด้วย 4 เพื่อขอรับจำนวนของกิโลไบต์ที่ต้องการ คุณอาจต้องการเพิ่มในเคอร์เนล ส่วนขยายเคอร์เนล และค่าเซ็กเมนต์ข้อความไลบรารีที่แบ่งใช้ กับค่านี้แม้ว่า ค่านี้จะแบ่งใช้โดยการประมวลผลทั้งหมด บนระบบ ตัวอย่างเช่น แอ็พพลิเคชันบางตัว เช่น CATIA และฐานข้อมูลใช้โมดูลไลบรารีที่แบ่งใช้ ที่มีขนาดใหญ่มาก หมายเหตุ เนื่องจากเรามีข้อมูลสถิติที่ใช้จาก snapshot เดี่ยว จึงไม่มีการรับประกันค่านั้นที่เราได้รับมาจากสูตร ว่าเป็นค่าที่ถูกต้องสำหรับขนาดชุดการทำงานต่ำสุด ของการประมวลผล หากต้องการรับขนาดชุดการทำงาน คุณต้องรันเครื่องมือ เช่น คำสั่ง **rmss** หรือใช้ snapshot จำนวนมาก ในระหว่างอายุของการประมวลผล และพิจารณาค่าเฉลี่ยจาก snapshot เหล่านั้น โปรดดู “การประเมินผลข้อกำหนดด้านหน่วยความจำด้วยคำสั่ง rmss” ในหน้า 155 สำหรับข้อมูลเพิ่มเติม

โปรแกรมหน่วยความจำรั่ว

หน่วยความจำรั่ว คือข้อผิดพลาดของโปรแกรมที่ประกอบด้วยการจัดสรรหน่วยความจำซ้ำๆ การใช้หน่วยความจำ และการละทิ้งคืนหน่วยความจำ

หน่วยความจำรั่วในโปรแกรมที่รันค่อนข้างนาน เช่น แอ็พพลิเคชันแบบโต้ตอบ คือปัญหาที่รุนแรง เนื่องจากอาจส่งผลถึงการแตกแฟรกเมนต์หน่วยความจำ และการสะสมจำนวนที่มีขนาดใหญ่ของเพจที่เต็มไปด้วยขยะในหน่วยความจำจริง และพื้นที่เพจ ระบบจะทราบว่ารันโดยมีพื้นที่เพจที่ไม่เพียงพอ เนื่องจากหน่วยความจำรั่วในโปรแกรมเดี่ยว

หน่วยความจำรั่วสามารถถูกตรวจพบด้วยคำสั่ง **svmon** โดยมองหาการประมวลผลที่มีเซ็กเมนต์ใช้งานที่โตขึ้นอย่างต่อเนื่อง การรั่วในเซ็กเมนต์เคอร์เนล สามารถมีสาเหตุมาจาก mbuf หรือโดยไดร์เวอร์อุปกรณ์ ส่วนขยายเคอร์เนล หรือเคอร์เนล หากต้องการพิจารณาเซ็กเมนต์ที่กำลังโต ให้ใช้คำสั่ง **svmon** ด้วยอ็อพชัน **–i** เพื่อดูการประมวลผลหรือกลุ่มของการประมวลผล และดูถ้าเซ็กเมนต์ใดๆ ที่ยังคงมีการโต

การระบุรูทีนการรบกวนหรือบรรทัดของโค้ดจะมีความยากขึ้น โดยเฉพาะในแอ็พพลิเคชัน AIXwindows ซึ่งจะสร้างจำนวนที่มีขนาดใหญ่ของการเรียก **malloc( )** และ **free( )** C++ จะเตรียม HeapView Debugger สำหรับการใช้เพื่อการวิเคราะห์/การปรับและการรั่ว โปรแกรมในกลุ่มที่สามบางกลุ่มสำหรับการวิเคราะห์รอยรั่ว แต่โปรแกรมเหล่านั้นต้องการเข้าถึง ซอร์สโค้ดโปรแกรม

การใช้รูทีน **realloc( )** บางตัว ขณะที่ไม่มีข้อผิดพลาดในการโปรแกรมมิง สามารถได้รับผลกระทบแบบเดียวกันกับหน่วยความจำรั่ว ถ้าโปรแกรมใช้รูทีนย่อย **realloc( )** บ่อยเพื่อเพิ่มขนาดของพื้นที่ข้อมูล เซ็กเมนต์ใช้งานของการประมวลผลสามารถเป็นการแฟรกเมนต์ หากหน่วยเก็บที่ถูกปล่อยโดยรูทีนย่อย **realloc( )** ไม่สามารถนำกลับมาใช้ได้อีก

ใช้การเรียกของระบบ **disclaim( )** และการเรียก **free( )** เพื่อปล่อยหน่วยความจำที่ไม่ได้ใช้งานอีกต่อไป การเรียกของระบบ **disclaim( )** ต้องถูกเรียกก่อนที่จะเรียก **free( )** ซึ่งจะไม่ได้ใช้เวลา CPU เพื่อล้างหน่วยความอิสระ หลังจากการเรียก **malloc( )** ถ้าโปรแกรมจะเสร็จสิ้นในไม่ช้า เมื่อโปรแกรมยกเลิก เซ็กเมนต์การทำงานจะถูกทำลาย และกรอบของเพจหน่วยความจำ ที่มีข้อมูลเซ็กเมนต์การทำงานจะถูกเพิ่มให้กับรายการอิสระ ตัวอย่างต่อไปนี้คือโปรแกรมที่มีการรั่วของหน่วยความจำซึ่งค่า `Inuse`, `Pgspace` และ `Address Range` ของเซ็กเมนต์การทำงานส่วนตัวจะยังคงโตขึ้น :

```
# svmon -P 13548 -i 1 3
  Pid                     Command        Inuse        Pin      Pgsp  Virtual 64-bit Mthrd LPage
13548                       pacman         8535       2178       847     8533      N     N     N

Vsid     Esid Type  Description          LPage  Inuse    Pin  Pgsp  Virtual
   0        0 work  kernel seg               -   4375   2176   847     4375
48412       2 work  process private          -   2357      2     0     2357
6c01b       d work  shared library text      -   1790      0     0     1790
4c413       f work  shared library data      -     11      0     0       11
3040c       1 pers  code,/dev/prodlv:4097    -      2      0     -        -
```

```
ginger :svmon -P 13548 -i 1 3

  Pid                     Command        Inuse        Pin      Pgsp  Virtual 64-bit Mthrd LPage
13548                      pacman         8589       2178       847     8587      N     N     N

Vsid     Esid Type  Description          LPage  Inuse    Pin  Pgsp  Virtual
   0        0 work  kernel seg             -   4375   2176   847     4375
48412       2 work  process private        -   2411      2     0     2411
6c01b       d work  shared library text    -   1790      0     0     1790
4c413       f work  shared library data    -     11      0     0       11
3040c       1 pers  code,/dev/prodlv:4097  -      2      0     -        -

  Pid                     Command        Inuse        Pin      Pgsp  Virtual 64-bit Mthrd LPage
13548                      pacman         8599       2178       847     8597      N     N     N

 Vsid     Esid Type  Description          LPage  Inuse    Pin  Pgsp  Virtual
    0        0 work  kernel seg             -   4375   2176   847     4375
48412        2 work  process private        -   2421      2     0     2421
6c01b        d work  shared library text    -   1790      0     0     1790
4c413        f work  shared library data    -     11      0     0       11
3040c        1 pers  code,/dev/prodlv:4097  -      2      0     -        -
```

การประเมินผลข้อกำหนดด้านหน่วยความจำด้วยคำสั่ง rmss

สำหรับคำสั่ง **rmss** Reduced-Memory System Simulator จะจัดเตรียมความหมายสำหรับการจำลองขนาดต่างๆ ของหน่วยความจำที่ใช้จริง ซึ่งมีขนาดเล็กกว่าเครื่องจริงของคุณ โดยไม่มีการแตก และแทนที่บอร์ดหน่วยความจำ มากไปกว่านั้น คำสั่ง **rmss** จะจัดเตรียมตัวช่วยเพื่อรันแอ็พพลิเคชันผ่านช่วงของขนาดหน่วยความจำ การแสดงผล สำหรับขนาดหน่วยความจำแต่ละขนาด ข้อมูลสถิติผลการทำงาน เช่น เวลาตอบสนองของแอ็พพลิเคชัน และจำนวนของการเพจ

คำสั่ง **rmss** จะถูกออกแบบมาเพื่อช่วยคุณตอบคำถาม: "จำนวนเมกะไบต์ของหน่วยความจำที่ใช้จริงที่ระบบต้องการ เพื่อรันระบบปฏิบัติการ และแอ็พพลิเคชันที่กำหนดด้วยระดับความสามารถในการยอมรับ ของผลการทำงาน?" ในบริบทแบบผู้ใช้หลายราย จะออกแบบมาเพื่อช่วยคุณตอบคำถาม: "จำนวนผู้ใช้ที่สามารถรันแอ็พพลิเคชันนี้ พร้อมกันในเครื่องด้วย *X* เมกะไบต์ของหน่วยความจำที่ใช้จริง?"

การใช้หลักสำหรับคำสั่ง **rmss** คือเครื่องมือการประมาณความสามารถ เพื่อพิจารณาจำนวนหน่วยความจำที่เวิร์กโหลดต้องการ ซึ่ง สามารถนำมาใช้กับเครื่องมือการกำหนดปัญหา โดยเฉพาะอย่างยิ่งสำหรับในกรณีที่มีหน่วยความจำเพิ่มเติมที่ลดระดับผลการทำงาน

หากต้องการพิจารณาว่า ได้ติดตั้งคำสั่ง **rmss** แล้วและพร้อมใช้งาน ให้รันคำสั่งต่อไปนี้:

```
# lslpp -lI bos.perf.tools
```

ไม่ว่าคำสั่ง **rmss** จะเปลี่ยนขนาดหน่วยความหรือไม่ก็ตาม minperm และ maxperm ต้องถูกปรับเปลี่ยนเป็นพารามิเตอร์ใหม่ และจำนวนเพจ lruable ไม่เปลี่ยนแปลงเพื่อให้เหมาะกับขนาดของหน่วยความจำ ที่จำลอง เหตุการณ์นี้สามารถนำไปสู่ลักษณะการทำงานที่ไม่ได้คาดคิดไว้โดยที่แคชบัฟฟอร์จะโตขึ้นนอกสัดส่วน ตามลำดับ ระบบสามารถมีหน่วยความจำไม่เพียงพอได้

ซึ่งเป็นสิ่งสำคัญในการระลึกไว้เสมอว่า ขนาดหน่วยความจำที่จำลองโดยคำสั่ง **rmss** คือขนาดทั้งหมดของหน่วยความจำที่เกิดขึ้นจริงของเครื่อง ซึ่งประกอบด้วยหน่วยความจำที่ใช้โดยระบบปฏิบัติการ และโปรแกรมอื่นๆ ที่อาจรันอยู่ ซึ่งไม่ใช่จำนวนของหน่วยความจำ ที่ใช้เฉพาะโดยตัวของแอ็พพลิเคชันเอง เนื่องจากผลการทำงานที่ลดระดับลง จึงอาจเป็นสาเหตุของคำสั่ง **rmss** ที่สามารถนำมาใช้โดยผู้ใช้รากหรือสมาชิกของกลุ่มของระบบ

คำสั่ง rmss

คุณสามารถใช้คำสั่ง **rmss** เพื่อเปลี่ยนขนาดหน่วยความจำ และจบการทำงาน หรือเป็นไดรเวอร์โปรแกรมที่ดำเนินการแอ็พพลิเคชันที่ระบุ หลายครั้งในช่วงของขนาดหน่วยความจำ และแสดงสถิติสำคัญที่อธิบาย ประสิทธิภาพของแอ็พพลิเคชันที่แต่ละขนาดหน่วยความจำ

วิธีการแรกมีประโยชน์เมื่อคุณต้องการดูและรู้สึกว่าแอ็พพลิเคชันของคุณทำงาน อย่างไรที่ขนาดของหน่วยความจำระบบที่กำหนดให้ เมื่อแอ็พพลิเคชันซับซ้อน มากเกินกว่าที่จะระบุเป็นคำสั่งเดียว หรือเมื่อคุณต้องการรันหลายอินสแตนซ์ ของแอ็พพลิเคชัน วิธีการที่สองเหมาะสมเมื่อ คุณมีแอ็พพลิเคชันที่สามารถเรียกใช้เป็นโปรแกรมที่ดำเนินการได้หรือไฟล์ shell script

แฟล็ก -c, -p และ-r ของคำสั่ง rmss:

ข้อดีของการใช้แฟล็ก **-c, -p** และ **-r** ของคำสั่ง **rmss** คือ แฟล็กเหล่านี้อนุญาตให้คุณทำการทดสอบ ด้วยแอ็พพลิเคชันที่ซับซ้อนซึ่งไม่สามารถแสดงออกด้วยโปรแกรมเรียกใช้งานเดี่ยว หรือไฟล์สคริปต์ shell หรืออีกนัยหนึ่ง ข้อเสียของการใช้อ็อพชัน **-c, -p** และ **-r** คือ อ็อพชันเหล่านั้นจะบังคับให้คุณ ทำการวัดค่าผลการทำงานด้วยตัวคุณเอง แต่โชคดีที่คุณสามารถใช้คำสั่ง **vmstat -s** เพื่อวัดค่ากิจกรรมพื้นที่การเพจที่เกิดขึ้นขณะที่แอ็พพลิเคชันของคุณรันอยู่

ด้วยการรันคำสั่ง **vmstat -s** การรันแอ็พพลิเคชันของคุณ การรันคำสั่ง **vmstat -s** อีกครั้ง และการลบจำนวนของพื้นที่การเพจเพจขาเข้าก่อน จากจำนวนของพื้นที่การเพจ เพจขาเข้าหลัง คุณสสามารถพิจารณาจำนวนวของพื้นที่การเพจ เพจขาเข้า ที่เกิดขึ้นขณะที่โปรแกรมของคุณรันอยู่ ยิ่งไปกว่านั้น ด้วยเวลาของโปรแกรมของคุณ และการแบ่งจำนวนของพื้นที่การเพจ เพจขาเข้าด้วยเวลาที่ใช้ในการรันโปรแกรม คุณสามารถขอรับพื้นที่การเพจโดยเฉลี่ย อัตราการเพจเข้า

สิ่งสำคัญคือ การรันแอ็พพลิเคชันหลายครั้งที่ขนาดหน่วยความจำแต่ละขนาด ด้วยเหตุผลสองข้อ:

- ขณะเปลี่ยนขนาดหน่วยความจำ คำสั่ง **rmss** จะล้างหน่วยความจำจำนวนมาก ดังนั้น ในครั้งแรกที่คุณรันแอ็พพลิเคชันของคุณ หลังจากการเปลี่ยนขนาดของหน่วยความจำ จึงเป็นไปได้ที่ ส่วนของเวลารัน อาจเป็นสาเหตุทำให้แอ็พพลิเคชันของคุณอ่านไฟล์ในหน่วยความจำจริง แต่ เนื่องจากไฟล์อาจคงอยู่ในหน่วยความจำหลังจากที่ยกเลิกแอ็พพลิเคชันของคุณ การประมวลผลลำดับถัดมาของแอ็พพลิเคชันของคุณอาจส่งผลในเรื่องของเวลาที่ใช้ไปแบบสั้น
- หากต้องการขอรับผลการทำงานเฉลี่ยที่เป็นอิสระของแอ็พพลิเคชันที่ขนาดหน่วยความจำ ใดๆ จึงเป็นไปไม่ได้ที่จะทำซ้ำสถานะของระบบในแต่ละครั้งที่แอ็พพลิเคชันของคุณ รัน เนื่องด้วยเหตุนี้ ผลการทำงานของแอ็พพลิเคชันของคุณสามารถผันแปรจาก การรันสู่การรัน

หากต้องการสรุป ให้พิจารณาชุดของขั้นตอนต่อไปนี้เป็นวิธีที่ต้องกานเรียกใช้คำสั่ง **rmss** :

```
while there are interesting memory sizes to investigate:
  {
  change to an interesting memory size using rmss -c;
  run the application once as a warm-up;
  for a couple of iterations:
    {
    use vmstat -s to get the "before" value of paging-space page ins;
    run the application, while timing it;
    use vmstat -s to get the "after" value of paging-space page ins;
    subtract the "before" value from the "after" value to get the
       number of page ins that occurred while the application ran;
    divide the number of paging-space page ins by the response time
       to get the paging-space page-in rate;
```

```
    }
  }
run rmss -r to restore the system to normal memory size (or reboot)
```

การคำนวณของการเพจ(หลัง – ก่อน) หมายเลข I/O สามารถทำให้เป็นอัตโนมัติ โดยการใช้สคริปต์ **vmstatit** ที่อธิบายไว้ใน “ปัญหาเกี่ยวกับดิสก์หรือหน่วยความจำ” ในหน้า 40

การเปลี่ยนขนาดหน่วยความจำ:

หากต้องการเปลี่ยนขนาดหน่วยความและออก ให้ใช้คำสั่ง **–c** flag of the **rmss**

ตัวอย่างเช่น หากต้องการเปลี่ยนขนาดหน่วยความจำไปเป็น 128 MB ให้ใช้คำสั่งดังต่อไปนี้:

```
# rmss -c 128
```

ขนาดหน่วยความจำ คือเลขจำนวนเต็มหรือจำนวนเศษทศนิยามในหน่วยเมกะไบต์ (ตัวอย่างเช่น 128.25) นอกจากนั้น ขนาดต้องอยู่ระหว่าง 8 MB และจำนวนของหน่วยความจำฟิสิคัลจริง ในเครื่องของคุณ ขึ้นอยู่กับคอนฟิกูเรชันฮาร์ดแวร์และซอฟต์แวร์ คำสั่ง **rmss** อาจไม่สามารถเปลี่ยนแปลงขนาดหน่วยความจำ ไปเป็นขนาดที่เล็กลงได้ เนื่องจากขนาดของโครงสร้างระบบที่สืบทอดกัน เช่น เคอร์เนล เมื่อคำสั่ง **rmss** ไม่สามารถเปลี่ยนแปลงไปเป็นขนาดหน่วยความจำที่กำหนด ระบบจะแสดงข้อความแสดงความผิดพลาด

คำสั่ง **rmss** จะลดขนาดหน่วยความจำที่มีประสิทธิผลของระบบได้โดยการใช้กรอบของเพจอิสระ จากรายการของกรอบอิสระที่รักษาไว้โดย VMM กรอบที่นำมาใช้ จะเก็บอยู่ในพูลของกรอบที่ไม่สามารถใช้งานได้ และจะถูกส่งคืนไปยังรายการกรอบอิสระเมื่อขนาดหน่วยความจำที่มีประสิทธิผลถูกเพิ่มขึ้น และคำสั่ง **rmss** จะปรับเปลี่ยนตัวแปรระบบบางตัวและโครงสร้างข้อมูลที่ต้องเก็บสัดส่วนไว้ให้กับขนาดหน่วยความจำที่มีประสิทธิผล

ซึ่งอาจใช้เวลาสั้นๆ (มากที่สุด 15 ถึง 20 วินาที) ขณะที่เปลี่ยนขนาดหน่วยความจำ โดยทั่วไป คุณต้องการลดขนาดหน่วยความจำลงมากเท่าใด คำสั่ง **rmss** ก็จะใช้เวลามากขึ้นในการทำให้เสร็จสิ้น เมื่อทำสำเร็จแล้ว คำสั่ง **rmss** จะตอบกลับด้วยข้อความต่อไปนี้:

```
Simulated memory size changed to  128.00 Mb.
```

หากต้องการ แสดงขนาดหน่วยความจำปัจจุบัน ให้ใช้แฟล็ก **–p** ดังต่อไปนี้:

```
# rmss -p
```

เอาต์พุต **rmss** จะเป็นดังนี้:

```
Simulated memory size is  128.00 Mb.
```

ท้ายสุด ถ้าคุณต้องการรีเซ็ตขนาดหน่วยความจำให้เป็นขนาดหน่วยความที่แท้จริงของเครื่อง ให้ใช้แฟล็ก **–r** ดังต่อไปนี้:

```
# rmss -r
```

ไม่มีปัญหาใด กับขนาดหน่วยความจำที่จำลองไว้ในปัจจุบัน การใช้แฟล็ก **–r** จะตั้งค่าหน่วยความจำให้เป็นขนาดหน่วยความจำจริงแบบฟิสิคัลของเครื่อง

เนื่องจาก ตัวอย่างนี้จะรันอยู่บนเครื่อง 256 MB คำสั่ง **rmss** จะตอบกลับดังนี้:

```
Simulated memory size changed to  256.00 Mb.
```

หมายเหตุ: คำสั่ง **rmss** จะรายงานการใช้หน่วยความจำจริง บนเครื่องที่มีหน่วยความจำที่ต่ำกว่ามาตรฐาน หรือหน่วยความจำที่ใช้งานอยู่ คำสั่ง **rmss** จะรายงานจำนวนหน่วยความจำจริง ตามจำนวนของหน่วยความจำฟิสิคัลที่ใช้จริงลบกับหน่วยความจำที่ต่ำกว่ามาตรฐาน หรือใช้งานอยู่โดยระบบ ตัวอย่างเช่น คำสั่ง **rmss -r** อาจแสดงรายงาน:

```
Simulated memory size changed to 79.9062 Mb.
```

ซึ่งสามารถเป็นผลลัพธ์ของเพจบางเพจ ที่ทำเครื่องหมายว่าใช้งานไม่ได้ หรือผลลัพธ์ของอุปกรณ์ ที่จองเพจนั้นไว้สำหรับการใช้ของตนเอง และไม่พร้อมใช้งานกับผู้ใช้อื่น

การประมวลผลแอ็พพลิเคชันผ่านช่วงของขนาดหน่วยความจำด้วยคำสั่ง rmss:

สำหรับไดร์เวอร์โปรแกรม คำสั่ง **rmss** จะเรียกใช้งานแอ็พพลิเคชันที่ระบุไว้ผ่านช่วงของขนาดหน่วยความจำ และแสดงข้อมูลสถิติ ที่อธิบายถึงผลการทำงานของแอ็พพลิเคชันที่ขนาดหน่วยความจำแต่ละขนาด

แฟล็ก **-s, -f, -d, -n** และ **-o** ของคำสั่ง **rmss** จะถูกนำมาใช้รวมกันเพื่อเรียกใช้คำสั่ง **rmss** เป็นไดร์เวอร์โปรแกรม ไวยากรณ์สำหรับลักษณะการเรียกใช้งานนี้ของคำสั่ง **rmss** เป็นดังนี้:

```
rmss [ -s smemsize ] [ -f fmemsize ] [ -d memdelta ]
     [ -n numiterations ] [ -o outputfile ] command
```

แฟล็กต่อไปนี้แต่ละตัวจะถูกกล่าวถึงในรายละเอียดดังนี้ แฟล็ก **-s, -f** และ **-d** จะถูกใช้เพื่อระบุช่วงของขนาดหน่วยความจำ

-n แฟล็กนี้จะถูกใช้เพื่อระบุจำนวนครั้งที่รัน และวัดคำสั่งที่ขนาดหน่วยความจำแต่ละขนาด

-o แฟล็กนี้จะถูกนำมาใช้เพื่อระบุไฟล์ที่เขียนลงในรายงาน **rmss** ขณะที่คำสั่งคือแอ็พพลิเคชันที่คุณต้องการรัน และวัดที่ขนาดหน่วยความจำ แต่ละขนาด

-s แฟล็กนี้จะระบุขนาดเริ่มต้น

-f แฟล็กนี้จะระบุขนาดท้ายสุด

-d แฟล็กนี้จะระบุความแตกต่างระหว่างขนาด

ค่าทั้งหมดจะอยู่ในรูปของเลขจำนวนเต็มหรือเศษทศนิยมในหน่วยเมกะไบต์ ตัวอย่างเช่น ถ้าคุณต้องการรัน และวัดค่าคำสั่งที่ขนาด 256, 224, 192, 160 และ 128 MB คุณต้องใช้การรวมกันดังนี้:

```
-s 256 -f 128 -d 32
```

เช่นเดียวกัน ถ้าคุณต้องการรันและวัดค่าคำสั่งที่ 128, 160, 192, 224 และ 256 MB คุณต้องใช้การรวมกันดังนี้:

```
-s 128 -f 256 -d 32
```

ถ้าแฟล็ก **-s** ไม่ได้ระบุไว้ คำสั่ง **rmss** จะเริ่มต้นที่ขนาดหน่วยความจำที่ใช้จริงของเครื่อง ถ้าแฟล็ก **-f** ไม่ได้ระบุไว้ คำสั่ง **rmss** จะเสร็จสิ้นที่ขนาด 8 MB ถ้าแฟล็ก **-d** ไม่ได้ระบุไว้ จะมีค่าดีฟอลต์ขนาด 8 MB ระหว่างขนาดหน่วยความจำ

ค่าที่คุณควรเลือกสำหรับแฟล็ก **-s, -f** และ **-d** คือค่าใด? ตัวเลือกปกติจะครอบคลุมขนาดหน่วยความจำของระบบ ที่จะนำมาพิจารณาเพื่อรันแอ็พพลิเคชันที่คุณกำลังวัดค่า อย่างไรก็ตาม การเพิ่มขึ้นของขนาดที่น้อยกว่า 8 MB อาจมีข้อได้เปรียบ เนื่องจากคุณสามารถขอรับการประมาณการของ จำนวนพื้นที่ที่คุณจะต้องมี เมื่อคุณตัดสินใจใช้ขนาดที่กำหนด ตัวอย่างเช่น ถ้า thrash ของแอ็พพลิเคชันที่กำหนดไว้ที่ 120 MB แต่รันโดยไม่มีเพจขาเข้าที่ 128 MB แอ็พพลิเคชันนั้นจะได้รับประโยชน์ในการทราบตำแหน่งภายในช่วงของแอ็พพลิเคชันตั้งแต่ 120 ถึง 128 MB ที่เริ่มต้นการ thrash ถ้าเริ่มต้นที่ 127 MB คุณอาจ

ต้องการพิจารณาปรับแต่งระบบด้วยหน่วยความจำที่มากกว่า 128 MB หรือคุณอาจต้องการลองแก้ไขแอ็พพลิเคชัน เพื่อให้มีพื้นที่มากขึ้น หรืออีกนัยหนึ่ง ถ้าการ thrash เริ่มต้นที่ 121 MB คุณจะทราบว่า คุณมีพื้นที่เพียงพอพร้อมกับเครื่องที่มีขนาด 128 MB

แฟล็ก **–n** จะถูกนำมาใช้เพื่อระบุจำนวนครั้งที่รัน และวัดคำสั่งที่ขนาดของหน่วยความจำแต่ละขนาด หลังจากที่รันและวัดค่าคำสั่งที่ระบุจำนวนครั้งแล้ว คำสั่ง **rmss** จะแสดงข้อมูลสถิติที่กล่าวถึงผลการทำงานโดยเฉลี่ย ของแอ็พพลิเคชันที่ขนาดของหน่วยความจำนั้น หากต้องการรันคำสั่ง 3 ครั้งที่ขนาดหน่วยความจำแต่ละขนาด คุณควรใช้คำสั่งต่อไปนี้:

```
-n 3
```

ถ้าไม่ระบุแฟล็ก **–n** คำสั่ง **rmss** จะพิจารณาในระหว่างการกำหนดค่าเริ่มต้นให้กับจำนวนแอ็พพลิเคชันของคุณที่ต้องรัน เพื่อสะสมจำนวนเวลาที่รันทั้งหมดใน 10 วินาที คำสั่ง **rmss** จำทำสิ่งนี้เพื่อมั่นใจว่า ข้อมูลสถิติเกี่ยวกับผลการทำงานสำหรับโปรแกรมที่รันในระยะเวลาที่สั้น จะไม่ตรงกันส่วนที่อิทธิพลภายนอก เช่น daemon

หมายเหตุ: ถ้าคุณกำลังวัดค่าโปรแกรมที่สั้นมาก จำนวนของการวนซ้ำอาจจำเป็นต้องมี เพื่อสะสมเวลา 10 วินาทีของเวลา CPU ที่อาจมีขนาดใหญ่มาก เนื่องจากการประมวลผลโปรแกรมแต่ละโปรแกรม จะใช้เวลาประมาณ 2 วินาทีของการใช้ **rmss** ให้ระบุพารามิเตอร์ **–n** อย่างชัดเจนสำหรับโปรแกรมที่มีขนาดสั้น

ค่าที่ดีที่สุดที่ควรจะนำมาใช้สำหรับแฟล็ก **–n** คือค่าใด? ถ้าคุณทราบว่า แอ็พพลิเคชันของคุณใช้เวลามากกว่า 10 วินาทีในการรัน คุณสามารถระบุ **–n 1** เพื่อให้คำสั่งรันสองครั้ง แต่จะถูกวัดค่าเพียงครั้งเดียวในแต่ละขนาดของหน่วยความจำ ข้อได้เปรียบสำหรับการใช้แฟล็ก **–n** คือ คำสั่ง **rmss** จะเสร็จสิ้นเร็วขึ้น เนื่องจากคำสั่งจะไม่เสียเวลาในการกำหนดค่าเริ่มต้น เพื่อพิจารณาจำนวนครั้งที่รันโปรแกรมของคุณ การทำเช่นนี้จะมีค่ามาก เมื่อคำสั่งที่ต้องการวัดค่ามีการรันที่นานและเป็นแบบโต้ตอบ

สิ่งสำคัญคือ การจดบันทึกว่า คำสั่ง **rmss** จะรันคำสั่งหนึ่งครั้งในแต่ละขนาดหน่วยความจำที่ทดลองใช้ก่อนที่จะรัน และวัดค่าคำสั่ง การทดสอบการใช้จำเป็นต้องทำเพื่อหลีกเลี่ยง I/O ที่เกิดขึ้น เมื่อแอ็พพลิเคชันยังไม่ได้อยู่ในหน่วยความจำ แม้ว่า I/O บางตัวจะส่งผลต่อผลการทำงาน จึงไม่มีความจำเป็นเนื่องจากขาดแคลนหน่วยความจำที่ใช้จริง การรันการทดสอบการใช้จะไม่ถูกสอดแทรกในจำนวนของการวนซ้ำที่ระบุด้วยแฟล็ก **–n**

แฟล็ก **–o** จะถูกใช้เพื่อระบุไฟล์เพื่อเขียนรายงาน **rmss** ถ้าไม่ได้ระบุแฟล็ก **–o** รายงานจะถูกเขียนลงในไฟล์ `rmss.out`

ท้ายสุด **command** จะถูกใช้เพื่อระบุแอ็พพลิเคชัน ที่ต้องการวัดค่า ซึ่งสามารถเป็นโปรแกรมเรียกทำงานหรือสคริปต์ shell พร้อมด้วยหรือไม่มีอาร์กิวเมนต์บรรทัดรับคำสั่ง อย่างไรก็ตาม มีข้อจำกัดบางอย่างเกี่ยวกับรูปแบบของคำสั่ง อันดับแรก ไม่สามารถมีการเปลี่ยนทิศทางของอินพุตหรือเอาต์พุต (ตัวอย่างเช่น **foo > output** หรือ **foo < input**) นี่เป็นเพราะ คำสั่ง **rmss** จะปฏิบัติทุกสิ่งที่อยู่ด้านขวาของชื่อคำสั่ง ซึ่งเป็นอาร์กิวเมนต์ของคำสั่ง หากต้องการเปลี่ยนทิศทาง ให้วางคำสั่งลงในไฟล์สคริปต์ shell

โดยปกติ ถ้าคุณต้องการเก็บเอาต์พุต **rmss** ลงในไฟล์ที่ระบุเฉพาะ ให้ใช้อ็อพชัน **–o** ถ้าคุณต้องการเปลี่ยนทิศทางเอาต์พุตมาตรฐานของคำสั่ง **rmss** (ตัวอย่างเช่น เพื่อให้ความสนใจไปที่ส่วนท้ายของไฟล์ที่มีอยู่) ให้ใช้ Korn shell เพื่อปิดการเรียกใช้งาน **rmss** ในเครื่องหมายวงเล็บ ดังต่อไปนี้:

```
# (rmss -s 24 -f 8 foo) >> output
```

คำสั่ง rmss เป็นผลทำให้เกิดการตีความ

คำสั่ง **rmss** จะสร้างข้อมูลที่มีค่า

ตัวอย่างในหัวข้อ “รายงานที่สร้างขึ้นสำหรับโปรแกรม foo”จะถูกสร้างโดยการรันคำสั่ง **rmss** บนแอ็พพลิเคชันโปรแกรมจริง แม้ว่าชื่อของโปรแกรมได้ถูกเปลี่ยนเป็น `foo` สำหรับความไม่มีลักษณะเฉพาะ คำสั่งเฉพาะเพื่อสร้างรายงานเป็นดังนี้ :

```
# rmss -s 16 -f 8 -d 1 -n 1 -o rmss.out foo
```

รายงานที่สร้างขึ้นสำหรับโปรแกรม foo:

คำสั่ง rmss จัดทำรายงาน สำหรับโปรแกรม foo

```
Hostname:  aixhost1.austin.ibm.com
Real memory size:   16.00 Mb
Time of day:  Thu Mar 18 19:04:04 2004
Command:  foo

Simulated memory size initialized to  16.00 Mb.

Number of iterations per memory size = 1 warm-up + 1 measured = 2.

Memory size  Avg. Pageins  Avg. Response Time    Avg. Pagein Rate
(megabytes)                (sec.)                (pageins / sec.)
---------------------------------------------------------------
16.00           115.0           123.9                 0.9
15.00           112.0           125.1                 0.9
14.00           179.0           126.2                 1.4
13.00            81.0           125.7                 0.6
12.00           403.0           132.0                 3.1
11.00           855.0           141.5                 6.0
10.00          1161.0           146.8                 7.9
9.00           1529.0           161.3                 9.5
8.00           2931.0           202.5                14.5
```

รายงาน ประกอบด้วยสี่คอลัมน์ คอลัมน์ด้านซ้ายสุดแสดงขนาดหน่วยความจำ ในขณะที่คอลัมน์ *Avg. Pageins* แสดงจำนวนเฉลี่ยของ page-ins ที่เกิดขึ้นเมื่อแอ็พพลิเคชันรันที่ขนาดหน่วยความจำนั้น ข้อสำคัญคือ ควรทราบว่าคอลัมน์ *Avg. Pageins* แสดงการดำเนินงาน page-in ทั้งหมด รวมถึงรหัส ข้อมูล และการอ่านไฟล์ จากโปรแกรมทั้งหมดที่เสร็จสมบูรณ์ตั้งแต่ แอ็พพลิเคชันรัน คอลัมน์ *Avg. Response Time* แสดงจำนวนครั้งเฉลี่ย ที่ใช้ในการทำให้แอ็พพลิเคชันเสร็จสมบูรณ์ ในขณะที่คอลัมน์ *Avg. Pagein Rate* ให้อัตราเฉลี่ยของ page-ins

ควรให้ความสำคัญกับคอลัมน์ *Avg. Pagein Rate* ตั้งแต่ 16 MB ถึง 13 MB อัตรา page-in ค่อนข้าง ต่ำ (< 1.5 page-ins ต่อวินาที) อย่างไรก็ตาม ตั้งแต่ 13 MB ถึง 8 MB อัตรา page-in เพิ่มขึ้นอย่างช้าๆ ในช่วงแรก แล้วเพิ่มอย่างรวดเร็วเมื่อขึ้นถึง 8 MB คอลัมน์ *Avg. Response Time* มีรูปทรงที่คล้ายกันคือ ค่อนข้างแบนราบในช่วงแรก แล้ว เพิ่มขึ้นอย่างช้าๆ ในตอนท้ายเพิ่มขึ้นอย่างรวดเร็วเมื่อขนาดหน่วยความจำ ลดลงเป็น 8 MB

ที่นี่ อัตรา page-in ลดลงอย่างแท้จริงเมื่อขนาดหน่วยความจำเปลี่ยน จาก 14 MB (1.4 page-ins ต่อวินาที) เป็น 13 MB (0.6 page-ins ต่อวินาที) นี่เป็นสาเหตุของการแจ้งเตือน ในระบบจริง อาจไม่สามารถ คาดได้ว่าผลลัพธ์จะราบรื่นโดยสมบูรณ์ จุดสำคัญคืออัตรา page-in ค่อนข้างต่ำที่ทั้ง 14 MB และ 13 MB

สุดท้าย คุณสามารถได้ข้อสรุปสองสามข้อจากรายงาน ข้อแรก ถ้าประสิทธิภาพของ แอ็พพลิเคชันดูเหมือนไม่สามารถยอมรับได้ที่ 8 MB (อาจเกิดขึ้นได้) การเพิ่มหน่วยความจำอาจพัฒนาประสิทธิภาพได้เป็นอย่างมาก หมายเหตุว่า เวลาตอบกลับเพิ่ม

ขึ้นจากประมาณ 124 วินาทีที่ 16 MB เป็น 202 วินาที ที่ 8 MB เพิ่มขึ้น 63 เปอร์เซ็นต์ ในทางกลับกัน ถ้าประสิทธิภาพดูเหมือนไม่สามารถยอมรับได้ที่ 16 MB การเพิ่มหน่วยความจำจะไม่ช่วยพัฒนาประสิทธิภาพ มากนัก เนื่องจาก page-ins ไม่ทำให้โปรแกรมช้าลงที่ 16 MB

รายงานสำหรับสำเนาแบบรีโมต 16 MB:

ตัวอย่างต่อไปนี้แสดงรายงานที่สร้างขึ้น (บนเครื่องไคลเอ็นต์) โดยการรันคำสั่ง **rmss** บนคำสั่งที่ คัดลอกไฟล์ 16 MB จากเครื่องรีโมต (เซิร์ฟเวอร์) ผ่านทาง NFS

```
Hostname:  aixhost2.austin.ibm.com
Real memory size:   48.00 Mb
Time of day:  Mon Mar 22 18:16:42 2004
Command:  cp /mnt/a16Mfile /dev/null

Simulated memory size initialized to  48.00 Mb.

Number of iterations per memory size = 1 warm-up + 4 measured = 5.

Memory size   Avg. Pageins   Avg. Response Time  Avg. Pagein Rate
(megabytes)                     (sec.)          (pageins / sec.)
--------------------------------------------------------------------
48.00             0.0            2.7                  0.0
40.00             0.0            2.7                  0.0
32.00             0.0            2.7                  0.0
24.00             1520.8        26.9                 56.6
16.00             4104.2        67.5                 60.8
8.00              4106.8        66.9                 61.4
```

เวลาตอบกลับและอัตรา page-in ในรายงานนี้เริ่มต้นค่อนข้างต่ำ แล้วเพิ่มขึ้นอย่างรวดเร็วที่ขนาดหน่วยความจำ 24 MB จากนั้น เพิ่มสูงสุดที่ 16 และ 8 MB รายงานนี้แสดงความสำคัญของการเลือกช่วงกว้างของขนาดหน่วยความจำ เมื่อคุณใช้คำสั่ง **rmss** ถ้าผู้ใช้รายนี้ ดูเฉพาะขนาดหน่วยความจำตั้งแต่ 24 MB ถึง 8 MB ผู้ใช้อาจพลาดโอกาส การตั้งค่าคอนฟิกระบบด้วยหน่วยความจำที่เพียงพอจะสนับสนุน แอ็พพลิเคชันโดยไม่มี page-ins

คำแนะนำสำหรับการใช้แฟล็ก -s, -f, -d, -n และ -o:

คุณลักษณะหนึ่งที่มีประโยชน์ของคำสั่ง **rmss** เมื่อใช้ในวิธีนี้นั่นคือ คำสั่งสามารถถูกยกเลิกได้ด้วยคีย์อินเตอร์รัปต์ (**Ctrl + C** ตามค่าดีฟอลต์) โดยไม่ได้ทำลายรายงานที่ได้เขียนลงใน ไฟล์เอาต์พุตแล้ว นอกจากการเขียนรายงานลงในไฟล์เอาต์พุต การทำเช่นนี้อาจเป็นสาเหตุทำให้คำสั่ง **rmss** รีเซ็ตขนาดหน่วยความจำให้เป็นขนาดหน่วยความจำฟิสิคัลของเครื่อง

คุณสามารถรันคำสั่ง **rmss** ในส่วนหลัง หลังจากที่คุณได้ล็อกเอาต์โดยใช้คำสั่ง **nohup** หากต้องการทำเช่นนี้ ให้เรียกใช้คำสั่ง **rmss** ก่อนคำสั่ง **nohup** และตามหลังคำสั่งทั้งหมดด้วย & (เครื่องหมายแอมเปอร์แซนด์) ดังนี้:

```
# nohup rmss -s 48 -f 8 -o foo.out foo &
```

แนวทางในการพิจารณาเมื่อใช้คำสั่ง rmss

เนื่องจากไม่มีปัญหาในเรื่องลักษณะการเรียก **rmss** ที่คุณกำลังใช้ ดังนั้นจึงเป็นสิ่งสำคัญในการสร้างสภาวะแวดล้อมให้ใกล้เคียงกับผู้ใช้ขั้นปลายใหม่อีกครั้ง เท่าที่จะเป็นไปได้

ตัวอย่างเช่น คุณกำลังใช้แบบจำลอง CPU ที่เหมือนกัน แบบจำลองดิสก์ที่เหมือนกัน เน็ตเวิร์กที่เหมือนกันหรือไม่? ผู้ใช้มีแอ็พพลิเคชันไฟล์ที่ประกอบเข้าจากโหนดแบบรีโมต ผ่าน NFS หรือระบบไฟล์ที่กระจายอื่นๆ หรือไม่? จุดสุดท้ายนี้เป็นสิ่งสำคัญ เนื่องจากเพจจากรีโมตไฟล์จะถูกใช้โดย VMM ที่แตกต่างจากเพจจากโลคัลไฟล์

ยิ่งไปกว่านั้น สิ่งที่ดีที่สุดในการกำจัดกิจกรรมของระบบใดๆ ที่ไม่เกี่ยวข้องกับคอนฟิกูเรชันระบบที่ต้องการ หรือแอ็พพลิเคชันที่คุณกำลังวัดค่า ตัวอย่างเช่น คุณไม่ต้องการให้บุคคลต่างๆ ทำงานอยู่บนเครื่องเดียวกันตามคำสั่ง **rmss** เว้นเสียแต่บุคคลเหล่านั้นจะทำงานกับส่วนของเวิร์กโหลด ที่คุณกำลังวัดค่า

หมายเหตุ: คุณไม่สามารถรันการเรียกใช้คำสั่ง **rmss** จำนวนมากแบบพร้อมเพียงกันได้

เมื่อคุณเสร็จสิ้นการรันคำสั่ง **rmss** ทั้งหมดแล้ว ให้ปิดระบบและรีบูตระบบของคุณ การดำเนินการนี้จะลบการเปลี่ยนแปลงทั้งหมดที่คำสั่ง **rmss** ได้ทำไว้กับระบบ และจะเรียกคืนพารามิเตอร์การควบคุมโหลดหน่วยความจำ VMM ให้กับค่าติดตั้งตามปกติ

การปรับการควบคุมโหลดหน่วยความจำ VMM ด้วยคำสั่ง schedo

ด้วยคำสั่ง **schedo** ผู้ใช้รากสามารถเปลี่ยนเกณฑ์ที่ใช้เพื่อ กำหนด thrashing, เกณฑ์ที่ใช้เพื่อกำหนดกระบวนการที่จะพักไว้ ช่วงเวลาที่จะรอหลังจาก thrashing สิ้นสุดลงก่อน การเรียกใช้กระบวนการใหม่ จำนวนต่ำสุดของกระบวนการที่ได้รับยกเว้นจากการพักไว้ หรือรีเซ็ตค่าเป็นค่าดีฟอลต์

ฟังก์ชันการควบคุมโหลดหน่วยความจำ VMM ที่อธิบายใน “ฟังก์ชันการควบคุมโหลดหน่วยความจำ VMM” ในหน้า 56 ป้องกันไม่ให้ระบบที่โอเวอร์โหลด thrashing

สำหรับเวอร์ชันช่วงแรกของระบบปฏิบัติการ ถ้ากระบวนการจำนวนมาก เข้าถึงระบบพร้อมกัน หน่วยความจำจะ overcommitted และเกิด thrashing ซึ่งส่งผลให้ประสิทธิภาพด้อยลงมาก กลไกการควบคุมโหลดหน่วยความจำ ซึ่งสามารถตรวจพบ thrashing มีการพัฒนาขึ้น พารามิเตอร์บางตัวมีผลกระทบต่อฟังก์ชัน ของกลไกการควบคุมโหลด

เมื่อต้องการกำหนดว่าคำสั่ง **schedo** มีการติดตั้งและมีอยู่ หรือไม่ ให้รันคำสั่งต่อไปนี้:

```
# lslpp -lI bos.perf.tune
```

การปรับการควบคุมการโหลดหน่วยความจำ

การควบคุมการโหลดหน่วยความจำมีแนวโน้วในทางเรียบงาน ไม่ใช้งานด้วยความต้องการที่สูง ในโหลดที่อาจเป็นสาเหตุทำให้ระบบเกิด thrash

การควบคุมการโหลดหน่วยความจำจะแลกเปลี่ยนมัลติโปรแกรมมิงสำหรับทรูพุต และไม่ต้องการ ทำหน้าที่อย่างต่อเนื่องในคอนฟิกูเรชันที่มี RAM น้อยเกินไป ในการจัดการเวิร์กโหลดปกติ การออกแบบทำขึ้นสำหรับงานแบ็ตช์ และไม่ต้องแบ่งแยกมากนัก AIX Workload Manager จัดเตรียมโซลูชันที่ดีกว่าในการป้องกันภารกิจที่สำคัญ

โซลูชันที่ถูกต้องตามหลักการ การขาดแคลน RAM ที่ยังคงมีอยู่คือ การเพิ่ม RAM ไม่ต้องมีการทดสอบการควบคุมหน่วยความจำในความพยายามในการแลกเปลี่ยน เวลาตอบสนองสำหรับหน่วยความจำ สถานการณ์ในตัวช่วยการควบคุมการโหลดหน่วยความจำ อาจจำเป็นต้องถูกปรับ เนื่องจากมี RAM มากขึ้น ซึ่งไม่น้อยกว่าค่าดีฟอลต์ที่เลือก ตัวอย่างของคอนฟิกูเรชันซึ่งค่าดีฟอลต์จะถูกสงวนไว้

คุณไม่ความเปลี่ยนค่าติดตั้งพารามิเตอร์การควบคุมการโหลด จนกว่าเวิร์กโหลดของคุณสอดคล้องกัน และคุณเชื่อว่า พารามิเตอร์ดีฟอลต์ไม่เหมาะกับ เวิร์กโหลดของคุณ

ค่าติดตั้งพารามิเตอร์ดีฟอลต์จะจัดส่งมาพร้อมกับระบบจะบังคับไม่ให้เปลี่ยนแปลง ค่าดีฟอลต์ของพารามิเตอร์เหล่านี้ควรเลือก "ล้มเหลวอย่างปลอดภัย" ระหว่างช่วงความกว้างของเวิร์กโหลด พารามิเตอร์ที่เปลี่ยนแปลงครั้งสุดท้าย จนกว่าจะถึงการบูตระบบในครั้งถัดไป กิจกรรมการปรับการควบคุมโหมดหน่วยความจำ ต้องถูกทำโดยผู้ใช้ราก ผู้ดูแลระบบสามารถใช้คำสั่ง **schedo** เพื่อเปลี่ยนพารามิเตอร์ในการปรับอัลกอริธึมให้กับเวิร์กโหลดโดยเฉพาะ หรือปิดใช้งานทั้งหมด

ตัวอย่างต่อไปนี้จะแสดงค่าพารามิเตอร์ปัจจุบันด้วยคำสั่ง **schedo**:

```
# schedo -a
           v_repage_hi = 0
         v_repage_proc = 4
            v_sec_wait = 1
         v_min_process = 2
         v_exempt_secs = 2
              pacefork = 10
               sched_D = 16
               sched_R = 16
             timeslice = 1
               maxspin = 1
              %usDelta = 100
          affinity_lim = n/a
idle_migration_barrier = n/a
      fixed_pri_global = n/a
         big_tick_size = 1
             force_grq = n/a
```

ห้าพารามิเตอร์แรกระบุ threshold สำหรับอัลกอริธึมการควบคุมการโหลดของหน่วยความจำ พารามิเตอร์เหล่านี้จะตั้งค่าอัตราและ threshold สำหรับอัลกอริธึม ถ้าอัลกอริธึม แสดง RAM ที่ overcommit ค่า *v_repage_proc*, *v_min_process*, *v_sec_wait* และ *v_exempt_secs* จะถูกจำมาใช้ ไม่เช่นนั้น ค่าเหล่านั้นจะถูกละเว้น ถ้าการควบคุมโหลดของหน่วยความจำถูกปิดใช้งาน ค่าเหล่านี้จะไม่ถูกนำมาใช้

หลังจากที่ทดสอบการปรับแล้ว การควบคุมโหลดของหน่วยความจำสามารถรีเซ็ตไปเป็นคุณสมบัติดีฟอลต์ โดยเรียกใช้งานคำสั่ง **schedo -D**

พารามิเตอร์ v_repage_hi:

พารามิเตอร์ *v_repage_hi* ควบคุม threshold ที่กำหนด overcommitment หน่วยความจำ การควบคุมโหลดหน่วยความจำพยายามจะพัก กระบวนการเมื่อเกินกว่าค่า threshold นี้ในระหว่างช่วงหนึ่งวินาทีใดๆ

Threshold คือความสัมพันธ์ระหว่างการประเมินโดยตรงสองรายการ: จำนวนของ หน้าที่บันทึกลงในพื้นที่ว่างการเพจในวินาทีล่าสุด (`po`) และ จำนวนของการขโมยหน้าที่เกิดขึ้นในวินาทีล่าสุด (`fr`) คุณสามารถ ดูทั้งสองค่านี้ในเอาต์พุต **vmstat** จำนวนของการบันทึกหน้า โดยปกติแล้ว น้อยกว่าจำนวนของการขโมยหน้าอย่างมาก หน่วยความจำ ถูก overcommitted เมื่อเงื่อนไขต่อไปนี้เป็นจริง:

```
po/fr > 1/v_repage_hi or po*v_repage_hi > fr
```

คำสั่ง **schedo -o v_repage_hi=0** ปิดใช้งานการควบคุม โหลดหน่วยความจำได้อย่างมีประสิทธิภาพ ถ้าระบบมีหน่วยความจำอย่างน้อย 128 MB ค่าดีฟอลต์คือ 0 มิฉะนั้นค่าดีฟอลต์จะเป็น 6 โดยปกติแล้ว ด้วย RAM อย่างน้อย 128 MB ขั้นตอนวิธี VMM ปกติสามารถแก้ไขสภาพ thrashing ได้โดยเฉลี่ยมีประสิทธิภาพ มากกว่าการใช้การควบคุมโหลดหน่วยความจำ

ในสถานการณ์พิเศษบางอย่าง อาจเหมาะสมที่จะปิดใช้งานการควบคุมโหลด หน่วยความจำจาก outset ตัวอย่างเช่น ถ้าคุณกำลังใช้ terminal emulator ที่มีคุณลักษณะไทม์เอาต์เพื่อจำลองเวิร์กโหลดหลายผู้ใช้ การเข้าแทรกแซงของ การควบคุมโหลดหน่วยความจำอาจส่งผลให้การตอบกลับบางรายการล่าช้านานจนทำ ให้กระบวนการถูก killed โดยคุณลักษณะไทม์เอาต์ อีกตัวอย่างหนึ่งคือ ถ้าคุณกำลังใช้คำสั่ง **rmss** เพื่อสืบสวนผลกระทบของขนาดหน่วยความจำที่ลดลง ควรปิดใช้งานการควบคุมโหลดหน่วยความจำเพื่อหลีกเลี่ยงการเข้าแทรกแซง การประเมินของคุณ

ถ้าการปิดใช้งานการควบคุมโหลดหน่วยความจำส่งผลให้เกิดสถานการณ์ thrashing มากขึ้นแทนที่จะน้อยลง (พร้อมการตอบกลับที่แย่ลงตามมา) แสดงว่า การควบคุมโหลดหน่วยความจำมีบทบาทสนับสนุนในระบบของคุณในปัจจุบัน การปรับพารามิเตอร์ การควบคุมโหลดหน่วยความจำอาจส่งผลให้ประสิทธิภาพพัฒนาขึ้นหรือ คุณอาจต้องเพิ่ม RAM

ค่าที่ต่ำกว่าของ *v_repage_hi* เพิ่ม threshold การตรวจพบ thrashing นั่นคือระบบได้รับอนุญาตให้เข้าใกล้ thrashing มากขึ้นก่อนกระบวนการจะถูกพักไว้ โดยไม่คำนึงถึงการตั้งค่าคอนฟิกระบบ เมื่อเศษส่วน `po/fr` ข้างบนต่ำ thrashing ไม่ ค่อยเกิดขึ้น

เมื่อต้องการเปลี่ยน threshold เป็น 4 ให้ป้อนข้อมูลต่อไปนี้:

```
# schedo -o v_repage_hi=4
```

ในวิธีนี้ คุณอนุญาตให้ระบบเข้าใกล้ thrashing มากขึ้นก่อนขั้นตอนวิธี จะเริ่มต้นกระบวนการพักไว้

พารามิเตอร์ v_repage_proc:

พารามิเตอร์ *v_repage_proc* กำหนดว่ากระบวนการมีสิทธิ ถูกพักไว้หรือไม่ และใช้เพื่อตั้งค่า threshold สำหรับอัตราส่วนของการประเมินสองรายการที่มีการจัดทำสำหรับทุกกระบวนการ: จำนวนของ repages (*r*) และจำนวนของ page faults ที่กระบวนการสะสมไว้ในวินาทีล่าสุด (*f*)

อัตราส่วนสูงของ repages ต่อ page faults หมายความว่าแต่ละกระบวนการ thrashing กระบวนการมีการพิจารณาว่ามีสิทธิสำหรับการพักไว้ (คือ thrashing หรือมีส่วนใน thrashing โดยรวม) เมื่อเงื่อนไขต่อไปนี้เป็นจริง:

```
r/f > 1/v_repage_proc or r*v_repage_proc > f
```

ค่าดีฟอลต์ของ *v_repage_proc* คือ 4 ซึ่งหมายความว่า กระบวนการถูกพิจารณาว่าเป็น thrashing (และตัวเลือกสำหรับการพักไว้) เมื่อเศษส่วนของ repages ต่อ page faults ในวินาทีล่าสุดมากกว่า 25 เปอร์เซ็นต์ ค่าต่ำของ *v_repage_proc* ส่งผลให้ ระดับของ thrashing แต่ละกระบวนการที่จะได้รับอนุญาตก่อนกระบวนการ มีสิทธิพักไว้สูงขึ้น

เมื่อต้องการปิดทางไม่ให้กระบวนการถูกพักไว้โดยการควบคุมโหลดหน่วยความจำ ให้ทำดังต่อไปนี้:

```
# schedo -o v_repage_proc=0
```

หมายเหตุว่ากระบวนการที่มีระดับความสำคัญคงที่และกระบวนเคอร์นัลได้รับยกเว้น จากการพักไว้

พารามิเตอร์ v_min_process:

พารามิเตอร์ *v_min_process* กำหนดขีดจำกัดต่ำสุด สำหรับระดับของ multiprogramming ซึ่งมีการกำหนดไว้เป็นจำนวนของกระบวนการที่ใช้งานอยู่ กระบวนการที่ใช้งานอยู่คือกระบวนการที่สามารถรันและกำลังรอ page I/O กระบวนการที่กำลังรอเหตุการณ์และกระบวนการที่พักไว้ไม่ถือว่า เป็นกระบวนการที่ใช้งานอยู่เช่นเดียวกับกระบวนการรอ

ด้วยการตั้งค่าระดับ multiprogramming ต่ำสุด พารามิเตอร์ *v_min_process* สามารถช่วยให้ *v_min_process* กระบวนการไม่ต้องถูกพักไว้ได้ อย่างมีประสิทธิภาพ สมมุติว่าผู้ดูแลระบบทราบว่ามีอยู่อย่างน้อยสิบกระบวนการเสมอ และใช้งานอยู่ใน

RAM เพื่อประสิทธิภาพที่ดีขึ้น และสงสัยว่าการควบคุมโหลดหน่วยความจำพักกระบวนการมากเกินไป ถ้าออกใช้คำสั่ง **schedo -o v_min_process=10** ระบบจะไม่มีทางพักกระบวนการมากจนทำให้มีกระบวนการที่ช่วงชิงหน่วยความจำกัน เหลืออยู่น้อยกว่าสิบกระบวนการ พารามิเตอร์ *v_min_process* ไม่ได้นับ:

- กระบวนการเคอร์นัล
- กระบวนการที่มีการ pinned ใน RAM ด้วยการเรียกระบบ **plock( )**
- กระบวนการที่มีระดับความสำคัญคงที่ซึ่งมีค่าระดับความสำคัญน้อยกว่า 60
- กระบวนการที่กำลังรอเหตุการณ์

ค่าดีฟอลต์ระบบของ **v_min_process=2** ทำให้มั่นใจว่าเคอร์นัล ตัวประมวลผลที่ pinned ทั้งหมด และสองกระบวนการผู้ใช้จะมีอยู่ในชุดของ กระบวนการที่กำลังช่วงชิง RAM เสมอ

ในขณะที่ **v_min_process=2** เหมาะสมสำหรับเดสก์ท็อป ซึ่งเป็นการตั้งค่า คอนฟิกผู้ใช้รายเดียว บ่อยครั้งที่ค่านี้น้อยเกินไปสำหรับการตั้งค่าคอนฟิกเครื่องที่ใหญ่ขึ้น มีผู้ใช้หลายราย หรือเซิร์ฟเวอร์ที่มี RAM จำนวนมาก

ถ้าระบบที่คุณกำลังติดตั้งมีขนาดใหญ่กว่า 32 MB แต่น้อยกว่า 128 MB และคาดว่าจะใช้สนับสนุนผู้ใช้ที่ใช้งานอยู่มากกว่าห้ารายพร้อมกัน ให้พิจารณา การเพิ่มระดับต่ำสุดของ multiprogramming ของกลไกการควบคุมโหลด หน่วยความจำ VMM

ดังตัวอย่าง ถ้าคุณประมาณเผื่อไว้ว่าแอ็พพลิเคชันที่เน้นหน่วยความจำมากที่สุด สี่รายการควรจะสามารถรันพร้อมกันได้โดยจัดหน่วยความจำไว้อย่างน้อย 16 MB สำหรับระบบปฏิบัติการและ 25 เปอร์เซ็นต์ของหน่วยความจำจริงสำหรับ หน้าไฟล์ คุณควรเพิ่มระดับต่ำสุดของ multiprogramming จากค่าดีฟอลต์ 2 เป็น 4 ด้วยคำสั่งต่อไปนี้:

```
# schedo -o v_min_process=4
```

บนระบบเหล่านี้ การตั้งค่าพารามิเตอร์ *v_min_process* เป็น 4 หรือ 6 อาจทำให้ได้ประสิทธิภาพที่ดีที่สุด ค่าที่ต่ำกว่าของ *v_min_process* ซึ่งใช้ได้ หมายความว่ามีกระบวนการผู้ใช้ที่อาจใช้งานอยู่ในขณะหนึ่งได้น้อยลง

ถ้าทราบความต้องการหน่วยความจำของแอ็พพลิเคชันที่ thrashing จะสามารถเลือกค่า *v_min_process* ได้อย่างเหมาะสม สมมุติว่า thrashing เกิดจากอินสแตนซ์จำนวนมากของแอ็พพลิเคชัน หนึ่งที่มีขนาด *M* กำหนดให้ขนาดหน่วยความจำระบบเป็น *N* ควรมีการตั้งค่าพารามิเตอร์ *v_min_process* เป็นค่าที่ใกล้เคียงกับ *N/M* การตั้งค่า *v_min_process* ต่ำเกินไปอาจจำกัดจำนวนของกระบวน การที่สามารถใช้งานอยู่ในเวลาเดียวกันโดยไม่จำเป็น

พารามิเตอร์ v_sec_wait:

พารามิเตอร์ *v_sec_wait* ควบคุมจำนวนของช่วงเวลา หนึ่งวินาทีในระหว่างที่เศษส่วน `po/fr` ต้องยังคงต่ำกว่า `1/v_repage_hi` ก่อนจะเรียกใช้กระบวนการที่พักไว้ใหม่

ค่าดีฟอลต์ของหนึ่งวินาทีใกล้กับค่าต่ำสุดที่ใช้ได้ ซึ่งเป็นศูนย์ ค่าของหนึ่งวินาทีพยายามเรียกใช้กระบวนการใหม่อย่างเต็มที่ในทันทีที่รอบเวลาปลอดภัยของหนึ่งวินาทีเกิดขึ้น ค่ามากของ *v_sec_wait* มีความเสี่ยง ของเวลาตอบกลับที่ไม่ดีโดยไม่จำเป็นสำหรับกระบวนการที่พักไว้ในขณะที่ ตัวประมวลผล idle เนื่องจากขาดตัวประมวลผลที่ใช้งานอยู่ที่จะรัน

เมื่อต้องการเปลี่ยนเวลารอที่จะเรียกใช้กระบวนการใหม่นานกว่าสองวินาที ให้ป้อนข้อมูลต่อไปนี้:

```
# schedo -o v_sec_wait=2
```

พารามิเตอร์ v_exempt_secs:

ในแต่ละครั้งที่เรียกใช้กระบวนการที่พักไว้ใหม่ กระบวนการนั้นจะออกจาก สภาพการพักไว้หลังเวลาผ่านไปแล้ว *v_exempt_secs* วินาที ที่เป็นเช่นนี้เพื่อให้แน่ใจว่าต้นทุนสูงในดิสก์ I/O ของการเพจในหน้าของกระบวน การที่พักไว้ ส่งผลให้เกิดโอกาสของความคืบหน้าที่คุ้มค่า

ค่าดีฟอลต์ของ *v_exempt_secs* คือ 2 วินาที

เมื่อต้องการเปลี่ยนพารามิเตอร์นี้ ให้ป้อนข้อมูลดังต่อไปนี้:

```
# schedo -o v_exempt_secs=1
```

สมมุติว่า thrashing เกิดขึ้นเป็นครั้งคราวโดยแอ็พพลิเคชันที่ใช้หน่วยความจำ จำนวนมากแต่รันประมาณ *T* วินาที ค่าติดตั้งระบบดีฟอลต์ นาน 2 วินาทีสำหรับพารามิเตอร์ *v_exempt_secs* อาจส่งผลให้แอ็พพลิเคชันนี้ swapping in และ out *T*/2 ครั้งบนระบบที่ยุ่ง ในกรณีนี้ การรีเซ็ตพารามิเตอร์ *v_exempt_secs* เป็นเวลาที่นานขึ้นช่วยให้แอ็พพลิเคชันนี้พัฒนาขึ้น ประสิทธิภาพระบบจะดีขึ้น เมื่อผลักดันแอ็พพลิเคชันประเภทนี้ออกไปโดยเร็ว

การปรับการเปลี่ยนหน้า VMM

ขั้นตอนวิธีการจัดการหน่วยความจำพยายามรักษาขนาดของรายการที่ว่าง และเปอร์เซ็นต์ของหน่วยความจำจริงที่ใช้โดยหน้าเซกเมนต์ถาวร ภายในขอบเขตที่ระบุ

ขอบเขตเหล่านี้ที่อธิบายใน “การจัดการหน่วยความจำจริง” ในหน้า 51 สามารถเปลี่ยนได้ด้วยคำสั่ง **vmo** ซึ่งสามารถรันโดยผู้ใช้รากเท่านั้น การเปลี่ยนแปลงที่ทำโดยเครื่องมือนี้ยังคงมีผลจนกว่า รีบูตระบบครั้งถัดไป เมื่อต้องการกำหนดว่าคำสั่ง **vmo** มีการติดตั้งและมีอยู่ หรือไม่ ให้รันคำสั่งต่อไปนี้:

```
# lslpp -lI bos.perf.tune
```

การใช้คำสั่ง **vmo** ที่มีอ็อพชัน **–a** แสดงค่าติดตั้งพารามิเตอร์ปัจจุบัน

หมายเหตุ: คำสั่ง **vmo** เป็นคำสั่งที่อธิบายตัวเอง คุณอาจได้รับเอาต์พุตที่ต่างจาก เอาต์พุตตัวอย่างต่อไปนี้

```
# vmo -a
               ame_cpus_per_pool = n/a
                 ame_maxfree_mem = n/a
             ame_min_ucpool_size = n/a
                 ame_minfree_mem = n/a
                 ams_loan_policy = n/a
    enhanced_affinity_affin_time = 1
  enhanced_affinity_vmpool_limit = 10
                  esid_allocator = 1
             force_relalias_lite = 0
               kernel_heap_psize = 65536
                    lgpg_regions = 0
                       lgpg_size = 0
                low_ps_handling = 1
                        maxfree = 1088
                        maxperm = 843105
                         maxpin = 953840
                        maxpin% = 90
                  memory_frames = 1048576
                  memplace_data = 0
```

```
  memplace_mapped_file = 0
memplace_shm_anonymous = 0
    memplace_shm_named = 0
        memplace_stack = 0
         memplace_text = 0
memplace_unmapped_file = 0
               minfree = 960
               minperm = 28103
              minperm% = 3
           msem_nlocks = 0
             nokilluid = 0
               npskill = 1024
               npswarn = 4096
   num_locks_per_semid = 1
             numpsblks = 131072
           pgz_lpgrow = 2
             pgz_mode = 2
       pinnable_frames = 781272
   relalias_percentage = 0
                 scrub = 0
        thrpgio_inval = 1024
       thrpgio_npages = 1024
             v_pinshm = 0
        vm_cpu_thresh = 0
         vm_mmap_bmap = 1
      vmm_default_pspa = 0
        vmm_klock_mode = 2
     wlm_memlimit_nonpg = 1
```

ค่าสำหรับพารามิเตอร์ minfree และ maxfree

วัตถุประสงค์ของรายการที่ว่างคือเก็บประวัติของ page frames หน่วยความจำ จริงที่รีลีสโดยกระบวนการที่ยุติ และเพื่อจัดส่ง page frames ไปให้ผู้ร้องขอ ในทันที โดยไม่ต้องบังคับให้ผู้ร้องขอรอการขโมยหน้าและ I/O ที่มาด้วยกัน

ขีดจำกัด *minfree* ระบุขนาดรายการที่ว่างต่ำสุด ซึ่งการขโมยหน้าเพื่อเติมรายการที่ว่างจะเริ่มต้นขึ้น พารามิเตอร์ *maxfree* เป็นขนาดที่มากกว่า ฝั่งที่ขโมย และใช้ค่า *minfree* เพื่อเริ่มต้นการขโมยเพจ เมื่อจำนวนของเพจที่คงอยู่ เท่ากับหรือน้อยกว่าความแตกต่างระหว่างค่าของพารามิเตอร์ *maxfree* และ *minfree* หรือเมื่อจำนวนของเพจไคลเอ็นต์เท่ากับหรือน้อยกว่าความแตกต่าง ระหว่างค่าของพารามิเตอร์ *maxclient* and *minfree* การขโมยเพจจะเริ่มต้น

วัตถุประสงค์ในการปรับขีดจำกัดเหล่านี้คือเพื่อให้มั่นใจว่า:

- กิจกรรมใดๆ ที่เวลาการตอบกลับเป็นเรื่องสำคัญสามารถได้รับ page frames ที่ต้องการเสมอจากรายการที่ว่าง
- ระบบไม่มีระดับของ I/O ที่สูงโดยไม่จำเป็นเนื่องจาก การขโมยหน้าก่อนเวลาอันควรเพื่อขยายรายการที่ว่าง

ค่าดีฟอลต์ของพารามิเตอร์ *minfree* และ *maxfree* ขึ้นอยู่กับขนาดหน่วยความจำของเครื่อง ผลต่างระหว่างพารามิเตอร์ *maxfree* และ *minfree* ควรจะเท่ากับหรือมากกว่าค่าของพารามิเตอร์ *maxpgahead* เสมอ ถ้าคุณกำลังใช้ JFS สำหรับ Enhanced JFS ผลต่างระหว่างพารามิเตอร์ *maxfree* และ *minfree* ควรจะเท่ากับหรือมากกว่าค่าของพารามิเตอร์ *j2_maxPageReadAhead* เสมอ ถ้าคุณกำลังใช้ทั้ง JFS และ Enhanced JFS คุณควรตั้งค่าของพารามิเตอร์ *minfree* เป็นจำนวนที่มากกว่าหรือเท่ากับค่า pageahead ที่มากกว่าของ สองระบบไฟล์

ค่าพารามิเตอร์ *minfree* และ *maxfree* แตกต่างกันถ้ามีพูลหน่วยความจำมากกว่าหนึ่งพูล พูลหน่วยความจำ มีการใช้ครั้งแรกสำหรับระบบ MP ที่มี RAM จำนวนมาก แต่ละพูลหน่วยความจำมีค่า *minfree* และ *maxfree* ของตนเอง ในเวอร์ชัน AIX ก่อน

หน้านี้ ค่า *minfree* และ *maxfree* ที่แสดงขึ้นโดยคำสั่ง **vmo** คือผลรวมของค่า *minfree* และ *maxfree* สำหรับพูลหน่วยความจำทั้งหมด ค่าที่แสดงขึ้นโดยคำสั่ง **vmo** เป็นค่าสำหรับแต่ละพูลหน่วยความจำ จำนวนของพูลหน่วยความจำสามารถแสดงขึ้นด้วย **vmo -L mempools**

เครื่องมือที่มีความแม่นยำน้อยแต่ครอบคลุมมากขึ้นสำหรับการสืบสวนขนาดที่เหมาะสมสำหรับ *minfree* คือคำสั่ง **vmstat** ข้อมูลต่อไปนี้เป็นส่วนของเอาต์พุตคำสั่ง **vmstat** บนระบบ ที่กำลังจะถึงค่า *minfree*:

```
# vmstat 1
kthr     memory             page              faults        cpu
----- ----------- ------------------------ ------------ -----------
 r  b   avm   fre  re  pi  po  fr   sr  cy  in   sy  cs   us sy id wa
 2  0 70668   414   0   0   0   0    0   0 178 7364  257   35 14  0 51
 1  0 70669   755   0   0   0   0    0   0 196 19119 272   40 20  0 41
 1  0 70704   707   0   0   0   0    0   0 190 8506  272   37  8  0 55
 1  0 70670   725   0   0   0   0    0   0 205 8821  313   41 10  0 49
 6  4 73362   123   0   5  36 313 1646   0 361 16256 863   47 53  0  0
 5  3 73547   126   0   6  26 152  614   0 324 18243 1248  39 61  0  0
 4  4 73591   124   0   3  11  90  372   0 307 19741 1287  39 61  0  0
 6  4 73540   127   0   4  30 122  358   0 340 20097 970   44 56  0  0
 8  3 73825   116   0  18  22 220  781   0 324 16012 934   51 49  0  0
 8  4 74309    26   0  45  62 291 1079   0 352 14674 972   44 56  0  0
 2  9 75322     0   0  41  87 283  943   0 403 16950 1071  44 56  0  0
 5  7 75020    74   0  23 119 410 1611   0 353 15908 854   49 51  0  0
```

ในเอาต์พุตตัวอย่างข้างบน คุณสามารถเห็นว่าค่า *minfree* ที่เป็น 120 มีการขึ้นถึงอย่างต่อเนื่อง ดังนั้น การเปลี่ยนหน้าเกิดขึ้น และในกรณีเฉพาะนี้ แม้แต่รายการที่ว่างยังขึ้นถึง 0 ที่จุดหนึ่ง เมื่อกรณีนั้น เกิดขึ้น เธรดที่ต้องการเฟรมที่ว่างจะถูกบล็อกและไม่สามารถรันได้จนกว่าการเปลี่ยน หน้าจะทำให้บางหน้าว่าง เพื่อป้องกันสถานการณ์นี้ คุณอาจพิจารณา เพิ่มค่า *minfree* และ *maxfree*

หากคุณสรุปว่าคุณควรมีหน้าที่ว่างอย่างน้อย 1000 หน้าต่อพูลหน่วยความจำ เสมอ ให้รันคำสั่งต่อไปนี้:

```
# vmo -o minfree=1000 -o maxfree=1008
```

เพื่อทำให้การเปลี่ยนแปลงนี้เป็นแบบถาวร ให้รวมแฟล็ก **-p**:

```
# vmo -o minfree=1000 -o maxfree=1008 -p
```

ค่าดีฟอลต์ของพารามิเตอร์ *minfree* เพิ่มขึ้นเป็น 960 ต่อพูลหน่วยความจำ และค่าดีฟอลต์ของพารามิเตอร์ *maxfree* เพิ่มขึ้นเป็น 1088 ต่อพูลหน่วยความจำ

พูลหน่วยความจำ

คำสั่ง **vmo -o mempools**=*number_of_memory_pools* อนุญาตให้คุณเปลี่ยนจำนวนของพูลหน่วยความจำที่ถูกปรับแต่ง ณ ตอนที่บูตระบบ

อ็อพชัน **mempools** ไม่ใช่การเปลี่ยนแบบไดนามิก ซึ่งขอแนะนำว่า ไม่ให้เปลี่ยนค่านี้โดยที่คุณไม่เข้าใจลักษณะการทำงานของระบบ และอัลกอริธึม VMM

ความสามารถในการปรับแต่งนี้ควรจะถูกปรับเปลี่ยนเมื่อได้รับคำแนะนำให้เปลี่ยนโดยตัวแทนบริการของ IBM

LRU แบบอิงรายการ

อัลกอริทึม LRU ใช้รายการ ใน AIX เวอร์ชันก่อนหน้านี้ วิธี page frame table ยังพร้อมใช้งานด้วย อัลกอริธึมแบบอิงรายการจะจัดเตรียมรายการของเพจที่สแกนไว้สำหรับชนิดของเซ็กเมนต์แต่ละชนิด

ต่อไปนี้คือรายการของชนิดของเซ็กเมนต์:

- การทำงาน
- การคงอยู่
- ไคลเอ็นต์
- การบีบอัด

ถ้าเปิดใช้งาน WLM แล้ว จะมีรายการสำหรับคลาสแสดงอยู่เช่นกัน

การลดการใช้การสแกนหน่วยความจำด้วยพารามิเตอร์ lrubucket

การปรับด้วยพารามิเตอร์ *lrubucket* สามารถลดการใช้การสแกนบนระบบที่มีหน่วยความจำขนาดใหญ่

อัลกอริธึมการแทนที่เพจจะสแกนกรอบของหน่วยความจำที่มองหากรอบอิสระ ในระหว่างการสแกนนี้ การอ้างถึงบิตของเพจจะถูกรีเซ็ตค่า และถ้าไม่พบกรอบอิสระ การสแกนครั้งที่สองจะถูกดำเนินการ ในการสแกนครั้งที่สอง หากการอ้างถึงบิตยังคงปิดอยู่ กรอบจะถูกนำมาใช้สำหรับเพจใหม่ (การแทนที่เพจ)

สำหรับระบบที่มีหน่วยความจำขนาดใหญ่ อาจมีกรอบจำนวนมากเกินไปที่ต้องสแกน ดังนั้น หน่วยความจำปัจจุบันจะถูกแบ่งออกเป็นที่ฝากข้อมูลของกรอบ อัลกอริธึมการแทนที่เพจ จะสแกนกรอบในที่ฝากข้อมูล จากนั้นเริ่มต้นทำงานที่ฝากข้อมูลสำหรับการสแกนครั้งที่สอง ก่อนที่จะย้ายไปยังที่ฝากข้อมูลถัดไป จำนวนดีฟอลต์ของกรอบในที่ฝากข้อมูลนี้คือ 131072 หรือ 512 MB ของ RAM จำนวนของกรอบสามารถปรับแต่งได้ด้วยคำสั่ง **vmo -o lrubucket**=*new value* และค่าจะอยู่ในกรอบขนาด 4 KB

ค่าสำหรับพารามิเตอร์ minperm และ maxperm

ระบบปฏิบัติการใช้ประโยชน์จากความต้องการที่แตกต่างกันไปสำหรับ หน่วยความจำโดยการปล่อยไว้ในหน้าหน่วยความจำของไฟล์ที่ถูกอ่านหรือ บันทึก

ถ้ามีการร้องขอหน้าไฟล์อีกครั้งหนึ่งก่อนจะมีการกำหนด page frames ให้ใหม่ เทคนิคนี้จะช่วยประหยัดการดำเนินงาน I/O หน้าไฟล์เหล่านี้อาจมาจากระบบไฟล์แบบโลคัล หรือรีโมต (ตัวอย่างเช่น NFS)

อัตราส่วนของ page frames ที่ใช้สำหรับไฟล์ต่อจำนวนที่ใช้สำหรับเซกเมนต์ การคำนวณ (การทำงานหรือข้อความโปรแกรม) มีการควบคุมบางส่วนโดยค่า *minperm* และ *maxperm*:

- ถ้าเปอร์เซ็นต์ของ RAM ที่ใช้โดยหน้าไฟล์ลดลงต่ำกว่า *minperm* การเปลี่ยนหน้าจะขโมยทั้งหน้าไฟล์และหน้าการคำนวณ
- หากเปอร์เซ็นต์ของ RAM ที่ใช้โดยเพจไฟล์อยู่ระหว่าง *minperm* และ *maxperm* page-replacement จะขโมยเฉพาะเพจไฟล์

ในเวิร์กโหลดเฉพาะ อาจคุ้มค่าที่จะเน้นการหลีกเลี่ยง ไฟล์ I/O ในเวิร์กโหลดอื่น การเก็บหน้าเซกเมนต์การคำนวณไว้ในหน่วยความจำ อาจมีความสำคัญมาก เมื่อต้องการทราบอัตราที่ยังไม่มีการปรับ ให้ใช้คำสั่ง **vmstat** ที่มีอ็อพชัน **-v**

```
# vmstat -v
             1048576 memory pages
              936784 lruable pages
              683159 free pages
```

```
     1 memory pools
267588 pinned pages
  90.0 maxpin percentage
   3.0 minperm percentage
  90.0 maxperm percentage
   5.6 numperm percentage
 52533 file pages
   0.0 compressed percentage
     0 compressed pages
   5.6 numclient percentage
  90.0 maxclient percentage
 52533 client pages
     0 remote pageouts scheduled
     0 pending disk I/Os blocked with no pbuf
     0 paging space I/Os blocked with no psbuf
  2228 filesystem I/Os blocked with no fsbuf
    31 client filesystem I/Os blocked with no fsbuf
     0 external pager filesystem I/Os blocked with no fsbuf
  29.8 percentage of memory used for computational pages
```

ค่าเปอร์เซ็นต์ *numperm* แสดงเปอร์เซ็นต์ของ หน่วยความจำจริงที่ใช้โดยเซ็กเมนต์ไฟล์ ค่า 5.6% สอดคล้องกับ 52533 เพจไฟล์ในหน่วยความจำ

ขีดจำกัดระบบไฟล์ JFS ขั้นสูง Maxclient

Maxclient แทนจำนวนสูงสุด ของเพจของไคลเอ็นต์ที่สามารถใช้สำหรับบัฟเฟอร์แคช

ระบบไฟล์ JFS ที่ปรับปรุงแล้วจะใช้ไคลเอ็นต์เพจสำหรับแคชของบัฟเฟอร์ ขีดจำกัดสำหรับเพจไคลเอ็นต์ในหน่วยความจำจริงถูกบังคับใช้โดยใช้ *maxclient*

LRU daemon จะเริ่มต้นรันเมื่อจำนวนของเพจไคลเอ็นต์ อยู่ภายในจำนวนของเพจ *minfree* ของขีดจำกัดของ *maxclient* LRU daemon พยายามใช้ไคลเอ็นต์เพจที่ไม่ได้อ้างถึงล่าสุด ถ้าจำนวนของเพจไฟล์ต่ำกว่าค่าของพารามิเตอร์ *minperm* เพจใดๆ ที่ไม่ได้อ้างถึงสามารถเขียนได้สำหรับการแทนที่

Maxclient ยังมีผลกับไคลเอ็นต์ NFS และ เพจที่บีบอัด นอกจากนี้ โปรดสังเกตว่า โดยทั่วไป *maxclient* จะถูกตั้งค่าเป็นค่าที่น้อยกว่าหรือเท่ากับพารามิเตอร์ *maxperm*

การจัดสรรพื้นที่ว่างหน้า

มีนโยบายการจัดสรรพื้นที่เพจจำนวนมากที่ใช้ใน AIX

- การจัดสรรพื้นที่ว่างหน้าที่เลื่อนออกไป (Deferred Page Space Allocation หรือ DPSA)
- การจัดสรรพื้นที่ว่างหน้าโดยเร็ว (Early Page Space Allocation หรือ EPSA)

การจัดสรรพื้นที่เพจที่เลื่อนออกไป

นโยบายการจัดสรรเพจที่เลื่อนออกไปคือนโยบายดีฟอลต์ใน AIX

ด้วยการจัดสรรพื้นที่เพจที่เลื่อนออกไป การจัดสรรบล็อกดิสก์ของพื้นที่การเพจ จะถูกเลื่อนออกไปจนกว่าจะจำเป็นต้องเพจออกจากเพจ ซึ่งส่งผลทำให้การจัดสรรพื้นที่การเพจไม่ได้ใช้ ซึ่งอนุญาตให้ อัลกอริทึมแบบ deferred พยายามจัดสรรพื้นที่การเพจมากกว่า ที่มี ซึ่งทำให้เกิด over-commitment ของพื้นที่การเพจ

หลังจากที่เพจได้ถูกเพจออกไปจากพื้นที่การเพจแล้ว การบล็อกดิสก์ จะถูกสงวนไว้สำหรับเพจนั้น หากเพจนั้นถูกเพจกลับไปยัง RAM ดังนั้น ค่าเปอร์เซ็นต์ที่ใช้สำหรับพื้นที่การเพจอาจไม่จำเป็นสำหรับการสะท้อนถึงจำนวนของเพจ ที่อยู่ในพื้นที่การเพจเท่านั้น เนื่องจากเพจนั้นอาจกลับไปยัง RAM เช่นกัน ถ้าเพจไม่ถูกเพจกลับไปยังหน่วยเก็บใช้งาน ของเธรด และถ้าเธรดปล่อยหน่วยความจำ ที่เชื่อมโยงกับเพจนั้นแล้ว หรือถ้าเธรดนั้นมีอยู่แล้ว บล็อกดิสก์สำหรับเพจนั้นจะถูกปล่อย ดิสก์บล็อกที่อยู่ในพื้นที่การเพจของเพจ ที่ถูกอ่านกลับเข้าในหน่วยความจำหลักสามารถรีลีสได้โดยใช้ คุณลักษณะการรวบรวมขยะในพื้นที่การเพจ สำหรับข้อมูลโดยละเอียด โปรดดู “การเก็บรวบรวมขยะของพื้นที่ว่างการเพจ” ในหน้า 173

ถ้าการเก็บรวบรวมขยะสำหรับพื้นที่การเพจไม่ได้เปิดใช้งาน สิ่งสำคัญอย่างยิ่งคือ การปรับแต่งจำนวนของพื้นที่เพจให้ถูกต้อง หากแคชไฟล์ ต่ำกว่า minperm และหากไม่ได้กำหนดค่าพื้นที่การเพจที่เพียงพอ คุณอาจต้องปรับแต่งระบบเพื่อป้องกันเพจของหน่วยเก็บข้อมูลที่ใช้งาน ถูกเพจออกเนื่องจากกิจกรรมการเพจไฟล์ หากความต้องการหน่วยเก็บข้อมูลที่ใช้งาน ของเวิร์กโหลดน้อยกว่าจำนวนของหน่วยความจำจริง และหากระบบได้รับการปรับแต่งเพื่อให้กิจกรรมเพจไฟล์ไม่ เพจเพจหน่วยเก็บข้อมูลที่ใช้งานออก จำนวนของพื้นที่การเพจที่ต้องการ อาจน้อย บางเซ็กเมนต์พื้นที่ตารางเพจ (PTA) ซึ่งไม่ใช่เซ็กเมนต์ การจัดสรรแบบ deferred เรียกว่าเซ็กเมนต์หน่วยความจำเคอร์เนล AIX ภายใน เพื่อรวมการสงวนพื้นที่การเพจที่เซ็กเมนต์เหล่านี้ต้องการ ระบบแนะนำให้มีพื้นที่การเพจ 512 MB หาก ระบบใช้พื้นที่ PTA จำนวนมาก คุณอาจต้องการ พื้นที่การเพจมากขึ้น ซึ่งสามารถตรวจสอบโดยใช้คำสั่ง **svmon -S**

ถ้าข้อกำหนดหน่วยเก็บใช้งานมีค่าสูงกว่าจำนวนของหน่วยความจำที่ใช้จริง คุณต้องมีอย่างน้อยเท่ากับพื้นที่การเพจที่ปรับแต่งไว้เป็นขนาดของ หน่วยความจำเสมือนหน่วยเก็บใช้งาน หรือ ระบบอาจมีพื้นที่การเพจ ที่ไม่เพียงพอ

การจัดสรรพื้นที่เพจก่อนหน้า

ถ้าคุณต้องการมั่นใจว่า การประมวลผลไม่ได้หยุดทำงาน เนื่องจากเงื่อนไขการเพจต่ำ การประมวลผลนี้สามารถจัดสรรพื้นที่การเพจไว้ล่วงหน้า โดยใช้นโยบายการจัดสรรพื้นที่การเพจก่อนหน้า

เหตุการณ์นี้จะทำโดยการตั้งค่าตัวแปรสภาวะแวดล้อมที่เรียกว่า *PSALLOC* ให้มีค่าก่อนหน้า ซึ่งสามารถทำได้ภายในการประมวลผลหรือที่บรรทัดรับคำสั่ง (คำสั่ง **PSALLOC=early**) เมื่อการประมวลผลใช้รูทีนย่อย **malloc( )** เพื่อจัดสรรหน่วยความ หน่วยความจำนี้จะมีบล็อกดิสก์สำหรับพื้นที่การเพจที่สงวนไว้ สำหรับการประมวลผลนี้ นั่นคือ บล็อกเหล่านี้จะถูกสงวนไว้สำหรับการประมวลผลนี้ ดังนั้น จึงมีการรับประกันว่า การประมวลผลจำเป็นต้องเพจออก ซึ่งจะเป็นสล็อตพื้นที่การเพจที่พร้อมใช้งาน ถ้าการใช้นโยบายก่อนหน้า และถ้าการประหยัด CPU ถูกนำมาพิจารณาร่วม คุณอาจต้องการตั้งค่าตัวแปรสภาวะแวดล้อมอื่นๆ ที่เรียกว่า **NODISCLAIM=true** ดังนั้น แต่ละการเรียกรูทีนย่อย **free( )** จะไม่ส่งผลในการเรียกของระบบ **disclaim( )**

พื้นที่ว่างการเพจและหน่วยความจำเสมือน

คำสั่ง **vmstat**(*avm* column), คำสั่ง **ps** (*SIZE*, *SZ*) และยูทิลิตีอื่นรายงานจำนวนของหน่วยความจำเสมือน ที่เข้าถึงอย่างแท้จริง เนื่องจากเมื่อใช้ DPSA อาจไม่มีการแตะต้องพื้นที่ว่าง การเพจ

การใช้คำสั่ง **lsps -s** ปลอดภัยกว่าการใช้ คำสั่ง **lsps -a** เพื่อดูพื้นที่ว่างการเพจที่มีอยู่ เนื่องจากคำสั่ง **lsps -a** แสดงเฉพาะ พื้นที่ว่างการเพจที่กำลังใช้อยู่จริงเท่านั้น แต่คำสั่ง **lsps -s** รวม พื้นที่ว่างการเพจที่กำลังใช้อยู่และพื้นที่ว่างการเพจที่ถูกสำรองไว้โดยการใช้นโยบาย EPSA

การปรับ thresholds ของพื้นที่ว่างการเพจ

ถ้าพื้นที่ว่างการเพจที่มีอยู่ใช้จนเหลือในระดับต่ำแล้ว ระบบปฏิบัติการจะ พยายามรีลีสรีซอร์สโดยการเตือนกระบวนการให้รีลีสพื้นที่ว่างการเพจก่อน เป็นอันดับแรก จากนั้นจะ kill กระบวนการถ้ายังคงมีพื้นที่ว่างการเพจอยู่ไม่เพียง พอสำหรับกระบวนการปัจจุบัน

ค่าสำหรับพารามิเตอร์ npswarn และ npskill

npswarn และ *npskill* thresholds ใช้โดย VMM เพื่อกำหนดว่ากระบวนการเตือนครั้งแรกเกิดขึ้นเมื่อไร และในที่สุด จะ kill กระบวนการเมื่อไร

พารามิเตอร์สองตัวนี้สามารถตั้งค่าผ่านทางคำสั่ง **vmo**:

npswarn

ระบุจำนวนของหน้าพื้นที่ว่างการเพจที่ว่างซึ่งระบบปฏิบัติการจะเริ่มต้น การส่งสัญญาณ SIGDANGER ไปยังกระบวนการ ถ้าใช้งานถึง *npswarn* threshold และกระบวนการกำลังจัดการสัญญาณนี้ กระบวนการสามารถเลือกที่จะละเว้น หรือทำการดำเนินการอื่นบางอย่าง เช่น จบการทำงาน หรือรีลีสหน่วยความจำ โดยใช้รูทีนย่อย **disclaim( )**

ค่าของ *npswarn* ต้องมากกว่าศูนย์ และน้อยกว่าจำนวนทั้งหมดของหน้าพื้นที่ว่างการเพจบนระบบ ค่านี้สามารถเปลี่ยนได้ด้วยคำสั่ง **vmo -o npswarn=value**

npskill ระบุจำนวนของหน้าพื้นที่ว่างการเพจที่ว่างซึ่งระบบปฏิบัติการจะเริ่มต้น killing กระบวนการ ถ้าใช้งานถึง *npskill* threshold สัญญาณ SIGKILL จะถูกส่งไปยังกระบวนการที่อายุน้อยที่สุด กระบวนการที่กำลัง จัดการ SIGDANGER หรือกระบวนการที่กำลังใช้การจัดสรรพื้นที่ว่างหน้าระยะแรก (พื้นที่ว่างการเพจมีการจัดสรรในทันทีที่ร้องขอหน่วยความจำ) จะถูกยกเว้นจากการ killed สูตรในการกำหนดค่าดีฟอลต์ของ *npskill* เป็นดังนี้:

```
npskill = maximum (64, number_of_paging_space_pages/128)
```

ค่า **npskill** ต้องมากกว่าศูนย์หรือน้อยกว่าจำนวนทั้งหมดของหน้าพื้นที่ว่างการเพจ บนระบบ ค่านี้สามารถ เปลี่ยนได้ด้วยคำสั่ง **vmo -o npskill=value**

nokilluid

โดยการตั้งค่าอ็อพชัน *nokilluid* เป็นค่าที่ไม่ใช่ศูนย์ ด้วยคำสั่ง **vmo -o nokilluid** IDs ผู้ใช้ ที่ต่ำกว่าค่านี้จะได้รับยกเว้นจากการถูก killed เนื่องจากเงื่อนไขพื้นที่ว่างหน้า ต่ำ ตัวอย่างเช่น ถ้า **nokilluid** มีการตั้งค่าเป็น 1 กระบวนการที่เป็นของรากจะได้รับยกเว้น จากการถูก killed เมื่อใช้งานถึง **npskill** threshold

พารามิเตอร์ช่วงเวลาในการลองพยายาม fork()

ถ้าการประมวลผลไม่สามารถแตกออกได้เนื่องจากขาดเพจพื้นที่การเพจ ตัวกำหนดตารางจะลองพยายามแตกห้าครั้ง ในระหว่างความพยายามลองแต่ละครั้ง ตัวกำหนดตารางเวลาจะหน่วงเวลาสำหรับค่าดีฟอลต์แท่งนาฬิกา 10 แท่ง

พารามิเตอร์ *pacefork* ของคำสั่ง **schedo** จะระบุจำนวนของแท่งนาฬิกาเพื่อรอก่อนที่จะพยายามลองเรียก **fork( )** ที่ล้มเหลวอีกครั้ง ตัวอย่างเช่น ถ้าการเรียกรูทีนย่อย **fork( )** เกิดความล้มเหลว เนื่องจากมีพื้นที่ไม่เพียงพอในการสร้างการประมวลผลใหม่ ระบบจะพยายามลองเรียกใหม่ หลังจากที่รองจำนวนของแท่งนาฬิกาที่ระบุไว้ ค่าดีฟอลต์คือ 10 และเนื่องจากมีแท่งนาฬิกาเพียงแท่งเดียวในทุกๆ 10 มิลลิวินาที ระบบจะพยายามลองเรียก **fork( )** ทุกๆ 100 มิลลิวินาที

ถ้าพื้นที่การเพจมีขนาดน้อยลงเนื่องจากการย่อ การกระจายเวิร์กโหลดสูงสุด การเพิ่มช่วงเวลาในการพยายามอาจอนุญาตให้การประมวลผลเกิดเวลาหน่วงยาวนานเพียงพอที่จะปล่อย เช่นเดียวกับตัวอย่างต่อไปนี้:

```
# schedo -o pacefork=15
```

ด้วยวิธีนี้ เมื่อระบบพยายามลองเรียก **fork( )** แล้ว การเรียกนั้นมีโอกาสที่จะประสบความสำเร็จสูง เนื่องจากการประมวลผลบางอย่างอาจเสร็จสิ้นการประมวลผล และปล่อยเพจออกจากพื้นที่การเพจ

การเก็บรวบรวมขยะของพื้นที่ว่างการเพจ

คุณสามารถใช้คุณลักษณะการรวบรวมขยะของพื้นที่ว่างการเพจ เพื่อทำให้ดิสก์บล็อกของพื้นที่ว่างการเพจเป็นอิสระภายใต้เงื่อนไขเฉพาะ เพื่อให้ คุณไม่ต้องกำหนดคอนฟิกพื้นที่ว่างการเพจมากเท่ากับจำนวนของ หน่วยความจำเสมือนซึ่งใช้สำหรับเวิร์กโหลดเฉพาะ คุณลักษณะการเก็บรวบรวมขยะมีอยู่สำหรับนโยบาย การจัดสรรพื้นที่ว่างหน้าที่เลื่อนออกไปเท่านั้น

การเก็บรวบรวมขยะบนบล็อกพื้นที่การเพจหลังจากที่เพจเข้าใหม่

เมธอดของการล้างบล็อกดิสก์พื้นที่การเพจหลังจากเพจได้อ่านกลับไปยังหน่วยความจำ จากพื้นที่การเพจที่ถูกใช้ตามค่าดีฟอลต์

เหตุผลที่ไม่มีการล้างข้อมูลสำหรับการเพจเข้าทุกครั้ง เนื่องจากการทิ้งบล็อกให้อยู่ในพื้นที่การเพจซึ่งแสดงถึงผลการทำงานที่ดีกว่าในกรณีของ เพจหน่วยเก็บใช้งานที่ไม่แก้ไขที่ถูกใช้โดย LRU daemon ถ้าเพจได้ถูกนำมาใช้ จึงไม่มีความจำเป็นที่จะต้องดำเนินการฟังก์ชันเพจออกอีกครั้ง

คุณสามารถปรับพารามิเตอร์ต่อไปนี้ด้วยคำสั่ง **vmo** :

การปรับแต่งพารามิเตอร์ **npsrpgmin**:

Item	Descriptor
วัตถุประสงค์:	ระบุจำนวนของ threshold ของบล็อกพื้นที่การเพจที่ล้างข้อมูลแล้ว เมื่อเพจเข้าอีกครั้งของการเก็บรวบรวมขยะที่เริ่มต้นขึ้น
ค่า:	ดีฟอลต์: MAX (768, **npswarn**+ (**npswarn**/2)
ช่วง:	0 ถึงจำนวนทั้งหมดของบล็อกพื้นที่การเพจ ในระบบ

การปรับแต่งพารามิเตอร์ **npsrpgax**:

Item	Descriptor
วัตถุประสงค์:	ระบุจำนวนของ threshold บล็อกพื้นที่การเพจอิสระ เมื่อเพจเข้าของการเก็บรวบรวมขยะหยุดทำงาน
ค่า:	ดีฟอลต์: MAX (1024, **npswarn***2)

การปรับแต่งพารามิเตอร์ **rpgclean**:

Item	Descriptor
วัตถุประสงค์:	เปิดใช้งานหรือปิดใช้งานการล้างข้อมูลของ บล็อกพื้นที่การเพจจากนโยบายการจัดสรรพื้นที่เพจที่เลื่อน สำหรับการเข้าถึงการอ่าน
ค่า:	ดีฟอลต์: 0 หมายถึงบล็อกของดิสก์พื้นที่การเพจอิสระเฉพาะเพจเข้าของเพจ ที่กำลังแก้ไข ค่า 1 หมายถึงบล็อกดิสก์พื้นที่การเพจอิสระสำหรับเพจเข้าของ เพจที่กำลังแก้ไขหรือเข้าถึง หรืออ่าน
ช่วง:	0 ! 1

การปรับแต่งพารามิเตอร์ **rpgcontrol**:

Item	Descriptor
วัตถุประสงค์:	เปิดใช้งานหรือปิดใช้งานการล้างข้อมูลของ บล็อกพื้นที่การเพจที่เพจเข้าของเพจจากนโยบายการจัดสรรพื้นที่เพจที่เลื่อนออกไป
ค่า:	ดีฟอลต์: 2 หมายถึง เปิดใช้งานการล้างข้อมูลของบล็อกดิสก์พื้นที่การเพจสำหรับเพจเข้า โดยไม่พิจารณาถึง thresholds **หมายเหตุ:** การเข้าถึงการอ่านที่ประมวลผลแล้ว ถ้าค่าของพารามิเตอร์ **rpgcontrol** คือ 1 ตามค่าดีฟอลต์ การเข้าถึงการเขียน จะถูกประมวลผลเสมอค่า 0 จะปิดใช้งานการล้างข้อมูลบล็อกดิสก์พื้นที่การเพจ สำหรับเพจเข้า
ช่วง:	0 ⁞ 1 ⁞ 2

การเก็บรวบรวมขยะโดยล้างหน่วยความจำ

เมธอดอื่นๆ ของการเก็บรวบรวมขยะของพื้นที่การเพจโดยการล้างหน่วยความจำ ซึ่งนำมาใช้พร้อมกับการประมวลผลเคอร์เนล **psgc**

การประมวลผลเคอร์เนล **psgc** จะล้างข้อมูลบล็อกดิสก์พื้นที่การเพจสำหรับเพจหน่วยความจำที่แก้ไข ซึ่งยังไม่ได้เพจออกอีกครั้งหรือสำหรับเพจที่ไม่ได้แก้ไขไว้ ซึ่งบล็อกดิสก์การเพจยังคงอยู่

การประมวลผลเคอร์เนล **psgc** จะใช้พารามิเตอร์ที่สามารถปรับแต่งได้ต่อไปนี้ ซึ่งคุณสามารถปรับได้ด้วยคำสั่ง **vmo** :

การปรับแต่งพารามิเตอร์ **npsscrubmin** ประกอบด้วย ฟิลด์ต่อไปนี้:

Item	Descriptor
วัตถุประสงค์:	ระบุจำนวนของการล้างข้อมูลบล็อกพื้นที่การเพจ ที่การล้างเพจหน่วยความจำเริ่มต้นเพื่อล้างข้อมูลบล็อกดิสก์ จากเพจจากนโยบายการจัดสรรพื้นที่เพจที่เลื่อนออกไป
ค่า:	ดีฟอลต์: MAX (768, ค่าของพารามิเตอร์ **npsrpgmin**)
ช่วง:	0 ถึงจำนวนทั้งหมดของบล็อกพื้นที่การเพจ ในระบบ

การปรับแต่งพารามิเตอร์ **npsscrubmax** ประกอบด้วยฟิลด์ต่อไปนี้:

Item	Descriptor
วัตถุประสงค์:	ระบุจำนวนของการล้างข้อมูลบล็อกพื้นที่การเพจ ที่การล้างเพจหน่วยความจำหยุดทำงานเพื่อล้างข้อมูลบล็อกดิสก์ จากเพจจากนโยบายการจัดสรรพื้นที่เพจที่เลื่อนออกไป
ค่า:	ดีฟอลต์: MAX (1024, ค่าของพารามิเตอร์ **npsrpgmax**)
ช่วง:	0 ถึงจำนวนทั้งหมดของบล็อกพื้นที่การเพจ ในระบบ

การปรับแต่งพารามิเตอร์ **scrub** ประกอบด้วย ฟิลด์ต่อไปนี้:

Item	Descriptor
วัตถุประสงค์:	เปิดใช้งานหรือปิดใช้งานการล้างข้อมูลบล็อกดิสก์พื้นที่การเพจ จากเพจในหน่วยความจำจากเพจของนโยบายการจัดสรร พื้นที่เพจที่เลื่อนออกไป
ค่า:	ดีฟอลต์: 0 ซึ่งเปิดใช้งานการล้างหน่วยความจำอย่างสมบูรณ์ ถ้าค่าถูกตั้งค่าเป็น 1 การล้างข้อมูลหน่วยความจำของบล็อกดิสก์พื้นที่การเพจถูกเปิดใช้งาน เมื่อจำนวนของระบบ สำหรับบล็อกพื้นที่การเพจอิสระจะมีค่าต่ำกว่าค่าของพารามิเตอร์ **npsscrubmin** และมากกว่าค่าของพารามิเตอร์ *npsscrubmax*
ช่วง:	0 ⁞ 1

การปรับแต่งพารามิเตอร์ **scrubclean** ประกอบด้วยฟิลด์ต่อไปนี้:

Item	Descriptor
วัตถุประสงค์:	เปิดใช้งานหรือปิดใช้งานการล้างข้อมูลบล็อกดิสก์พื้นที่การเพจจากเพจ ในหน่วยความจำจากเพจของนโยบายการจัดสรรพื้นที่เพจที่ไม่ได้แก้ไข
ค่า:	ดีฟอลต์: 0 หมายถึงบล็อกดิสก์พื้นที่การเพจอิสระเฉพาะสำหรับเพจที่แก้ไขในหน่วยความจำ ถ้าค่าถูกตั้งค่าเป็น 1 ให้ล้างข้อมูลบล็อกดิสก์พื้นที่การเพจสำหรับเพจที่แก้ไข หรือไม่ได้แก้ไข
ช่วง:	0 | 1

หน่วยความจำแบบแบ่งใช้

โดยการใช้รูทีนย่อย **shmat( )** หรือ **mmap( )** สามารถแม็พ ไฟล์เข้าในหน่วยความจำด้วยตนเองได้ กระบวนการนี้ช่วยหลีกเลี่ยงการบัฟเฟอร์และโอเวอร์เฮด การเรียกระบบ

พื้นที่หน่วยความจำมีการเรียกกันว่าเซกเมนต์หรือส่วนหน่วยความจำแบบแบ่งใช้ สำหรับแอ็พพลิเคชัน 32 บิตที่ได้รับผลกระทบ มีการรีลีสเซ็กเมนต์ 14 ซึ่งให้เซ็กเมนต์หน่วยความจำแบบแบ่งใช้จำนวน 11 เซ็กเมนต์ที่ ไม่รวมเซ็กเมนต์ข้อมูลไลบรารีแบบแบ่งใช้หรือข้อความไลบรารีแบบแบ่งใช้ เมธอดนี้ใช้สำหรับกระบวนการที่มีเซ็กเมนต์ 3–12 และ 14 แต่ละ เซ็กเมนต์เหล่านี้มีขนาด 256 MB แอ็พพลิเคชันสามารถอ่านหรือเขียน ไฟล์โดยอ่านหรือเขียนในเซ็กเมนต์ แอ็พพลิเคชันสามารถหลีกเลี่ยงโอเวอร์เฮด ของการเรียกระบบการอ่านหรือการเขียนโดยใช้ตัวชี้ในเซ็กเมนต์ที่แม็พ เหล่านี้

ไฟล์หรือข้อมูลยังสามารถแบ่งใช้ระหว่างหลายกระบวนการหรือเธรดได้ด้วย อย่างไรก็ตาม ต้องมีการซิงโครไนซ์ระหว่างกระบวนการหรือเธรดเหล่านี้ และการจัดการ คำร้องขอดังกล่าวขึ้นอยู่กับแอ็พพลิเคชัน โดยปกติ หน่วยความจำแบบแบ่งใช้มีการใช้โดยแอ็พพลิเคชันฐานข้อมูล ซึ่งใช้ ฐานข้อมูลเป็นแคชบัฟเฟอร์ฐานข้อมูลขนาดใหญ่

พื้นที่ว่างการเพจมีการจัดสรรสำหรับส่วนหน่วยความจำแบบแบ่งใช้เช่นเดียวกับ ที่จัดสรรสำหรับเซกเมนต์ส่วนตัวของกระบวนการ พื้นที่ว่างการเพจมีการใช้เมื่อเข้าถึง เพจ ถ้านโยบายการจัดสรรพื้นที่ว่างเพจที่เลื่อนออกไปเป็น ปิด

หน่วยความจำที่แบ่งใช้ซึ่งขยายเพิ่ม

หน่วยความจำที่แบ่งใช้ซึ่งขยายเพิ่มอนุญาตให้ใช้การประมวลผลแบบ 32 บิต เพื่อจัดสรรเซ็กเมนต์หน่วยความจำที่แบ่งใช้ให้เล็กที่สุดเท่ากับหนึ่งไบต์ แล้วจึงปัดให้ใกล้เคียงกับเพจมากที่สุด คุณลักษณะนี้ พร้อมใช้งานเพื่อประมวลผลที่มีตัวแปร **EXTSHM** ที่ตั้งค่าเป็น **ON, 1SEG** หรือ **MSEG** อย่างใดอย่างหนึ่งในสภาวะแวดล้อมการประมวลผล

หน่วยความจำที่แบ่งใช้ที่ขยายเพิ่มจะลบข้อจำกัดของส่วนของหน่วยความจำที่แบ่งใช้ 11 ส่วนเท่านั้น การประมวลผลแบบ 64 บิตจะไม่กระทบกับตัวแปร **EXTSHM**

การตั้งค่า **EXTSHM** ให้มีค่าเป็น **ON** มีผลกระทบแบบเดียวกับ การตั้งค่าตัวแปรให้มีค่า **1SEG** ด้วยค่าติดตั้งอย่างใดอย่างหนึ่งนี้ หน่วยความจำที่แบ่งใช้ใดๆ ที่มีขนาดน้อยกว่า 256 MB จะถูกสร้างขึ้นเป็นเซ็กเมนต์ mmap อยู่ภายใน และมีผลการทำงานแบบเดียวกับ mmap ที่เกี่ยวข้อง หน่วยความจำที่แบ่งใช้ใดๆ ที่มีขนาดมากกว่าหรือเท่ากับ 256 MB เซ็กเมนต์การทำงานภายใน

ถ้า **EXTSHM** มีค่าเป็น **MSEG** หน่วยความจำที่แบ่งใช้ทั้งหมดจะถูกสร้างขึ้นเป็นเซ็กเมนต์ mmap อยู่ภายใน ซึ่งอนุญาตสำหรับการใช้ประโยชน์จากหน่วยความจำได้ดีขึ้นกว่าเดิม

ไม่มีข้อจำกัดเกี่ยวกับจำนวนของส่วนของหน่วยความจำที่แบ่งใช้ซึ่งการประมวลผลพ่วงต่ออยู่ การแม็พไฟล์จะสนับสนุนเหมือนก่อน แต่จะใช้พื้นที่แอดเดรส ที่มี 256 MB (ขนาดเซ็กเมนต์) จำนวนมาก การปรับขนาดส่วนของหน่วยความจำที่แบ่งใช้ไม่ได้รับการสนับสนุนในโหมดนี้ การประมวลผลเคอร์เนลมีลักษณะการทำงานที่เหมือนเดิม

หน่วยความจำที่แบ่งใช้ที่ขยายเพิ่มมีข้อจำกัดต่อไปนี้:

- ส่วนสนับสนุน I/O ที่ถูกจำกัดตามปกติสำหรับส่วนของหน่วยความจำที่แม็พ
- เฉพาะ **uphysio( )** ที่เป็นชนิด I/O จะได้รับการสนับสนุน (ไม่มี I/O)
- ส่วนของหน่วยความจำที่แบ่งใช้เหล่านี้ไม่สามารถใช้เป็นบัฟเฟอร์ I/O ซึ่งการยกเลิกการตรึงบัฟเฟอร์ไว้จะเกิดขึ้นใน handler อินเตอร์รัปต์ ตัวอย่างเช่น ส่วนเหล่านี้ไม่สามารถนำมาใช้ สำหรับบัฟเฟอร์ I/O อะซิงโครนัส
- เซ็กเมนต์ไม่สามารถตรึงโดยใช้รูทีนย่อย **plock( )** ได้ เนื่องจากเซ็กเมนต์หน่วยความจำที่แม็พไม่สามารถตรึงได้ด้วยรูทีนย่อย **plock( )**

การกำหนดสมนามเซกเมนต์ 1 TB

การกำหนดสมนามเซกเมนต์ 1TB ช่วยปรับปรุงประสิทธิภาพโดยใช้การแปล เซกเมนต์ 1 TB บนพื้นที่หน่วยความจำแบบแบ่งใช้ที่มีขนาดเซกเมนต์ 256 MB การสนับสนุนนี้ มีอยู่ในแอ็พพลิเคชัน 64 บิตทั้งหมดที่ใช้พื้นที่หน่วยความจำ แบบแบ่งใช้ การติดตั้งหน่วยความจำแบบแบ่งใช้ทั้งทางตรงและทางอ้อมมีสิทธิ กำหนดสมนามเซกเมนต์ 1 TB ได้

หากแอ็พพลิเคชันมีสิทธิมีพื้นที่หน่วยความจำแบบแบ่งใช้เพื่อใช้ สมนาม 1 TB ระบบปฏิบัติการ AIX จะใช้การแปลเซกเมนต์ 1 TB โดยไม่เปลี่ยนแอ็พพลิเคชัน ซึ่งจำเป็นต้องใช้ `shm_1tb_shared` VMO tunable, `shm_1tb_unshared` VMO tunable, และ `esid_allocator` VMO tunable

`shm_1tb_shared` VMO สามารถตั้งค่าได้ในแบบ ต่อกระบวนการโดยใช้ตัวแปรสภาวะแวดล้อม *"SHM_1TB_SHARED=" VMM_CNTRL* ค่าดีฟอลต์มีการตั้งค่าแบบไดนามิก ณ เวลาบูตตาม ความสามารถของตัวประมวลผล หากพื้นที่หน่วยความจำแบบแบ่งใช้เดี่ยว มีจำนวนบังคับของ ESIDs พื้นที่นั้นจะถูกเปลี่ยนเป็นสมนามแบบแบ่งใช้ โดยอัตโนมัติ ค่าที่ยอมรับได้อยู่ในช่วงตั้งแต่ 0 ถึง 4 KB (ต้องการ 256 MB ESIDs โดยประมาณในช่วง 1 TB)

`shm_1tb_unshared` VMO สามารถตั้งค่าได้ในแบบ ต่อกระบวนการโดยใช้ตัวแปรสภาวะแวดล้อม *"SHM_1TB_UNSHARED=" VMM_CNTRL* ค่าดีฟอลต์มีการตั้งค่าเป็น 256 ค่าที่ยอมรับได้อยู่ใน ช่วงตั้งแต่ 0 ถึง 4 KB ค่าดีฟอลต์มีการตั้งค่าอย่างระมัดระวัง (ต้องการ ประชากรสูงสุดถึง 64 GB พื้นที่แอดเดรส) ก่อนการย้ายไปยังสมนาม 1 TB แบบไม่แบ่งใช้ จำนวน threshold มีการตั้งค่าเป็นเซกเมนต์ 256 MB ซึ่ง พื้นที่หน่วยความจำแบบแบ่งใช้ถูกเลื่อนขึ้นเพื่อใช้สมนาม 1 TB ค่าที่ต่ำกว่า ต้องใช้พื้นที่หน่วยความจำแบบแบ่งใช้อย่างระมัดระวังในการใช้สมนาม 1 TB ซึ่งอาจทำให้ segment look-aside buffer (SLB) misses ลดลงแต่ยังสามารถทำให้ page table entry (PTE) misses เพิ่มสูงขึ้นได้ด้วย ถ้ามีการกำหนดสมนาม พื้นที่หน่วยความจำแบบแบ่งใช้จำนวนมากซึ่งไม่ได้ใช้งานในกระบวนการต่างๆ

`esid_allocator` VMO tunable สามารถตั้งค่าได้ในแบบ ต่อกระบวนการโดยใช้ตัวแปรสภาวะแวดล้อม *"ESID_ALLOCATOR=" VMM_CNTRL* ค่าดีฟอลต์มีการตั้งค่าเป็น 0 สำหรับ AIX เวอร์ชัน 6.1 และ 1 สำหรับ AIX เวอร์ชัน 7.0 ค่าสามารถเป็น 0 หรือ 1 อย่างใดอย่างหนึ่ง เมื่อตั้งค่า เป็น 0 จะมีการเปิดใช้งานตัวจัดสรรเก่าของการติดตั้งทางอ้อม มิฉะนั้น จะใช้นโยบายการจัดสรรพื้นที่แอดเดรสใหม่สำหรับการติดตั้งทางอ้อม ตัวจัดสรรพื้นที่แอดเดรสใหม่นี้แนบการจัดสรรทางอ้อมใดๆ (เช่น SHM และ MMAP) เข้ากับช่วงแอดเดรสใหม่ของ 0x0A00000000000000 – 0x0AFFFFFFFFFFFFFF ในพื้นที่แอดเดรสของแอ็พพลิเคชัน ตัวจัดสรร ใช้การจัดสรรให้เกิดประโยชน์สูงสุดเพื่อนำเสนอโอกาสเป็นไปได้ที่ดีที่สุด ของการเลื่อนขั้นสมนาม 1 TB การใช้ให้เกิดประโยชน์สูงสุดดังกล่าวอาจส่งผลให้พื้นที่ แอดเดรสเกิด "ช่องโหว่" ซึ่งถือว่าเป็นสภาพปกติเมื่อใช้การติดตั้งทางอ้อม การติดตั้งทางตรงมีการทำในช่วง 0x0700000000000000 – 0x07FFFFFFFFFFFFFF ดัง

นั้นจึงรักษาความเข้ากันได้กับเวอร์ชันก่อนหน้านี้ ในบางกรณี ที่นโยบายการจัดสรรใหม่นี้สร้างปัญหาความเข้ากันได้กับรูปแบบไบนารี สามารถเรียกคืนลักษณะการทำงานของตัวจัดสรรที่ใช้กันมาได้โดยตั้งค่า tunable เป็น 0

พื้นที่หน่วยความจำแบบแบ่งใช้ที่ไม่มีสิทธิเลื่อนขั้นเป็นสมนามแบบ แบ่งใช้จะถูกจัดกลุ่มไว้ในพื้นที่ 1 TB ในกลุ่มของพื้นที่หน่วยความจำแบบแบ่งใช้ ในพื้นที่ 1 TB ของเนื้อที่แอดเดรสของแอ็พพลิเคชัน ถ้าแอ็พพลิเคชันเกินค่าขีดจำกัด 256 MB เซกเมนต์ แอ็พพลิเคชันจะถูกเลื่อนขั้นไปใช้นามแฝง1 TB แบบไม่แบ่งใช้ในแอ็พพลิเคชันที่หน่วยความจำแบบแบ่งใช้มักถูกเชื่อมต่อและถอดออก ค่าที่ต่ำลงของขีดจำกัดของนามแฝงแบบไม่แบ่งใช้จะทำประสิทธิภาพลดลง

เพื่อหลีกเลี่ยงการทำลายพื้นที่ชื่อสภาวะแวดล้อม tunables สภาวะแวดล้อม ทั้งหมดจะมีการใช้ภายใต้ tunable `VMM_CNTRL` ต้นแบบ Tunable ต้นแบบมีการระบุโดยมีสัญลักษณ์@ แบ่ง ระหว่างคำสั่ง ตัวอย่างสำหรับการใช้ `VMM_CNTRL` คือ:

```
VMM_CNTRL=SHM_1TB_UNSHARED=32@SHM_1TB_SHARED=5
```

ค่าติดตั้งตัวแปรสภาวะแวดล้อมทั้งหมดมีการสืบทอดโดยชายน์บน `fork()` และเริ่มต้นเป็นค่าดีฟอลต์ของระบบ ที่ `exec()` แอ็พพลิเคชัน 32 บิตทั้งหมดไม่ได้รับผลกระทบจาก การเปลี่ยนแปลง VMO tunable หรือตัวแปรสภาวะแวดล้อม

VMO tunables และตัวแปรสภาวะแวดล้อมทั้งหมดมีคำสั่ง **vm_pattr** แบบแอนะล็อก ข้อยกเว้นคือ `esid_allocator` tunable Tunable นี้ ไม่มีอยู่ในอ็อพชัน `vm_pattr` เพื่อหลีกเลี่ยงสถานการณ์ ที่บางส่วนของพื้นที่แอดเดรสหน่วยความจำแบบแบ่งใช้ถูกจัดสรรก่อน การรันคำสั่ง

ส่วนสนับสนุนความเกี่ยวข้องของหน่วยความจำ AIX

AIX จัดเตรียมความสามารถในการจัดสรรหน่วยความจำสำหรับการประมวลผลจากโมดูลที่มีตัวประมวลผล ซึ่งเป็นสาเหตุของข้อบกพร่องของเพจ คุณสามารถใช้ความสามารถนี้ได้ หากเปิดใช้งานส่วนสนับสนุนความเกี่ยวข้องของหน่วยความจำบนระบบของคุณ และโดยการตั้งค่าตัวแปรสภาวะแวดล้อม *MEMORY_AFFINITY* ความเกี่ยวข้องของหน่วยความจำมีการเปิดใช้งานโดยดีฟอลต์ แต่คุณสามารถปิดใช้งานได้

ระบบฮาร์ดแวร์ IBM POWER processor-based platform SMP มีโมดูลที่มีความสามารถในการสนับสนุนชิปตัวประมวลผลเดี่ยว คู่ หรือจำนวนมาก ซึ่งขึ้นอยู่กับแต่ละระบบ แต่ละโมดูลเหล่านี้จะมีตัวประมวลผลจำนวนมาก และหน่วยความจำระบบจะพ่วงต่อกับโมดูลเหล่านี้ ขณะที่ตัวประมวลผลใดๆ สามารถเข้าถึงหน่วยความจำทั้งหมดในระบบได้ ตัวประมวลผลจะมีการเข้าถึงได้เร็วกว่า และมีแบนด์วิธที่สูงกว่า เมื่อกำหนดแอดเดรสหน่วยความจำ ที่พ่วงต่อกับโมดูลที่เป็นเจ้าของแทนหน่วยความจำที่พ่วงต่อกับโมดูลอื่นๆ ในระบบ

เมื่อเปิดใช้งานความเกี่ยวข้องของหน่วยความจำ แต่ละโมดูลจะมี vmpool เป็นของตนเอง ซึ่งจะมีพูลหน่วยความจำตั้งแต่หนึ่งกลุ่มขึ้นไป แต่ละพูลหน่วยความจำจะมี daemon การแทนที่เพจของตนเอง นั่นคือ **lrud** จำนวนของหน่วยความจำในแต่ละพูลจะอ้างอิงตามจำนวนของหน่วยความจำ ที่พร้อมใช้งานในโมดูล หรือได้จัดสรรให้กับ VMM โดยเลเยอร์ hypervisor

ถ้าคุณกำลังใช้ระบบปฏิบัติการ AIX ที่ความเกี่ยวข้องของหน่วยความจำ ปิดใช้งาน จำนวนของพูลหน่วยความจำขึ้นอยู่กับจำนวนของ หน่วยความจำและจำนวน CPUs ในระบบ

เมื่อต้องการปิดใช้งานการสนับสนุนความเกี่ยวข้องของหน่วยความจำบน AIX คุณสามารถใช้คำสั่ง **vmo** ต่อไปนี้:

```
vmo -o memory_affinity=0
```

หมายเหตุ: `bosboot` และ `reboot` จำเป็นต้องมีหากต้องการให้เกิดผลบังคับใช้
ค่าดีฟอลต์คือ 1 ซึ่งหมายความว่า ได้เปิดใช้งานส่วนสนับสนุนความเกี่ยวข้องของหน่วยความจำแล้ว

การเปิดใช้งานส่วนสนับสนุนความเกี่ยวข้องของหน่วยความจำจะแจ้งให้ระบบปฏิบัติการ จัดการโครงสร้างข้อมูลตามขอบเขตของโมดูล นโยบายการจัดสรรหน่วยความจำที่เป็นค่าดีฟอลต์ จะเปลี่ยนไปตาม MCM หากต้องการขอรับการจัดสรรหน่วยความจำ MCM โลคัลเป็นพิเศษ แอ็พพลิเคชันต้องเอ็กซ์พอร์ตตัวแปรสภาวะแวดล้อม MEMORY_AFFINITY ดังต่อไปนี้:

```
MEMORY_AFFINITY=MCM
```

ลักษณะการทำงานนี้จะถูกกระจาย ระหว่างการแบ่งแยก อย่างไรก็ตาม สำหรับลักษณะการทำงานนี้ที่ต้องการให้คงอยู่ระหว่างการเรียกฟังก์ชัน **exec** ตัวแปรต้องมีอยู่ในสตริงสภาวะแวดล้อม ที่ส่งผ่านไปยังการเรียกฟังก์ชัน **exec**

ข้อมูลที่เกี่ยวข้อง

คำสั่ง **vmo** และ “การปรับการเปลี่ยนหน้า VMM” ในหน้า 166

คำสั่งหรือรูทีนย่อย **bindprocessor**

แอ็ตทริบิวต์ WLM Class และ Resource Set

ผลกระทบต่อประสิทธิภาพจากการจัดสรรหน่วยความจำ MCM แบบโลคัล

ผลกระทบจากการจัดสรรหน่วยความจำ MCM แบบโลคัลบนแอ็พพลิเคชันเฉพาะ เป็นสิ่งที่คาดการณ์ได้ยาก บางแอ็พพลิเคชันอาจไม่ได้รับผลกระทบ บางแอ็พพลิเคชันอาจดีขึ้น และบางแอ็พพลิเคชันอาจแย่ลง

แอ็พพลิเคชันส่วนใหญ่ต้องอาศัยตัวประมวลผลเพื่อให้ได้รับประโยชน์ด้าน ประสิทธิภาพจาก affinity หน่วยความจำ สิ่งที่ต้องทำคือการป้องกันไม่ให้ AIX dispatcher ย้ายแอ็พพลิเคชันไปยังตัวประมวลผลใน MCMs อื่นในขณะที่แอ็พพลิเคชัน กำลังดำเนินการ

วิธีที่น่าจะเป็นไปได้มากที่สุดในการได้รับประโยชน์จาก affinity หน่วยความจำคือ การจำกัดให้แอ็พพลิเคชันรันเฉพาะบนตัวประมวลผลที่มีอยู่ใน MCM เดียวเท่านั้น ซึ่งสามารถทำได้โดยใช้คำสั่ง **bindprocessor** และฟังก์ชัน **bindprocessor( )** และยังสามารถทำโดยใช้คำสั่งและการบริการ **resource set** affinity ได้เช่นกัน

เมื่อแอ็พพลิเคชันต้องการตัวประมวลผลมากกว่าที่มีอยู่ใน MCM เดียว ประโยชน์ของประสิทธิภาพจากการใช้ affinity หน่วยความจำขึ้นอยู่กับรูปแบบ การจัดสรรและการเข้าถึงหน่วยความจำของเธรดต่างๆ ในแอ็พพลิเคชัน แอ็พพลิเคชันที่มีเธรดซึ่งจัดสรรและอ้างอิงพื้นที่เฉพาะทีละรายการ อาจเห็นประสิทธิภาพที่พัฒนาขึ้น แอ็พพลิเคชันที่แบ่งใช้หน่วยความจำระหว่างเธรด ทั้งหมดมีโอกาสมากที่จะได้รับประสิทธิภาพด้อยลงจาก affinity หน่วยความจำ

การกำหนดตำแหน่งหน่วยความจำด้วยคำสั่ง vmo

คุณสามารถจัดสรรหน่วยความจำผู้ใช้ด้วยพารามิเตอร์ของคำสั่ง **vmo** คุณยังสามารถพิจารณาว่า คุณต้องการใช้นโยบายการกำหนดตารางเวลาในการสัมผัสครั้งแรก หรือนโยบายการกำหนดตารางเวลาแบบ round-robin

ด้วยนโยบายการกำหนดตารางเวลาในการสัมผัสครั้งแรก หน่วยความจำจะถูกจัดสรรไว้จากชิปโมดูล ซึ่งเธรดต้องรัน เมื่อสัมผัสในครั้งแรกซึ่งเซ็กเมนต์หน่วยความจำ ที่เป็นข้อบกพร่องของเพจแรก ด้วยนโยบายการกำหนดตารางเวลา round-robin ซึ่งเป็นค่าดีฟอลต์สำหรับชนิดของหน่วยความ การจัดสรรหน่วยความจำที่ strip ระหว่าง vmpools แต่ละค่า

พารามิเตอร์ต่อไปนี้ของคำสั่ง **vmo** ที่ควบคุมการกำหนดตำแหน่งของหน่วยความจำผู้ใช้ และสามารถมีค่า 1 ซึ่งมีความหมายถึงนโยบายการกำหนดตารางเวลาสัมผัสในครั้งแรก หรือ 2 หมายถึงนโยบายการกำหนดตารางเวลาแบบ round-robin:

memplace_data

พารามิเตอร์นี้ระบุการกำหนดตำแหน่งหน่วยความจำสำหรับชนิดของข้อมูลต่อไปนี้ :

- ข้อมูลของความสามารถในการเรียกทำงานหลักที่ถูกกำหนดค่าเริ่มต้นหรือไม่ได้กำหนดค่าเริ่มต้น
- ข้อมูลเช็กเมนต์ฮีป
- ข้อมูลไลบรารีที่แบ่งใช้
- ข้อมูลของอ็อบเจ็กต์โมดูลที่ถูกโหลดที่รันไทม์

ค่าดีฟอลต์สำหรับพารามิเตอร์นี้คือ 2

memplace_mapped_file

พารามิเตอร์นี้ระบุการกำหนดตำแหน่งหน่วยความจำสำหรับไฟล์ที่ถูกแม็พในพื้นที่แอดเดรสของการประมวลผล เช่น ฟังก์ชัน **shmat( )** และฟังก์ชัน **mmap( )** ค่าดีฟอลต์สำหรับพารามิเตอร์นี้คือ 2

memplace_shm_anonymous

พารามิเตอร์นี้ระบุการกำหนดหน่วยความจำสำหรับหน่วยความที่แบ่งใช้แบบอะซิงโครนัส ที่ทำหน้าที่เป็นหน่วยเก็บหน่วยความจำใช้งานที่สร้างโดยการเรียกฟังก์ชัน **shmget( )** หรือฟังก์ชัน **mmap( )** หน่วยความจำสามารถเข้าถึงได้โดยการสร้างการประมวลผล หรือการสืบทอดและไม่เชื่อมโยงกับชื่อหรือคีย์ ค่าดีฟอลต์สำหรับพารามิเตอร์นี้คือ 2

memplace_shm_named

พารามิเตอร์นี้ระบุการกำหนดตำแหน่งหน่วยความจำสำหรับหน่วยความจำที่แบ่งใช้ที่มีชื่อแล้ว ซึ่งทำหน้าที่เป็นหน่วยเก็บหน่วยความจำที่ใช้งานซึ่งสร้างโดยการเรียกฟังก์ชัน **shmget( )** หรือฟังก์ชัน **shm_open( )** ซึ่งเชื่อมโยงกับชื่อหรือคีย์ ที่อนุญาตให้มีการประมวลผลมากกว่าหนึ่งกระบวนการเพื่อเข้าถึงได้อย่างพร้อมกัน ค่าดีฟอลต์ สำหรับพารามิเตอร์นี้คือ 2

memplace_stack

พารามิเตอร์นี้ระบุการกำหนดตำแหน่งหน่วยความจำสำหรับโปรแกรมสแต็ก ค่าดีฟอลต์ สำหรับพารามิเตอร์นี้คือ 2

memplace_text

พารามิเตอร์นี้ระบุการกำหนดตำแหน่งหน่วยความจำสำหรับข้อความแอ็พพลิเคชัน ของความสามารถในการเรียกทำงานหลัก แต่ไม่ใช่สำหรับการพึ่งพา ค่าดีฟอลต์ สำหรับพารามิเตอร์นี้คือ 2

memplace_unmapped_file

พารามิเตอร์นี้ระบุการกำหนดตำแหน่งหน่วยความจำสำหรับการเข้าถึงไฟล์ที่ไม่ได้แม็พ เช่น ฟังก์ชัน **read( )** หรือ **write( )** ค่าดีฟอลต์ สำหรับพารามิเตอร์นี้คือ 2

การกำหนดตำแหน่งหน่วยความจำด้วยตัวแปรสภาวะแวดล้อม MEMORY_AFFINITY

ที่ระดับของการประมวลผล คุณสามารถปรับแต่งการกำหนดตำแหน่งของหน่วยความจำผู้ใช้ด้วยตัวแปรสภาวะแวดล้อม *MEMORY_AFFINITY* ซึ่งแทนที่การกำหนดตำแหน่งหน่วยความจำด้วยพารามิเตอร์ของคำสั่ง **vmo**

ตารางต่อไปนี้แสดงค่าที่เป็นไปได้สำหรับตัวแปรสภาวะแวดล้อม *MEMORY_AFFINITY* :

ค่า	ลักษณะการทำงาน
MCM	หน่วยความจำส่วนบุคคลคือหน่วยความจำโลคัล และหน่วยความจำที่แบ่งใช้คือโลคัล
SHM=RR	ทั้งระบบ System V และหน่วยความจำที่แบ่งใช้ Posix Real-Time จะถูก strip ระหว่าง MCM ให้ใช้อ็อบเจ็กต์หน่วยความจำที่แบ่งใช้ขนาด 4 KB และ อ็อบเจ็กต์หน่วยความจำที่แบ่งใช้แบบ large-page-backed ค่านี้จะใช้งานได้สำหรับเคอร์เนลแบบ 64 บิต และถ้าค่า MCM ยังคงถูกนิยามไว้
LRU=EARLY	LRU daemon จะเริ่มต้นบนหน่วยความจำโลคัลในทันทีที่ threshold มีค่าลดลง เช่น พารามิเตอร์ *minfree* มีค่าที่เข้าใกล้ซึ่งจะไม่รอให้พูลของระบบทั้งหมดมีค่าเข้าใกล้ threshold ที่มีค่าลดลงนั้น ค่านี้จะใช้งานได้สำหรับค่าที่ได้นิยามไว้

คุณสามารถตั้งค่าจำนวนมากสำหรับสภาวะแวดล้อม *MEMORY_AFFINITY* โดยแยกค่าแต่ละค่าด้วยเครื่องหมาย (@)

เพจขนาดใหญ่

วัตถุประสงค์หลักสำหรับการใช้เพจขนาดใหญ่คือ การปรับปรุงผลการทำงานของระบบ สำหรับแอ็พพลิเคชันแบบ high performance computing (HPC) หรือแอ็พพลิเคชันแบบ memory-access-intensive ที่ใช้จำนวนของหน่วยความจำเสมือนที่มีขนาดใหญ่ การปรับแรงในสตรีมของผลการทำงานของระบบ จากการลดลงของการทำรายการโดยรอบบัฟเฟอร์ (TLB) จะหายไป เนื่องจากความสามารถของ TLB ในการแม็พช่วงของหน่วยความจำเสมือนที่ใหญ่กว่า

เพจขนาดใหญ่ยังปรับปรุงการดึงข้อมูลล่วงหน้าของหน่วยความจำโดยจำกัดความต้องการในการรีสตาร์ท การดำเนินการดึงข้อมูลล่วงหน้าบนขอบเขตที่มีขนาด 4 KB AIX สนับสนุนการใช้เพจขนาดใหญ่โดยแอ็พพลิคชันแบบ 32 บิต และ 64 บิต

สถาปัตยกรรมของเพจขนาดใหญ่ POWER4 ต้องการเพจเสมือนที่มีเซ็กเมนต์ขนาด 256 MB ที่ต้องเป็นขนาดเดียวกัน AIX สนับสนุนสถาปัตยกรรมนี้โดยใช้โหมดการผสมแบบจำลองการประมวลผล ซึ่งเซ็กเมนต์ในการประมวลผล รองรับเพจขนาด 4 KB ขณะที่เซ็กเมนต์อื่นรองรับเพจขนาด 16 MB แอ็พพลิเคชันสามารถร้องขอเซ็กเมนต์ฮีปหรือเซ็กเมนต์หน่วยความจำที่รองรับ เพจขนาดใหญ่ สำหรับข้อมูลโดยละเอียด โปรดอ้างถึง “คอนฟิกูเรชันแอ็พพลิเคชันสำหรับเพจขนาดใหญ่”

AIX รักษาพูลหน่วยความจำฟิสิคัลขนาด 4 KB และ 16 MB แยกจากกัน คุณสามารถระบุจำนวนของหน่วยความจำฟิสิคัลในพูลหน่วยความจำขนาด 16 MB โดยใช้คำสั่ง **vmo** พูลของเพจขนาดใหญ่เป็นแบบไดนามิก ดังนั้น จำนวนของหน่วยความจำฟิสิคัลที่ คุณระบุจะมีผลบังคับใช้ในทันที และไม่ต้องรีบูต ระบบ หน่วยความจำฟิสิคัลที่เหลืออยู่ จะรองรับเพจเสมือนขนาด 4 KB

AIX ใช้เพจขนาดใหญ่ เป็นหน่วยความจำที่ตรึงไว้ AIX ไม่ได้จัดเตรียมส่วนสนับสนุนการเพจสำหรับเพจที่มีขนาดใหญ่ ข้อมูลของแอ็พพลิเคชันที่รองรับเพจขนาดใหญ่ ยังคงอยู่ในหน่วยความจำฟิสิคัลจนกว่าแอ็พพลิเคชันจะเสร็จสิ้น กลไกการควบคุมการเข้าถึงการรักษาความปลอดภัยจะป้องกันแอ็พพลิเคชันที่ไม่ได้รับอนุญาต จากการใช้เพจขนาดใหญ่ หรือหน่วยความจำฟิสิคัลที่มีเพจขนาดใหญ่ กลไกการควบคุมการเข้าถึงการรักษาความปลอดภัย ยังปกป้องผู้ใช้ที่ไม่มีสิทธิการใช้งานจากการใช้เพจขนาดใหญ่ สำหรับแอ็พพลิเคชันของผู้ใช้ สำหรับ id ผู้ใช้ที่ไม่ใช่ผู้ใช้ราก คุณต้องเปิดใช้งานความสามารถ **CAP_BYPASS_RAC_VMM** ด้วยคำสั่ง **chuser** หากต้องการใช้เพจขนาดใหญ่ ตัวอย่างต่อไปนี้ สาธิตวิธีการให้สิทธิ **CAP_BYPASS_RAC_VMM** ที่เป็น superuser:

```
# chuser capabilities=CAP_BYPASS_RAC_VMM,CAP_PROPAGATE <user id>
```

คอนฟิกูเรชันแอ็พพลิเคชันสำหรับเพจขนาดใหญ่

มีหลายวิธีในการปรับแต่งแอ็พพลิเคชันให้ใช้เพจขนาดใหญ่

การใช้เพจขนาดใหญ่เพื่อรองรับข้อมูลและเซ็กเมนต์ฮีป:

คุณต้องพิจารณาข้อมูลเพจขนาดใหญ่ของแอ็พพลิเคชัน หรือการใช้ฮีปเมื่อคุณเรียกใช้งานแอ็พพลิเคชัน เนื่องจากแอ็พพลิเคชันไม่สามารถสับเปลี่ยนโหมดได้ หลังจากที่เริ่มต้นการเรียกใช้งานแล้ว การใช้เพจขนาดใหญ่จะสืบทอดโดย children process ของฟังก์ชัน **fork( )**

คุณสามารถปรับแต่งแอ็พพลิเคชันเพื่อร้องขอเพจขนาดใหญ่ที่รองรับข้อมูลโปรแกรมที่กำหนดค่าเริ่มต้นแล้ว ข้อมูลโปรแกรมที่ยังไม่ได้กำหนดค่าเริ่มต้น (BSS) และเซ็กเมนต์ฮีปด้วยเมธอดต่อไปนี้:

- "การทำเครื่องหมายไฟล์ที่ดำเนินการได้เพื่อร้องขอหน้าขนาดใหญ่"
- "การตั้งค่าตัวแปรสภาพแวดล้อมเพื่อร้องขอหน้าขนาดใหญ่"

คุณสามารถระบุได้ว่า คุณต้องการให้แอ็พพลิเคชันใช้เพจขนาดใหญ่สำหรับข้อมูล หรือเซ็กเมนต์ฮีปในโหมดต่อไปนี้ อย่างใดอย่างหนึ่ง:

- "โหมดคำแนะนำ" ในหน้า 182
- "โหมดบังคับ" ในหน้า 182

แอ็พพลิเคชันแบบ 32 บิตที่ใช้เพจขนาดใหญ่สำหรับข้อมูล และเซ็กเมนต์ฮีปจะใช้แบบจำลองการประมวลผลเพจขนาดใหญ่ที่มีขนาด 32 บิต เนื่องจากส่วนของการป้องกันเพจ ของเพจที่มีขนาดใหญ่ แบบจำลองการประมวลผลอื่นๆ จะใช้เพจที่มีขนาด 4 KB พร้อมกับแอ็ตทริบิวต์การป้องกันที่แตกต่างกัน ในเซ็กเมนต์เดียวกัน ซึ่งจะไม่ทำงาน เมื่อส่วนการป้องกันมีขนาด 16 MB

การทำเครื่องหมายไฟล์ที่ดำเนินการได้เพื่อร้องขอหน้าขนาดใหญ่:

ส่วนหัว XCOFF ในไฟล์ที่ดำเนินการได้มีแฟล็ก **blpdata** เพื่อบ่งชี้ว่าแอ็พพลิเคชันต้องการใช้หน้าขนาดใหญ่เพื่อสนับสนุนข้อมูล และ heap เซกเมนต์

เมื่อต้องการทำเครื่องหมายไฟล์ที่ดำเนินการได้เพื่อร้องขอหน้าขนาดใหญ่ ให้ใช้คำสั่งต่อไปนี้:

```
# ldedit -blpdata <filename>
```

ถ้าคุณ ตัดสินใจที่จะไม่ใช้หน้าขนาดใหญ่เพื่อสนับสนุนข้อมูลและ heap เซกเมนต์อีกต่อไป ให้ใช้คำสั่งต่อไปนี้เพื่อล้างแฟล็กหน้าขนาดใหญ่:

```
# ldedit -bnolpdata <filename>
```

คุณยัง สามารถตั้งค่าอ็อพชัน **blpdata** เมื่อลิงก์และยึดโดยใช้คำสั่ง **cc**

การตั้งค่าตัวแปรสภาพแวดล้อมเพื่อร้องขอหน้าขนาดใหญ่:

คุณสามารถใช้ตัวแปรสภาพแวดล้อม *LDR_CNTRL* เพื่อตั้งค่าคอนฟิกแอ็พพลิเคชันให้ใช้หน้าขนาดใหญ่สำหรับข้อมูล และ heap เซกเมนต์ของแอ็พพลิเคชัน ตัวแปรสภาพแวดล้อมมีความสำคัญมากกว่าแฟล็ก **blpdata** ใน ไฟล์ที่ดำเนินการได้

อ็อพชันที่มีอยู่สำหรับตัวแปรสภาพแวดล้อม *LDR_CNTRL* มีดังต่อไปนี้:

- อ็อพชัน **LDR_CNTRL=LARGE_PAGE_DATA=Y** ระบุว่า แอ็พพลิเคชันที่ดำเนินการควรใช้หน้าขนาดใหญ่สำหรับข้อมูลและ heap เซกเมนต์ของแอ็พพลิเคชัน ซึ่งเหมือนกับการทำเครื่องหมายไฟล์ที่ดำเนินการได้ให้ใช้หน้าขนาดใหญ่
- อ็อพชัน **LDR_CNTRL=LARGE_PAGE_DATA=N** ระบุว่า แอ็พพลิเคชันที่ดำเนินการไม่ควรใช้หน้าขนาดใหญ่สำหรับข้อมูลและ heap เซกเมนต์ของแอ็พพลิเคชัน ซึ่งยกเลิกค่าติดตั้งในไฟล์ที่ดำเนินการได้ซึ่งทำเครื่องหมายให้ใช้หน้าขนาดใหญ่

- อ็อพชัน **LDR_CNTRL=LARGE_PAGE_DATA=M** ระบุว่า แอ็พพลิเคชันที่ดำเนินการควรใช้หน้าขนาดใหญ่ในโหมดบังคับสำหรับข้อมูลและ heap เซกเมนต์ของแอ็พพลิเคชัน

หมายเหตุ: ตั้งค่าตัวแปรสภาพแวดล้อมหน้าขนาดใหญ่เฉพาะสำหรับ แอ็พพลิเคชันเฉพาะที่อาจได้รับประโยชน์จากการใช้หน้าขนาดใหญ่เท่านั้น มิฉะนั้น คุณอาจพบประสิทธิภาพ ที่ด้อยลงบางอย่างบนระบบของคุณ

โหมดคำแนะนำ:

ในโหมดคำแนะนำ มีความเป็นไปได้สำหรับแอ็พพลิเคชันที่มีเซ็กเมนต์ของฮีปบางส่วนที่สนับสนุน เพจขนาดใหญ่ และเซ็กเมนต์บางส่วนที่สนับสนุนเพจที่มีขนาด 4 KB เพจที่มีขนาด 4 KB จะรองรับข้อมูลหรือเซ็กเมนต์ฮีป เมื่อมีเพจขนาดไม่เพียงพอ ที่พร้อมใช้งานเพื่อรองรับเซ็กเมนต์

ในโหมดคำแนะนำ แอ็พพลิเคชันใช้เพจขนาดใหญ่หากเป็นไปได้ ซึ่งขึ้นอยู่กับเงื่อนไขดังต่อไปนี้:
- userid ที่ได้รับการสิทธิในการใช้เพจขนาดใหญ่
- ฮาร์ดแวร์ของระบบมีคุณลักษณะเชิงสถาปัตยกรรมของเพจขนาดใหญ่
- คุณได้กำหนดพูลของหน่วยความจำแบบเพจขนาดใหญ่
- มีเพจที่เพียงพออยู่ในพูลของหน่วยความจำแบบเพจขนาดใหญ่เพื่อรองรับเซ็กเมนต์ ที่มีเพจขนาดใหญ่

ถ้าไม่มีการปฏิบัติตามเงื่อนไขข้างต้นใดๆ แล้ว ข้อมูลของแอ็พพลิเคชัน และเซ็กเมนต์ฮีปจะสนับสนุนเพจที่มีขนาด 4 KB

ไฟล์เรียกทำงานซึ่งถูกทำเครื่องหมายเพื่อใช้เพจขนาดใหญ่จะรันอยู่ในโหมดคำแนะนำ

โหมดบังคับ:

ในโหมดบังคับ ถ้าแอ็พพลิเคชันร้องขอเซ็กเมนต์ฮีป และมีเพจขนาดใหญ่ไม่เพียงพอในต่อคำร้องขอแล้ว คำร้องขอการจัดสรรจะล้มเหลว ซึ่งอาจทำให้แอ็พพลิเคชันยกเลิกด้วยข้อผิดพลาด

ถ้าคุณใช้โหมดบังคับ คุณต้องมอนิเตอร์ขนาดของพูลของเพจขนาดใหญ่ และตรวจสอบให้แน่ใจว่า พูลนั้นไม่ได้ทำงานโดยมีเพจขนาดใหญ่ที่ไม่เพียงพอ มิฉะนั้น แอ็พพลิเคชันเพจขนาดใหญ่ในโหมดบังคับของคุณจะล้มเหลว

การใช้เพจขนาดใหญ่เพื่อรองรับเซ็กเมนต์หน่วยความจำที่แบ่งใช้:

หากต้องการรองรับเซ็กเมนต์หน่วยความจำที่แบ่งใช้ของแอ็พพลิเคชันที่มีเพจขนาดใหญ่ คุณต้องระบุแฟล็ก **SHM_LGPAGE** และ **SHM_PIN** ในฟังก์ชัน **shmget( )** ถ้าเพจขนาดใหญ่ไม่พร้อมใช้งาน เพจขนาด 4 KB จะรองรับเซ็กเมนต์หน่วยความจำที่แบ่งใช้

หน่วยความจำฟิสิคัลที่รองรับเพจขนาดใหญ่ที่มีหน่วยความจำที่แบ่งใช้ และข้อมูลเพจขนาดใหญ่และเซ็กเมนต์ฮีปจะมาพร้อมกับเพจขนาดใหญ่ที่มีพูลหน่วยความจำฟิสิคัล คุณต้องมั่นใจว่า เพจขนาดใหญ่ที่มีพูลหน่วยความจำฟิสิคัลมีเพจขนาดใหญ่พอสำหรับหน่วยความจำที่แบ่งใช้และข้อมูลและฮีปที่มีการใช้เพจขนาดใหญ่

การตั้งค่าคอนฟิกระบบสำหรับหน้าขนาดใหญ่

คุณต้องตั้งค่าคอนฟิกระบบของคุณเพื่อใช้หน้าขนาดใหญ่ และคุณยังต้อง ระบุจำนวนของหน่วยความจำฟิสิคัลที่คุณต้องการจัดสรรเพื่อสนับสนุน หน้าขนาดใหญ่ด้วย

โดยดีฟอลต์ ระบบไม่มี หน่วยความจำที่จัดสรรให้กับพูลหน่วยความจำฟิสิคัลของหน้าขนาดใหญ่ คุณสามารถใช้คำสั่ง **vmo** เพื่อตั้งค่าคอนฟิกขนาดของพูลหน่วยความจำฟิสิคัลหน้าขนาดใหญ่ ตัวอย่างต่อไปนี้ จัดสรรหน่วยความจำ 1 GB ให้กับพูลหน่วยความจำฟิสิคัลของหน้าขนาดใหญ่:

```
# vmo -r -o lgpg_regions=64 -o lgpg_size=16777216
```

เพื่อใช้ หน้าขนาดใหญ่สำหรับหน่วยความจำแบบแบ่งใช้ คุณต้องเปิดใช้งานการเรียกระบบ **SHM_PIN shmget( )** ด้วยคำสั่งต่อไปนี้ ซึ่งจะติดอยู่ถาวรในการรีบูตระบบ:

```
# vmo -p -o v_pinshm=1
```

เมื่อต้องการ ดูจำนวนหน้าขนาดใหญ่ที่ใช้อยู่บนระบบของคุณ ให้ใช้คำสั่ง **vmstat –l** ดังในตัวอย่างต่อไปนี้:

```
# vmstat -l

kthr     memory             page              faults        cpu      large-page

----- ----------- ------------------------ ------------ ----------- ------------
 r  b   avm   fre  re  pi  po  fr   sr  cy  in   sy  cs us sy id wa   alp   flp
 2  1 52238 124523   0   0   0   0    0   0 142   41  73  0  3 97  0     16     16
```

จากตัวอย่างข้างบน คุณสามารถทราบว่ามีหน้าขนาดใหญ่ที่ใช้งานอยู่จำนวน 16 หน้า, `alp`, และหน้าขนาดใหญ่ที่ว่างจำนวน 16 หน้า, `flp`

ข้อควรพิจารณาสำหรับการใช้เพจขนาดใหญ่

ส่วนสนับสนุนเพจขนาดใหญ่คือวัตถุประสงค์พิเศษสำหรับคุณลักษณะการปรับปรุงผลการทำงาน และไม่แนะนำให้ใช้ทั่วไป หมายเหตุ ไม่ใช่แอ็พพลิเคชันทุกตัวที่ได้รับประโยชน์จากการใช้ เพจขนาดใหญ่ในความเป็นจริง บางแอ็พพลิเคชัน เช่น แอ็พพลิเคชันที่ดำเนินการกับจำนวนของฟังก์ชัน **fork( )** ที่มีขนาดใหญ่ และมีแนวโน้มในการลดระดับผลการทำงานลง ขณะที่ใช้เพจขนาดใหญ่

แทนที่จะใช้ตัวแปรสภาวะแวดล้อม *LDR_CNTRL* ให้พิจารณาการทำเครื่องหมายไฟล์เรียกทำงานเฉพาะเพื่อใช้เพจขนาดใหญ่ เนื่องจากมีข้อจำกัดในการใช้เพจขนาดใหญ่กับแอ็พพลิเคชันที่ระบุไว้ซึ่งมีข้อได้เปรียบจากการใช้เพจขนาดใหญ่

ถ้าคุณกำลังพิจารณาการใช้เพจขนาดใหญ่ ให้คิดถึงผลการทำงานโดยรวม ที่จะกระทบกับระบบของคุณ ขณะที่แอ็พพลิเคชันเฉพาะบางตัวอาจได้รับประโยชน์จากการใช้เพจขนาดใหญ่ คุณอาจมองเห็นการลดระดับผลการทำงานลงในผลการทำงานของระบบโดยรวม เนื่องจากการลดหน่วยเก็บของเพจขนาด 4 KB พร้อมใช้งานบนระบบนั้น ถ้าระบบของคุณมีหน่วยความจำฟิสิคัลที่เพียงพอ เช่น การลดจำนวนของเพจขนาด 4 KB ไม่ได้เป็นอุปสรรคต่อผลการทำงานของระบบ ดังนั้น คุณจึงอาจพิจารณาการใช้เพจขนาดใหญ่ได้

การสนับสนุนหลายขนาดหน้า

ตัวประมวลผล POWER5+ สนับสนุน ขนาดหน้าของหน่วยความเสมือนสี่ขนาดคือ: 4 KB, 64 KB, 16 MB, และ 16 GB นอกจากนี้ ตัวประมวลผล POWER6® ยังสนับสนุน การใช้หน้าขนาด 64 KB ในเซกเมนต์ที่มีขนาดหน้าพื้นฐาน 4 KB ด้วย AIX ใช้กระบวนการนี้เพื่อนำเสนอข้อดีของประสิทธิภาพหน้าขนาด 64 KB เมื่อมีประโยชน์ หรือการเปลี่ยนไปใช้หน้าขนาด 4 KB ในกรณีที่หน้า 64 KB อาจทำให้สิ้นเปลืองหน่วยความจำมากเกินไป เช่น เมื่อมีการจัดสรรหน้าขนาดดังกล่าวแต่แอ็พพลิเคชันไม่ได้ใช้งาน

การใช้ขนาดหน้าหน่วยความจำเสมือนที่ใหญ่ขึ้น เช่น ขนาด 64 KB สำหรับหน่วยความจำ ของแอ็พพลิเคชันสามารถพัฒนาประสิทธิภาพและผลผลิตของแอ็พพลิเคชันได้เป็นอย่างมาก เนื่องจากประสิทธิผลของฮาร์ดแวร์เชื่อมโยงกับขนาดหน้าที่ใหญ่ขึ้น การใช้ขนาดหน้า ที่ใหญ่ขึ้นสามารถลดเวลาแฝงฮาร์ดแวร์ของการแปลที่อยู่หน้าเสมือน เป็นที่อยู่หน้าฟิสิคัล เวลาแฝงที่ลดลงนี้เป็นผลมาจากการพัฒนา ประสิทธิผลของแคชการแปลฮาร์ดแวร์ เช่น translation lookaside buffer (TLB) ของตัวประมวลผล เนื่องจากแคชการแปลฮาร์ดแวร์ มีจำนวนรายการที่จำกัด ดังนั้นการใช้ขนาดหน้าที่ใหญ่ขึ้นจึงเพิ่มจำนวนของ หน่วยความจำเสมือนที่สามารถแปลได้โดยแต่ละรายการในแคช ซึ่งส่งผลให้จำนวนหน่วยความจำที่แอ็พพลิเคชันสามารถเข้าถึงได้เพิ่มมากขึ้น โดยไม่เกิดความล่าช้าในการแปลฮาร์ดแวร์

ในขณะที่หน้า 16 MB และ 16 GB มีไว้สำหรับสภาพแวดล้อมประสิทธิภาพสูง มากเท่านั้น แต่หน้าขนาด 64 KB ถือเป็นหน้าอเนกประสงค์ที่ใช้งานได้ทั่วไป และเวิร์คโหลดส่วนใหญ่จะได้รับประโยชน์จากการใช้หน้า 64 KB แทนหน้าขนาด 4 KB

ขนาดหน้าที่สนับสนุนตามชนิดตัวประมวลผล

ใช้คำสั่ง **pagesize** ที่มีอ็อพชัน **-a** เพื่อกำหนดขนาดหน้าทั้งหมดที่สนับสนุนโดย AIX บน ระบบเฉพาะ

AIX 6.1 และใหม่กว่า สนับสนุนเซกเมนต์ที่มีขนาดเพจสองขนาด: 4 KB และ 64 KB โดยค่าดีฟอลต์ กระบวนการใช้เซกเมนต์ขนาดหน้า ผันแปรเหล่านี้ ซึ่งอาจถูกแทนที่ได้โดยกลไกการเลือกขนาดหน้า ที่มีอยู่

ตารางที่ 2. การสนับสนุนขนาดหน้าโดย AIX และฮาร์ดแวร์ System p อื่น

ขนาดหน้า	ฮาร์ดแวร์ที่ต้องการ	ต้องการการตั้งค่าคอนฟิกผู้ใช้	จำกัด
4 KB	ทั้งหมด	ไม่	ไม่
64 KB	POWER5+ หรือ ใหม่กว่า	ไม่	ไม่
16 MB	POWER4 หรือ ใหม่กว่า	ใช่	ใช่
16 GB	POWER5+ หรือ ใหม่กว่า	ใช่	ใช่

ตารางที่ 3. ขนาดหน้าเซกเมนต์ที่สนับสนุน

ขนาดหน้าพื้นฐานของเซกเมนต์	ขนาดหน้าที่สนับสนุน	ฮาร์ดแวร์ที่ต้องการต่ำสุด
4 KB	4 KB / 64 KB	POWER6
64 KB	64 KB	POWER5+
16 MB	16 MB	POWER4
16 GB	16 GB	POWER5+

ในเวอร์ชันก่อนหน้านี้ทั้งหมดของ AIX 4 KB เป็นขนาดหน้า ดีฟอลต์ กระบวนจะยังคงใช้หน้าขนาด 4 KB ต่อไป ยกเว้นว่าผู้ใช้ร้องขอที่จะใช้ขนาดหน้า อื่นโดยเฉพาะ

ส่วนสนับสนุนเพจที่มีขนาด 64 KB

เนื่องจากเพจที่มีขนาด 64 KB ง่ายต่อการใช้ และเนื่องจากมีการคาดการณ์ขนาดของเพจไว้ว่า แอ็พพลิเคชันจำนวนมากจะมองเห็นข้อได้เปรียบของผลการทำงาน ขณะที่ใช้เพจที่มีขนาด 64 KB แทนเพจที่มีขนาด 4 KB AIX จะมีส่วนสนับสนุนสำหรับเพจที่มีขนาด 64 KB

ไม่มีการเปลี่ยนแปลงคอนฟิกูเรชันระบบที่จำเป็นต่อการเปิดใช้งานระบบ เพื่อใช้เพจที่มีขนาด 64 KB บนระบบที่สนับสนุนเพจที่มีขนาด 64 KB เคอร์เนล AIX จะปรับแต่งระบบโดยอัตโนมัติเพื่อใช้เพจนั้น เพจที่มีขนาด 64 KB จะสามารถเพจได้โดย

สมบูรณ์ และขนาดพูลของกรอบเพจขนาด 64 KB บนระบบจะเป็นแบบไดนามิก และจะถูกจัดการโดย AIX ทั้งหมด AIX จะผันแปรตามจำนวนของกรอบเพจขนาด 4 KB และ 64 KB บนระบบเพื่อให้ตรงกับความต้องการ ขนาดของเพจที่แตกต่างกัน ทั้งคำสั่ง **svmon** และ **vmstat** สามารถนำมาใช้เพื่อมอนิเตอร์จำนวนของกรอบเพจที่มีขนาด 4 KB และ 64 KB บนระบบได้

ส่วนสนับสนุนตัวแปรขนาดเพจแบบไดนามิก

ตัวประมวลผลก่อนPOWER6 จะสนับสนุขนาดเพจเดี่ยวต่อเซ็กเมนต์ ผู้ดูแลระบบหรือผู้ใช้ ต้องเลือกขนาดเพจที่เหมาะสมที่สุดสำหรับแอ็พพลิเคชันที่ระบุเฉพาะตาม รอยทางของหน่วยความจำ

การเลือกเพจขนาด 4 KB จะไม่ได้ใช้ประโยชน์จากจำนวนหน่วยความจำที่น้อยที่สุด เนื่องจากเพจที่มีขนาด 4 KB เหล่านั้นที่อ้างอิงถึงถูกนำมาใช้จริง ขนาดเพจที่มีขนาดใหญ่กว่า สามารถทิ้งหน่วยความจำจำนวนมากได้ (จัดสรรแล้ว หรือไม่เคยใช้) ขึ้นอยู่กับตำแหน่งชุดการทำงาน และได้รับการส่งเสริมผลการทำงานแบบเสมือนให้กับการแปลแบบฟิสิคัล ที่ต้องใช้ นอกจากนี้ ขนาดเพจที่ใหญ่กว่า 4 KB ต้องการแทรกแซงผู้ใช้เพื่อให้เลือกขนาดเพจที่ระบุเฉพาะ

POWER6 แนะนำแนวคิด ของขนาดเพจแบบผสมภายในเซ็กเมนต์เดี่ยว สถาปัตยกรรมสนับสนุนการเปลี่ยนลำดับของขนาดเพจต่างๆ อย่างไรก็ตาม POWER6 จะสนับสนุนเฉพาะขนาดเพจที่ผสมกันระหว่าง 4 KB และ 64 KB AIX 6.1 ใช้ประโยชน์จากความสามารถของฮาร์ดแวร์ใหม่นี้เพื่อรวมการใช้หน่วยความจำที่สงวนไว้ขนาด 4 KB ในส่วนของหน่วยความจำที่อ้างอิงกับผลการทำงานที่ได้ประโยชน์ของขนาดเพจ 64 KB ในส่วนของหน่วยความจำที่อ้างอิง และใช้แบบอัตโนมัติโดยไม่มีการแทรกแซงผู้ใช้ คุณลักษณะนี้ AIX ถูกอ้างถึงเป็น Dynamic Variable Page Size Support (VPSS) เพื่อหลีกเลี่ยงปัญหาความเข้ากันได้ย้อนกลับ VPSS จะปิดใช้งานในเซ็กเมนต์ที่มีขนาดเพจที่เลือกไว้โดยผู้ใช้ (โปรดดู ส่วนสนับสนุนแอ็พพลิเคชันที่มีขนาดเพจจำนวนมาก)

ค่ากำหนดดีฟอลต์สำหรับตัวแปรเซ็กเมนต์ขนาดเพจ 4 KB และการแปล 4 KB ในขนาด 64 KB และส่วนการจัดเรียงจนกระทั่งเพจ 16 4 KB ทั้งหมดจะถูกอ้างอิงถึง หากเพจ 16 เพจทั้งหมดถูกอ้างถึง การตรวจสอบจะถูกทำเพื่อให้แน่ใจว่า เพจเหล่านั้นมีสถานะเดียวกัน (เช่น การป้องกันการอ่าน/เขียนเพจ การป้องกันที่ไม่สามารถเรียกทำงานได้ การป้องกันหน่วยเก็บคีย์ และไม่ได้อยู่ในสถานะ I/O) ถ้าเพจเหล่านั้นทำสิ่งนี้ การแปลขนาด 4 KB จะถูกลบออก และแทนที่ด้วยการแปลขนาด 64 KB

การแปลขนาด 64 KB จะถูกใช้ตราบนานเท่าที่เพจขนาด 16 KB ทั้งหมดยังคงมี สถานะเดียวกัน หากหนึ่งในเพจเหล่านั้นเปลี่ยนสถานะ (ตัวอย่างเช่น ผ่านรูทีนย่อย **mprotect** หรือการนำ LRU มาใช้ เพจเหล่านั้น จะลดระดับไปเป็นเพจขนาด 4 KB จนกว่าจะกู้คืนสถานะเดิมได้

บางแอ็พพลิเคชันอาจต้องการใช้ขนาดเพจที่ใหญ่กว่า เมื่อส่วนที่มีขนาด 64 KB จะไม่ได้ถูกอ้างอิงทั้งหมด ปัจจัยเกี่ยวกับขนาดของเพจ (PSPA) สามารถนำมาใช้เพื่อลดหน่วยความจำที่อ้างอิงถึงความต้องการที่จุดของกลุ่มของเพจ ที่มีขนาด 4 KB จะถูกโปรโมตไปเป็นขนาดเพจ 64 KB PSPA สามารถตั้งค่าระบบทั้งหมดผ่าน **vmm_default_pspa vmo** ที่สามารถปรับแต่งได้ สำหรับการประมวลผลเฉพาะผ่านคำสั่ง **vm_pattr**

เนื่องจากขนาดของเพจ 64 KB ที่ได้รับการสนับสนุน คำสั่ง **svmon** จะถูกอัพเพื่อรายงานตัวแปรการใช้ขนาดเพจ สำหรับข้อมูลเพิ่มเติมเกี่ยวคำสั่ง **vmo** โปรดดู *Commands Reference, Volume 6*

ขนาดหน้าสำหรับสภาพแวดล้อมประสิทธิภาพสูงมาก

นอกเหนือจากขนาดหน้า 4 KB และ 64 KB แล้ว AIX ยังสนับสนุน หน้า 16 MB ซึ่งเรียกอีกอย่างว่า *หน้าขนาดใหญ่* และหน้า 16 GB ซึ่งเรียกอีกอย่างว่า *หน้าขนาดใหญ่พิเศษ* ขนาดหน้าเหล่านี้มีไว้สำหรับการใช้เฉพาะในสภาพแวดล้อมประสิทธิภาพสูงเท่านั้น และ AIX จะไม่ตั้งค่าคอนฟิก ระบบให้ใช้ขนาดหน้าเหล่านี้โดยอัตโนมัติ

AIX ต้องมีการตั้งค่าคอนฟิกให้ใช้ ขนาดหน้าเหล่านี้ จำนวนหน้าของแต่ละขนาดหน้าเหล่านี้ต้องมีการ ตั้งค่าคอนฟิกด้วย AIX สามารถ เปลี่ยนจำนวนของหน้า 16 MB หรือ 16 GB ที่ตั้งค่าคอนฟิกไว้ ตามความต้องการได้โดยอัตโนมัติ

หน่วยความจำที่จัดสรรให้กับหน้าขนาดใหญ่ 16 MB สามารถใช้สำหรับหน้าขนาดใหญ่ 16 MB เท่านั้น และหน่วยความจำที่จัดสรรให้กับหน้าขนาดใหญ่พิเศษ 16 GB สามารถใช้สำหรับหน้าขนาดใหญ่พิเศษ 16 GB เท่านั้น ดังนั้น หน้าที่มีขนาดดังกล่าวนี้จึงควรมีการตั้งค่าคอนฟิกเฉพาะในสภาพ แวดล้อมประสิทธิภาพสูงเท่านั้น นอกจากนี้ การใช้หน้าขนาด 16 MB และ 16 GB ยังมีข้อจำกัด: เพื่อ จัดสรรหน้าที่มีขนาดเหล่านี้ ผู้ใช้ต้องมีความสามารถ **CAP_BYPASS_RAC_VMM** และ **CAP_PROPAGATE,** หรือสิทธิ **ราก**

การปรับแต่งจำนวนของเพจขนาดใหญ่:

ใช้คำสั่ง **vmo** เพื่อปรับแต่งจำนวนของเพจขนาดใหญ่ที่มีขนาด 16 MB บนระบบ

ตัวอย่างต่อไปนี้จะจัดสรรเพจขนาดใหญ่ที่มีขนาด 1 GB ของเพจขนาด 16 MB:

```
# vmo -r -o lgpg_regions=64 -o lgpg_size=16777216
```

ในตัวอย่างข้างต้น การเปลี่ยนคอนฟิกูเรชันเพจขนาดใหญ่จะไม่ได้รับผลกระทบ จนกระทั่งคุณรันคำสั่ง **bosboot** และรีบูตระบบ บนระบบที่สนับสนุน dynamic logical partitioning (DLPAR) อ็อพชัน **-r** สามารถละเว้นได้จากคำสั่งข้างต้นเพื่อปรับแต่งเพจขนาดใหญ่ที่มีขนาด 16 MB แบบไดนามิก โดยไม่ต้องรีบูตระบบ

สำหรับข้อมูลเพิ่มเติมเกี่ยวกับเพจขนาดใหญ่ที่มีขนาด 16 MB โปรดดู “เพจขนาดใหญ่” ในหน้า 180

การปรับแต่งจำนวนของเพจขนาดใหญ่:

เพจขนาดใหญ่ต้องถูกปรับแต่งผ่าน Hardware Management Console (HMC) ของระบบ

1. สำหรับระบบที่ถูกจัดการให้ไปที่ **คุณสมบัติ** > **หน่วยความจำ** > **อ็อพชันขั้นสูง** > **แสดงรายละเอียด** เพื่อเปลี่ยนจำนวนของเพจขนาด 16 GB
2. กำหนดเพจขนาดใหญ่ 16 GB ให้กับพาร์ติชันโดยเปลี่ยนโปรไฟล์ของพาร์ติชัน

การสนับสนุนแอ็พพลิเคชันหลายขนาดหน้า

คุณสามารถระบุขนาดหน้าที่จะใช้สำหรับพื้นที่สี่ส่วนของพื้นที่ว่างที่อยู่ ของกระบวนการ 32-บิตหรือ 64-บิต

ขนาดหน้าเหล่านี้สามารถตั้งค่าคอนฟิกด้วย ตัวแปรสภาพแวดล้อม หรือด้วยค่าติดตั้งใน XCOFF ไบนารี ของแอ็พพลิเคชัน โดยใช้คำสั่ง **ldedit** หรือ **ld** ดังนี้:

พื้นที่	อ็อพชัน ld หรือ ldedit	ตัวแปรสภาพแวดล้อม LDR_CNTRL	คำอธิบาย
ข้อมูล	-bdatapsize	DATAPSIZE	เริ่มต้นข้อมูล, bss, heap
สแต็ค	-bstackpsize	STACKPSIZE	เริ่มต้นเธรดสแต็ค
ข้อความ	-btextpsize	TEXTPSIZE	ข้อความหลักที่ดำเนินการได้
หน่วยความจำแบบแบ่งใช้	ไม่มี	SHMPSIZE	หน่วยความจำแบบแบ่งใช้ที่จัดสรรโดยกระบวนการ

คุณสามารถระบุขนาดหน้าที่แตกต่างอื่น เพื่อใช้สำหรับแต่ละพื้นที่ของสี่พื้นที่ของพื้นที่ที่อยู่ของกระบวนการ สำหรับอินเตอร์เฟสทั้งสองประเภท ควรระบุขนาดหน้าในหน่วยไบต์ ขนาดหน้า ที่ระบุอาจมีคำเติมท้ายเพื่อบ่งชี้หน่วย ของขนาด คำเติมท้ายที่ได้รับการสนับสนุนมีดังนี้:

- K (กิโลไบต์)
- M (เมกะไบต์)
- G (กิกะไบต์)

ข้อมูลเหล่านี้สามารถระบุเป็นตัวอักษรพิมพ์ใหญ่หรือพิมพ์เล็กก็ได้

เฉพาะขนาดหน้า 4 KB และ 64 KB เท่านั้นที่ ได้รับการสนับสนุนสำหรับพื้นที่หน่วยความจำทั้งสี่ส่วน ขนาดหน้า 16 MB ได้รับการสนับสนุน สำหรับข้อมูลกระบวนการ ข้อความกระบวนการ และพื้นที่หน่วยความจำแบบแบ่งใช้ ของกระบวนการเท่านั้น ขนาดหน้า 16 GB ได้รับการสนับสนุน สำหรับพื้นที่หน่วยความจำแบบแบ่งใช้ของกระบวนการเท่านั้น

โดยการเลือกขนาดหน้าที่ไม่ใช่ค่าดีฟอลต์ คุณต้องปิดใช้งานการใช้ขนาดหน้า ที่เล็กกว่าขนาดหน้าที่เลือกไว้ในเซกเมนต์เดียวกัน ด้วยตนเอง

ถ้าระบุขนาดหน้าที่ไม่ได้รับการสนับสนุน เคอร์เนลจะใช้ขนาดหน้าเล็กที่สุด ในลำดับถัดไปที่ได้รับการสนับสนุน ถ้าไม่มีขนาดหน้าที่เล็กกว่า ขนาดหน้าที่ระบุ เคอร์เนลจะใช้ขนาดหน้า 4 KB

การสนับสนุนการระบุขนาดเพจที่จะใช้สำหรับหน่วยความจำที่ แบ่งใช้ของกระบวนการที่มีตัวแปรสภาวะแวดล้อม SHMPSIZE พร้อมใช้งาน บน เวอร์ชันก่อนหน้านี้ของ AIX ตัวแปรสภาพแวดล้อม SHMPSIZE จะถูกเพิกเฉยไป ตัวแปรสภาพแวดล้อม SHMPSIZE ใช้เฉพาะกับพื้นที่หน่วยความจำแบบแบ่งใช้ของระบบ V ที่สร้างขึ้นโดย กระบวนการ เมื่อเรียกรูทีนย่อย **shmget,** รูทีนย่อย **ra_shmget,** และรูทีนย่อย **ra_shmgetv** เท่านั้น ตัวแปรสภาวะแวดล้อม SHMPSIZE ไม่ได้ใช้กับพื้นที่หน่วยความจำแบบแบ่งใช้ EXTSHM และพื้นที่หน่วยความจำแบบ แบ่งใช้เรียลไทม์ POSIX ตัวแปรสภาพแวดล้อม SHMPSIZE ของ กระบวนการไม่ได้ใช้กับพื้นที่หน่วยความจำแบบแบ่งใช้ เนื่องจาก กระบวนการใช้พื้นที่หน่วยความจำแบบแบ่งใช้ที่สร้างขึ้นโดยกระบวนการอื่น

การตั้งค่าขนาดหน้าที่ต้องการของแอ็พพลิเคชันด้วยคำสั่ง ldedit หรือ ld:

คุณสามารถตั้งค่าขนาดหน้าที่ต้องการของแอ็พพลิเคชันใน XCOFF/XCOFF64 ไบนารีด้วยคำสั่ง **ldedit** หรือ **ld**

คำสั่ง **ld** หรือ **cc** สามารถใช้ เพื่อตั้งค่าอ็อพชันขนาดหน้าเหล่านี้เมื่อคุณกำลังลิงก์ไฟล์ที่ดำเนินการได้:

```
ld -o mpsize.out -btextpsize:4K -bstackpsize:64K sub1.o sub2.o
cc -o mpsize.out -btextpsize:4K -bstackpsize:64K sub1.o sub2.o
```

คำสั่ง **ldedit** สามารถใช้เพื่อตั้งค่าอ็อพชันขนาดหน้าเหล่านี้ในไฟล์ที่ดำเนินการได้ที่มีอยู่:

```
ldedit -btextpsize=4K -bdatapsize=64K -bstackpsize=64K mpsize.out
```

หมายเหตุ: คำสั่ง **ldedit** ต้องการค่าสำหรับอ็อพชันขนาดหน้าที่จะระบุ พร้อมกับเครื่องหมายเท่ากับ (=) แต่คำสั่ง **ld** และ **cc** ต้องค่าสำหรับอ็อพชันขนาดหน้าที่จะระบุพร้อมกับเครื่องหมายโคลอน (:)

การตั้งค่าขนาดหน้าที่ต้องการของแอ็พพลิเคชันด้วย ตัวแปรสภาพแวดล้อม:

คุณสามารถตั้งค่าขนาดหน้าที่ต้องการของกระบวนการด้วย ตัวแปรสภาพแวดล้อม *LDR_CNTRL*

ดังตัวอย่าง คำสั่งต่อไปนี้จะทำให้ กระบวนการ `mpsize.out` ใช้หน้าขนาด 4 KB สำหรับข้อมูล หน้าขนาด 64 KB สำหรับข้อความ หน้าขนาด 64 KB สำหรับสแต็ค และหน้าขนาด 64 KB สำหรับหน่วยความจำแบบแบ่งใช้บนฮาร์ดแวร์ ที่สนับสนุน:

```
$ LDR_CNTRL=DATAPSIZE=4K@TEXTPSIZE=64K@SHMPSIZE=64K mpsize.out
```

ตัวแปร สภาพแวดล้อมขนาดหน้าแทนที่ค่าติดตั้งขนาดหน้าใดๆ ในส่วนหัว XCOFF ของไฟล์ที่ดำเนินการได้ นอกจากนี้ ตัวแปรสภาพแวดล้อม *DATAPSIZE* ยังแทนที่ค่าติดตั้งตัวแปรสภาพแวดล้อม *LARGE_PAGE_DATA* ใดๆ ด้วย

ข้อควรพิจารณาเกี่ยวกับการสนับสนุนแอ็พพลิเคชันหลายขนาดหน้า:

ปัญหาเกี่ยวกับกระบวนการ 32-บิต เธรดสแต็ค ไลบรารีแบบแบ่งใช้ หรือข้อมูลหน้าขนาดใหญ่สามารถส่งผลกระทบต่อความสามารถของ AIX ใน การสนับสนุนหลายขนาดหน้า

กระบวนการ 32-บิต

ด้วยรูปแบบพื้นที่ว่างที่อยู่ของกระบวนการ AIX 32-บิต ดีฟอลต์ เธรดสแต็คแรกเริ่มและข้อมูลของกระบวนการ อยู่ในเซกเมนต์ PowerPC® ขนาด 256 MB เดียวกัน เนื่องจากในปัจจุบัน สามารถใช้ขนาดหน้าได้เพียงหนึ่งขนาดเท่านั้นในเซกเมนต์หนึ่ง ดังนั้น ถ้ามีการระบุขนาดหน้าที่แตกต่างอื่นสำหรับสแต็คและข้อมูลของกระบวนการ 32-บิตมาตรฐาน ระบบจะใช้ขนาดหน้าที่เล็กลงสำหรับทั้งสแต็คและข้อมูล

อย่างไรก็ตาม กระบวนการ 32-บิตสามารถใช้ขนาดหน้าอื่นสำหรับเธรดสแต็คแรก เริ่มและข้อมูลได้โดยใช้รูปแบบพื้นที่ว่างที่อยู่อื่นอย่างใดอย่างหนึ่ง เพื่อสนับสนุนโปรแกรมขนาดใหญ่และ ขนาดใหญ่มาก ซึ่งระบุตำแหน่ง heap ข้อมูลของกระบวนการในเซกเมนต์ อื่นที่ไม่ใช่สแต็ค

เธรดสแต็ค

โดยค่าดีฟอลต์ เธรดสแต็คของกระบวนการแบบหลายเธรดมาจาก heap ข้อมูลของกระบวนการ ดังนั้น สำหรับกระบวนการแบบหลายเธรด ค่าติดตั้งขนาดหน้าสแต็ค จะใช้กับสแต็คสำหรับเธรดแรกเริ่มของกระบวนการเท่านั้น สแต็คสำหรับเธรดในลำดับต่อมาจะมีการจัดสรรจาก heap ข้อมูลของ กระบวนการ และสแต็คเหล่านี้จะใช้หน้าของขนาดที่ระบุโดยค่าติดตั้งขนาดหน้าข้อมูล

นอกจากนี้ การใช้หน้า 64 KB แทนหน้าขนาด 4 KB สำหรับข้อมูลของกระบวนการ แบบหลายเธรดยังลดจำนวน threads สูงสุดที่กระบวนการสามารถสร้างได้ด้วย เนื่อง จากข้อกำหนดการจัดวางสำหรับหน้าการป้องกันสแต็ค แอ็พพลิเคชันที่พบ ข้อจำกัดนี้สามารถปิดใช้งานหน้าการป้องกันสแต็ค เพื่อให้สามารถสร้างเธรดได้มาก ขึ้นโดยการตั้งค่าตัวแปรสภาพแวดล้อม *AIXTHREAD_GUARDPAGES* เป็น 0

ไลบรารีแบบแบ่งใช้

บนระบบที่สนับสนุนหน้า 64 KB AIX จะ ใช้หน้า 64 KB สำหรับพื้นที่ข้อความของไลบรารีแบบแบ่งใช้สากลเพื่อให้ประสิทธิภาพการทำงานดีขึ้น

ข้อมูลหน้าขนาดใหญ่

ตัวแปรสภาพแวดล้อม *DATAPSIZE* จะแทนที่ ตัวแปรสภาพแวดล้อม *LARGE_PAGE_DATA* นอกจากนี้ ค่าติดตั้ง *DATAPSIZE* ใน XCOFF ไบนารีของแอ็พพลิเคชัน จะแทนที่ค่าติดตั้ง lpdata ใดๆ ในไลบรารีเดียวกัน

การสนับสนุนเพจขนาดใหญ่ที่เปลี่ยนแปลงได้

ตัวประมวลผล POWER7® สนับสนุนการใช้ขนาดเพจ 4 KB, 64 KB และ 16 MB ผสมกันภายในเซกเมนต์ เดียว

ตัวประมวลผล POWER6 ให้แนวคิดการใช้ขนาดเพจผสมกันภายในเซกเมนต์เดียว แต่มีการสนับสนุนสำหรับการใช้ขนาดเพจ 4 KB และ 64 KB ผสมกัน ตัวประมวลผล POWER7 ขยาย แนวคิดนี้เพื่อสนับสนุนการใช้ขนาดเพจ 4 KB, 64 KB และ 16 MB ผสมกัน ภายในเซกเมนต์เดียว

ระบบปฏิบัติการ AIX สนับสนุน การใช้เพจ 16 MB เพื่อปรับปรุงสภาวะแวดล้อมที่มีประสิทธิภาพการทำงานสูง อย่างไรก็ตาม เพจของหน่วยความจำไม่ยืดหยุ่น หรือจัดการได้ไม่ง่าย เพจ 16 MB ไม่สามารถถูกดึงเพจออก และเพจ 16 MB ใหม่ไม่สามารถถูกสร้างโดยอัตโนมัติ

ตัวประมวลผล POWER7 สนับสนุน เพจผสม 16 MB ที่ช่วยให้ระบบปฏิบัติการมีความยืดหยุ่นในการ จัดการส่วนย่อย KB หรือ 64 KB ขณะที่ให้แอ็พพลิเคชันสามารถ ใช้ประโยชน์ของการเข้าถึงหน่วยความจำโดยใช้การแปลเพจฮาร์ดแวร์ 16 MB การใช้คุณลักษณะฮาร์ดแวร์นี้บนระบบปฏิบัติการ AIX เรียกว่า *Variable large page size support* (VLPSS)

VLPSS รวม พื้นที่ขนาด 16 MB และจัดแนวของหน่วยความจำผู้ใช้เป็นบล็อกของเพจ ขนาด 4 KB หรือ 64 KB ต่อกันทางฟิสิคัล เพจหน่วยความจำเหล่านี้เข้าถึงได้ ผ่านการแปล 16 MB เดียว เนื่องจากใช้การแปลเพจ 16 MB เดียว เพจ 4 KB และ 64 KB ที่ต้องการต้องมีแอ็ตทริบิวต์เพจ เดียวกัน และต้องอยู่ในหน่วยความจำ แอ็ตทริบิวต์เพจประกอบด้วย การป้องกันการอ่าน/เขียนเพจ การป้องกันคีย์หน่วยเก็บข้อมูล และการป้องกัน มิให้เรียกทำงาน

เพจ 16 MB VLPSS สามารถปรับลดจากการแปล 16 MB เป็นการแปลขนาดเพจ 4 KB หรือ 64 KB เริ่มต้น โดยระบบปฏิบัติการ เพจถูกปรับลดเมื่อระบบปฏิบัติการ ต้องดึงเพจออกในส่วนของหน่วยความจำไปยังอุปกรณ์การเพจ หรือ เมื่อแอ็พพลิเคชันเปลี่ยนแปลงแอ็ตทริบิวต์เพจสำหรับพื้นที่ 16 MB เพื่อที่ไม่ให้เป็นแบบเดียวกัน เพจ 16 MB จริงจะไม่มีความยืดหยุ่นเช่นนี้

แอ็พพลิเคชัน สามารถใช้ประโยชน์ของคุณลักษณะ VPLSS โดยใช้การเรียกใช้คำสั่งระบบ vm_pattr และโดยการระบุ คำสั่ง **VM_PA_SET_PSIZE_EXTENDED** ระบบปฏิบัติการ สามารถเลือกยอมรับคำแนะนำจากการเรียกใช้คำสั่งระบบ vm_pattr หรือปฏิเสธคำแนะนำนั้นในกรณีที่ระบบได้รับผลกระทบ

ขนาด เพจ 16 MB เป็นขนาดใหญ่ขึ้นของหน่วยความจำที่อยู่ต่อกันเมื่อเปรียบเทียบ กับขนาดเพจที่เปลี่ยนแปลงแบบไดนามิก 64 KB การรวบรวม และการปรับเพิ่ม หน่วยความจำเพื่อใช้ VLPSS เป็นการดำเนินการที่มีค่าใช้จ่ายสูงจึงอาจส่งผลลบ ต่อประสิทธิภาพการทำงานของทั้งระบบ ดังนั้น การปรับเพิ่มหน่วยความจำ เป็นขนาดเพจ 16 MB มีข้อจำกัดที่ การสนับสนุนขนาด เพจที่เปลี่ยนแปลงได้แบบไดนามิก ไม่มี

คุณลักษณะ VLPSS ถูกจำกัดต่อผู้ใช้ที่มีความสามารถ CAP_BYPASS_RAC_VMM และ CAP_PROPAGATE หรือผู้ที่มีสิทธิ root

พื้นที่หน่วยความจำ ผู้ใช้ 16 MB ต้องอยู่หน่วยความจำครบทั้งหมดเพื่อให้สามารถใช้ คุณลักษณะ VLPSS ได้ ระบบปฏิบัติการต้องการหน่วยความจำระบบที่มีขนาดใหญ่ เพื่อใช้คุณลักษณะ VLPSS ขนาดหน่วยความจำต่ำสุดที่ต้องการ สำหรับคุณลักษณะนี้คือ 16 GB

ขนาดหน้าและหน่วยความจำแบบแบ่งใช้

คุณสามารถเลือกขนาดหน้าที่จะใช้สำหรับหน่วยความจำแบบแบ่งใช้ระบบ V ได้โดยใช้คำสั่ง **SHM_PAGESIZE** ไปยังการเรียกระบบ shmctl()

โดยการเลือกขนาดหน้าที่ไม่ใช่ค่าดีฟอลต์ คุณต้องปิดใช้งานการใช้ขนาดหน้า ที่เล็กกว่าขนาดหน้าที่เลือกไว้ในเซกเมนต์เดียวกันด้วยตนเอง

โปรดดูหัวข้อ shmctl() ใน *Technical Reference: Base Operating System and Extensions, Volume 2* สำหรับ ข้อมูลเพิ่มเติมเกี่ยวกับการใช้ shmctl() เพื่อเลือกขนาดหน้าสำหรับพื้นที่ หน่วยความจำแบบแบ่งใช้

การกำหนดขนาดหน้าของกระบวนการโดยใช้คำสั่ง ps

คำสั่ง **ps** สามารถใช้เพื่อมอนิเตอร์ขนาดหน้าพื้นฐาน ที่จะใช้สำหรับข้อมูลของกระบวนการ สแต็ค และข้อความ

ตัวอย่างต่อไปนี้แสดงเอาต์พุตของคำสั่ง **ps -Z** คอลัมน์ *DPGSZ* แสดงขนาดหน้าข้อมูลของกระบวนการ คอลัมน์ *SPGSZ* แสดงขนาดหน้าสแต็คของกระบวนการ และคอลัมน์ *TPGSZ* แสดง ขนาดหน้าข้อความของกระบวนการ

```
# ps -Z
    PID    TTY  TIME DPGSZ SPGSZ TPGSZ CMD
 311342  pts/4  0:00    4K    4K    4K ksh
 397526  pts/4  0:00    4K    4K    4K ps
 487558  pts/4  0:00   64K   64K    4K sleep
```

การมอนิเตอร์ขนาดหน้าโดยใช้คำสั่ง vmstat

คำสั่ง **vmstat** มีอยู่สองอ็อพชัน สำหรับการแสดงสถิติหน่วยความจำของขนาดหน้าเฉพาะ

vmstat -p
: แสดงข้อมูล vmstat สากลและรายละเอียดของสถิติสำหรับแต่ละ ขนาดหน้า

vmstat -P
: แสดงสถิติสำหรับแต่ละขนาดหน้า

อ็อพชันทั้งสองใช้รายการที่คั่นด้วยเครื่องหมายจุลภาคของขนาดหน้าเฉพาะ หรือคีย์เวิร์ด **all** เพื่อบ่งชี้ข้อมูลที่ควรจะแสดงขึ้นสำหรับขนาดหน้าที่สนับสนุนทั้งหมดซึ่งมีเฟรมหน้าตั้งแต่หนึ่งเฟรม ขึ้นไป ตัวอย่างต่อไปนี้แสดงข้อมูลสำหรับแต่ละขนาดหน้าของขนาดหน้าทั้งหมด ที่มีเฟรมหน้าบนระบบ:

```
# vmstat -P all
System configuration: mem=4096MB
pgsz            memory                           page
----- ------------------------- ------------------------------------
           siz      avm      fre    re    pi    po    fr     sr    cy
   4K   542846   202832   329649     0     0     0     0      0     0
  64K    31379      961    30484     0     0     0     0      0     0
```

การมอนิเตอร์ขนาดหน้าทั่วทั้งระบบโดยใช้คำสั่ง svmon

คำสั่ง **svmon** สามารถใช้เพื่อแสดงการใช้ขนาดหน้า บนระบบ

คำสั่ง **svmon** มีการพัฒนาขึ้นเพื่อนำเสนอสถิติสำหรับแต่ละ ขนาดหน้า ตัวอย่างเช่น เมื่อต้องการแสดงสถิติสากลเกี่ยวกับแต่ละ ขนาดหน้า สามารถใช้อ็อพชัน **-G** พร้อมกับคำสั่ง **svmon**:

```
# svmon -G
               size      inuse       free        pin    virtual
memory      8208384    5714226    2494158     453170    5674818
pg space     262144      20653

               work       pers       clnt
```

```
pin           453170          0          0
in use       5674818        110      39298

PageSize   PoolSize      inuse       pgsp        pin    virtual
s    4 KB          -    5379122      20653     380338    5339714
m   64 KB          -      20944          0       4552      20944
```

หากต้องการข้อมูลเพิ่มเติม ให้ดูคำสั่ง **svmon** ใน *Commands Reference, Volume 5*

ข้อควรพิจารณาเกี่ยวกับการใช้หน่วยความจำสำหรับเพจที่มีขนาดใหญ่

เมื่อคุณกำลังประเมินผลการทำงานที่อาจกระทบกับการใช้ขนาดของเพจที่มีขนาดใหญ่ สำหรับแอ็พพลิเคชัน การใช้หน่วยความจำของเวิร์กโหลด ต้องถูกนำมาพิจารณา

การใช้ขนาดของเพจที่มีขนาดใหญ่กว่าอาจส่งผลทำให้การติดตามหน่วยความจำที่เพิ่มขึ้นของเวิร์กโหลด เนื่องจากการแตกแฟรกเมนต์หน่วยความจำ คำสั่ง **svmon** และ **vmstat** สามารถนำมาใช้เพื่อมอนิเตอร์การใช้หน่วยความจำของเวิร์กโหลด เพื่อพิจารณาว่า การติดตามหน่วยความจำของเน็ตเวิร์กจะเพิ่มขึ้นขณะที่ใช้ขนาดของเพจที่มีขนาดใหญ่

ขณะที่พิจารณาขนาดของเพจ 64 KB ให้ใช้ขนาดของเพจดีฟอลต์แทน และอนุญาตให้ระบบปฏิบัติการตัดสินใจที่จะใช้ขนาดของเพจ ถ้าชุดการทำงานของแอ็พพลิเคชันเต็มไปด้วยข้อมูล เช่น ส่วนที่มีขนาดเต็ม 64 จำเป็นต้องมี การเลือกเพจขนาด 64 KB จะมีความเหมาะสม เนื่องจากจะมีหน่วยความจำขนาดเล็ก ที่ไม่ได้ใช้งาน (เช่น จัดสรรแล้ว แต่ไม่เคยใช้งาน)

การปรับหน่วยความจำอย่างต่อเนื่อง

โปรแกรมการปรับหน่วยความจำแบบต่อเนื่องจะรันการรวมเพจและการเลื่อนขั้นแบบไดนามิก

การสนับสนุนขนาดของเพจที่เปลี่ยนแปลงแบบไดนามิก

ความสามารถของ AIX Version 6.1 ในการรวมการใช้หน่วยความจำที่มีขนาดเพจ 4 KB ที่สงวนไว้ในพื้นที่หน่วยความจำที่อ้างอิงกับผลการทำงานที่ได้ประโยชน์จากขนาดของเพจ 64 KB ในส่วนของหน่วยความจำที่อ้างอิง และใช้โดยอัตโนมัติโดยไม่แทรกแซงผู้ใช้ เรียกว่า dynamic variable page size support (DVPSS)

DVPSS ขึ้นอยู่กับความสามารถของ POWER6 เพื่อสนับสนุนขนาดของเพจแบบผสมภายในเซกเมนต์เดียว สถาปัตยกรรมสนับสนุนการเปลี่ยนลำดับของขนาดเพจต่างๆ อย่างไรก็ตาม POWER6 สนับสนุนการรวมเพจขนาด 4 KB และ 64 KB

ค่าดีฟอลต์สำหรับเซกเมนต์ขนาดของเพจที่เปลี่ยนแปลง คือ เพจขนาด 4 KB และการแปลขนาด 4 KB ในขนาด 64 KB และส่วนการจัดเรียงจนกว่าเพจขนาด 16 4 KB ทั้งหมดจะถูกอ้างถึง เมื่อเพจ 16 เพจทั้งหมดถูกอ้างถึง DVPSS จะตรวจสอบสถานะเพื่อให้มีสถานะเดียวกัน (เช่น การป้องกับการอ่านเพจ หรือ เขียนเพจ ไม่มีการป้องกันการดำเนินการ การป้องกันคีย์ที่เก็บข้อมล และไม่อยู่ในสถานะ I/O เหมือนกัน) ถ้าเพจเหล่านั้นทำสิ่งนี้ การแปลขนาด 4 KB จะถูกลบออก และแทนที่ด้วยการแปลขนาด 64 KB

Continuous Program Optimization Agent (CPOagent)

ข้อจำดในการสนับสนุนเมธอด DVPPS โดยระบบปฏิบัติการ คือ เพจขนาด 4 KB ทั้ง 16 เพจต้องถูกอ้างถึงก่อนที่จะปรับขนาดของเพจเป็น 64 KB CPOagent จะช่วยให้ผ่านข้อจำกัดนี้โดยการใช้การปรับหน่วยความจำอย่างต่อเนื่องที่ทำการรวมเพจและการปรับแบบไดนามิก คุณลักษณะนี้ใช้ได้กับ AIX 6.1 เทคโนโลยีระดับ 6 และสูงกว่า

คุณสามารถเข้าถึง CPOagent ได้ที่:

usr/lib/perf/CPOagent

ไวยากรณ์

CPOagent [−f *configuration file*]

แฟล็ก

ไอเท็ม	คำอธิบาย
−f	เปลี่ยนชื่อไฟล์คอนฟิกูเรชันดีฟอลต์ ถ้าไม่ได้ระบุอ็อพชันนี้ จะสันนิษฐานว่าชื่อไฟล์มีอยู่ที่ตำแหน่งของไฟล์ /usr/lib/perf/CPOagent.cf

CPOagent จะไม่รันโดยดีฟอลต์ ผู้ใช้รูทสามารถสตาร์ท CPOagent ได้อย่างชัดเจน เมื่อถูกสตาร์ท CPOagent จะรันอยู่เบื้องหลังเพื่อระบุกระบวนการที่สามารถได้รับประโยชน์จากขนาดของเพจที่ใหญ่ขึ้น กระบวนการจะถูกระบุตามการใช้หน่วยความจำและตัวประมวลผลที่เกินค่าขีดจำกัดที่ระบุ

หมายเหตุ: ปัจจุบัน CPOagent สามารถปรับขนาดของเพจเป็น 64 KB ได้

ไฟล์คอนฟิกูเรชันของ CPOagent

เมื่อ CPOagent สตาร์ท ข้อมูลในไฟล์คอนฟิกูเรชันจะถูกอ่านและวิเคราะห์ ไฟล์คอนฟิกูเรชันเป็นไฟล์ stanza ที่มีฟิลด์ที่มี:

```
TCPU=<n1>
TMEM=<n2>
PATI=<n3>
PATM=<n4>
PPTS=<n5>
TOPM=<n6>
PFLR=<c>
```

คำอธิบายของฟิลด์ในไฟล์คอนฟิกูเรชันมีดังนี้:

ฟิลด์	คำอธิบาย
TCPU	ระบุขีดจำกัดการใช้งาน CPU ต่อกระบวนการ หน่วยเป็นเปอร์เซ็นต์ ดีฟอลต์: 25 ต่ำสุด: 10 สูงสุด: 100
TMEM	ระบุขีดจำกัดการใช้หน่วยความจำต่อกระบวนการ หน่วยเป็น MB ดีฟอลต์: 1 ต่ำสุด: 1
PATI	ระบุช่วงเวลาการวิเคราะห์เพจ (PATI) หน่วยเป็นนาที ซึ่งจะระบุช่วงเวลาที่กระบวนการถูกวิเคราะห์เพื่อระบุเพจที่สามารถถูกรวมและปรับเป็นขนาดที่สูงขึ้น ดีฟอลต์: 15 ต่ำสุด: 5 สูงสุด: 60

ฟิลด์	คำอธิบาย
PATM	ระบุการมอนิเตอร์เวลาการวิเคราะห์เพจ (PATM) หน่วยเป็นนาที ซึ่งจะระบุจำนวนเวลาของสถิติการใช้เพจที่จะถูกรวบรวมสำหรับการระบุเพจสำหรับการรวมและการปรับเพจ ดีฟอลต์: 30 ต่ำสุด: 5 สูงสุด: 180
PPTS	ระบุ page promotion trigger samples (PPTS) ซึ่งจะระบุจำนวนของตัวอย่างที่จะรวบรวมก่อนการทริกเกอร์การปรับเพจ ดีฟอลต์: 4 ต่ำสุด: 4
TOPM	ระบุจำนวนของกระบวนการที่ใช้ CPU สูงสุดต่อ CPU ที่ต้องถูกพิจารณาสำหรับการรวมและการปรับเพจ ดีฟอลต์: 2 ต่ำสุด: 1
PFLR	ระบุไวด์การ์ดและกระบวนการที่ตรงกับไวด์การ์ดที่จะถูกพิจารณาโดย CPOagent สำหรับการรวมและการปรับเพจ ซึ่งจะถูกอ้างถึงเป็น process filter wildcard (PFLR)

ข้อได้เปรียบของการใช้ CPOagent

ข้อได้เปรียบของการใช้กลไก CPOagent รวมถึง:

- แอ็พพลิเคชันจะไม่ทราบถึงการเปลี่ยนแปลง ดังนั้น จึงไม่มีความจำเป็นที่จะต้องทำการเปลี่ยนแปลงกับแอ็พพลิเคชัน
- การปรับเพจจะรันตามนโยบายที่ตั้งโดยผู้ดูแลระบบในไฟล์ CPOagent.cf ซึ่งขึ้นอยู่กับความต้องการเวิร์กโหลดบน CPU และหน่วยความจำ ซึ่งจะช่วยการปรับเพจอย่างมีประสิทธิภาพ การบวนการนี้มีการควบคุมเป็นส่วนย่อยๆ สำหรับการสนับสนุนขนาดของเพจที่เปลี่ยนแปลงแบบไดนามิกที่ให้แน่ใจว่าการปรับเพจจะรันสำหรับแอ็พพลิเคชันที่ต้องการ

สถานการณ์จำลองตัวอย่าง

พิจารณาแอ็พพลิเคชัน StressEngine ที่รันบนระบบ แอ็พพลิเคชันมีการใช้ CPU และหน่วยความจำสูง ถ้าไม่ได้ใช้ CPOagent แอ็พพลิเคชัน StressEngine จะไม่สามารถสนับสนนขนาดของเพจที่เปลี่ยนแปลงแบบไดนามิกได้ จนกว่าเพจ 16 เพจของเซกเมนต์ที่ระบุจะถูกอ้างถึงและเพจอยู่ในสถานะที่เหมือนกัน ขนาดของเพจสามารถตรวจสอบได้กับรายงานกระบวนการที่สร้างขึ้นด้วยคำสั่ง **svmon**

```
# svmon -P 8454254 -O pgsz=on,unit=MB,segment=on
Unit: MB
------------------------------------------------------------------------------
     Pid Command          Inuse      Pin     Pgsp  Virtual
 8454254 StressEngine   157.87     42.3        0   157.84

     PageSize      Inuse        Pin       Pgsp    Virtual
     s    4 KB               64.2       0.02          0       64.2
     m   64 KB               93.7       42.3          0       93.7

    Vsid      Esid Type Description              PSize  Inuse   Pin Pgsp Virtual
  86f49b         2 work process private             s   64.1  0.02    0    64.1
    9000         d work shared library text         m   47.9     0    0    47.9
    8002         0 work fork tree                   m   45.8  42.3    0    45.8
```

```
                  children=939b1c, 0
  80fdc3         f work shared library data          s   0.09     0    0    0.09
  85fd37         1 clnt code,/dev/hd1:10             s   0.02     0    -       -
```

ถ้า CPOagent ถูกสตาร์ท และมีไฟล์ CPOagent.cf ตัวอย่างต่อไปนี้

```
-----------------------------------------
TCPU=25
TMEM=50
PATI=15
PATM=30
PPTS=4
TOPM=2
PFLR=Stress*
-----------------------------------------
```

ขึ้นอยู่กับไฟล์คอนฟิกูเรชัน รอบของ CPOagent จะเป็นเวลา 15 นาที(PATI =15) สำหรับช่วงเวลา 15 นาทีนั้น จะมอนิเตอร์การใช้ CPU และหน่วยความจำของกระบวนการที่กำลังรัน กระบวนการ 2 กระบวนการสูงสุด (TOPM =2) ที่ชื่อของกระบวนการมี Stress (PFLR = Stress*), การใช้ CPU เกิน 25% (TCPU = 25) และการใช้หน่วยความจำเกิน 50 MB (TMEM = 50) เป็นเพจสำหรับการรวมและการปรับเพจ กระบวนการนี้จะถูกตรวจสอบโดยการรวบรวมตัวอย่าง (PPTS = 4) 4 ตัวอย่างก่อนการทริกเกอร์อัลกอริทึมสำหรับการรวมและการปรับเพจ นอกจากนี้ สถิติการใช้เพจจะถูกรวมรวม 30 วินาที (PATM =30) เพื่อระบุเพจสำหรับการรวมและการปรับเพจ ตอนนี้โดยการใช้ CPOagent จึงไม่ต้องรอให้เพจทั้ง 16 ของเซกเมนต์ที่ระบุถูกอ้างถึง CPOagent จะประเมินถ้าแอ็พพลิเคชันต้องการการรวมและการปรับเพจโดยการอ้างถึงไฟล์คอนฟิกูเรชัน CPOagent.cf และความต้องการของแอ็พพลิเคชันเกี่ยวกับการใช้ CPU และหน่วยความจำ เพจที่ถูกปรับเป็นหลักฐานจากรายงานกระบวนการที่สร้างขึ้นด้วยคำสั่ง **svmon**

```
# svmon -P 8454254 -O pgsz=on,unit=MB,segment=on
Unit: MB
--------------------------------------------------------------------------------
     Pid Command          Inuse      Pin     Pgsp  Virtual
 8454254 StressEngine    157.87     42.3        0   157.84

     PageSize      Inuse        Pin       Pgsp    Virtual
     s    4 KB               64.2       0.02          0       64.2
     m   64 KB               93.7       42.3          0       93.7

    Vsid      Esid Type Description              PSize  Inuse   Pin Pgsp Virtual
  86f49b         2 work process private            sm   64.1   0.02    0    64.1
    9000         d work shared library text         m   47.9      0    0    47.9
    8002         0 work fork tree                   m   45.8   42.3    0    45.8
                   children=939b1c, 0
  80fdc3         f work shared library data         sm   0.09      0    0    0.09
  85fd37         1 clnt code,/dev/hd1:10            s   0.02      0    -       -
```

VMM thread interrupt offload

โครงสร้างพื้นฐาน VMM thread interrupt offload (VTIOL) อนุญาตให้ VMM ลดภาระ การประมวลผลของเซอร์วิส iodone() สำหรับเคอร์เนลเธรด

ฟังก์ชัน VTIOL ใช้เพื่อลดความเป็นไปได้ที่กระบวนการ iodone() จะมีผลกระทบกับประสิทธิภาพของเธรดที่มีลำดับความสำคัญสูง ฟังก์ชัน VTIOL จะจัดการกับเซอร์วิส iodone() โดยใช้เธรดที่ทำงานอยู่เบื้องหลังแทนที่จะอินเตอร์รัปต์กระบวนการ

ที่มีลำดับความสำคัญสูง VMM ใช้การวิเคราะห์ต่างๆ สำหรับการพิจารณาว่าจะลดภาระการประมวลผล เซอร์วิส iodone() หรือไม่ เช่น การดำเนินการ I/O บางอย่างที่ไม่มีเธรด waiter ที่ชัดเจน เช่น การดำเนินการเขียนที่ทำงานอยู่เบื้องหลัง และการดำเนินการอ่านล่วงหน้า สามารถลดภาระลงได้ การดำเนินการ I/O ที่มีเธรด waiter ที่ชัดเจน อาจระบุว่าการดำเนินการ I/O ต้องเสร็จสมบูรณ์โดยมีลำดับความสำคัญสูง ในกรณีเหล่านี้ การดำเนินการ I/O จะไม่ถูกลดภาระ และถูกประมวลผลมที่ระดับอินเตอร์รัปต์

VMM thread interrupt offload reprioritization

โครงสร้างพื้นฐาน VMM thread interrupt offload reprioritization (VTIOLR) มีการปรับปรุง สำหรับฟังก์ชัน VMM thread interrupt offload (VTIOL)

โครงสร้างพื้นฐาน VTIOLR อนุญาตให้ระบบปฏิบัติการ AIX กำหนดลำดับความสำคัญลำดับที่การดำเนินการ VMM I/O ที่ถูกลดภาระถูกประมวลผลตาม เกณฑ์ที่ระบุ เช่น หากการดำเนินการ read() ขนาดเล็กถูกเข้าคิวหลังจากการดำเนินการ I/O ขนาดใหญ่ให้เสร็จสมบูรณ์จากกระบวนการ sync() ของระบบไฟล์ การดำเนินการ read() อาจถูกพิจารณาเป็นงานที่ทำงานอยู่เบื้องหน้าขณะที่กระบวนการ sync() สามารถพิจารณา เป็นงานที่ทำงานอยู่เบื้องหลัง ในกรณีนี้ คุณลักษณะ VTIOLR สามารถจัดลำดับความสำคัญการดำเนินการ read() ใหม่ และดำเนินการ I/O ให้เสร็จสมบูรณ์ก่อนที่จะประมวลผลบัฟเฟอร์ iodone() อื่น บัฟเฟอร์ iodone() ที่จัดลำดับความสำคัญใหม่ จะถูกประมวลผลโดยการออฟโหลดเป็นชุดของเธรดที่อยู่เบื้องหลัง ดังนั้น การดำเนินการ I/O จะเสร็จสมบูรณ์โดย เธรดที่อยู่เบื้องหลัง และไม่ใช่ที่ระดับอินเตอร์รัปต์

ผลการทำงานของโลจิคัลวอลุ่มและดิสก์ I/O

หัวข้อนี้เน้นถึงผลการทำงานของโลจิคัลวอลุ่มและดิสก์ไดร์ฟที่พ่วงต่อ แบบโลคัล

ถ้าคุณไม่คุ้นเคยกับหลักการเกี่ยวกับระบบปฏิบัติของกลุ่มวอลุ่ม โลจิคัลและฟิสิคัลวอลุ่ม หรือโลจิคัลและฟิสิคัลพาร์ติชัน โปรดอ่าน ภาพรวมของผลการทำงานของการจัดการหน่วยเก็บฮาร์ดดิสก์

การตัดสินใจเกี่ยวกับจำนวนและชนิดของฮาร์ดดิสก์ และขนาดและการวางตำแหน่งพื้นที่การเพจ และโลจิคัลวอลุ่มบนฮาร์ดดิสก์เหล่านั้นคือกระบวนการติดตั้งล่วงหน้าที่สำคัญ เนื่องจากเกี่ยวข้องกับผลการทำงาน สำหรับการกล่าวถึงข้อควรพิจารณาที่ครอบคลุมถึงเรื่องของ การวางแผนคอนฟิกูเรชันการติดตั้งดิสก์ล่วงหน้า โปรดดู แนวทางในการติดตั้งดิสก์ล่วงหน้า

หลักการที่เกี่ยวข้อง:

"คำแนะนำในการติดตั้งดิสก์ล่วงหน้า" ในหน้า 105

กลไกสำหรับการนิยามและการขยายโลจิคัลวอลุ่ม มีความพยายามในการทำตัวเลือกดีฟอลต์ที่เป็นไปได้มากที่สุด อย่างไรก็ตาม ผลการทำงานของดิสก์ I/O ที่น่าพึงพอใจจะมีเพิ่มมากขึ้น หากโปรแกรมติดตั้งของระบบ ปรับให้เหมาะสมกับขนาดและตำแหน่งของโลจิคัลวอลุ่มในหน่วยเก็บข้อมูล และข้อกำหนดเกี่ยวกับเวิร์กโหลด

การมอนิเตอร์ดิสก์ I/O

มีหลายปัญหาที่คุณควรพิจารณา เพื่อกำหนดวิธีการในการดำเนินการของคุณ เมื่อคุณกำลังมอนิเตอร์ดิสก์ I/O

- ค้นหาแอ็คทีฟไฟล์ส่วนใหญ่ ระบบไฟล์ และโลจิคัลวอลุ่ม:
  - สามารถทำให้ระบบไฟล์ "ที่ได้รับความนิยม" อยู่ในตำแหน่งที่ดีกว่าบนฟิสิคัลไดร์ฟ หรือกระจายระหว่างฟิสิคัลไดร์ฟจำนวนมากหรือไม่? (**lslv, iostat, filemon**)
  - ไฟล์ "ที่ได้รับความนิยม" อยู่บนโลคัลหรือรีโมต? (**filemon**)
  - พื้นที่การเพจครอบคลุมการใช้ประโยชน์จากดิสก์หรือไม่? (**vmstat, filemon**)

  - มีหน่วยความจำที่เพียงพอเพื่อแคชไฟล์เพจที่ใช้โดยการประมวลผลที่รันอยู่หรือไม่? (**vmstat, svmon**)
  - แอ็พพลิเคชันดำเนินการกับไฟล์ I/O แบบซิงโครนัส (ไม่ได้แคชไว้) เป็นจำนวนมากหรือไม่?
- ให้กำหนดการแตกแฟรกเมนต์ของไฟล์:
  - ไฟล์ "ที่ได้รับความนิยม" อย่างสูงถูกแฟรกเมนต์หรือไม่? (**fileplace**)
- ค้นหาฟิสิคัลวอลุ่มด้วยการใช้ประโยชน์สูงสุด:
  - ชนิดของไดร์ฟหรืออะแด็ปเตอร์ I/O เป็นสาเหตุทำให้เกิดคอขวดหรือไม่? (**iostat, filemon**)

โครงสร้างของเส้นบรรทัดการปรับล่วงหน้า

ก่อนที่คุณจะทำการเปลี่ยนแปลงที่สำคัญ ในคอนฟิกูเรชันดิสก์ของคุณ หรือปรับพารามิเตอร์ จึงเป็นความคิดที่ดีในการสร้างเส้นบรรทัดของการวัดที่บันทึกคอนฟิกูเรชัน และผลการทำงานในปัจจุบัน

การรายงานเวลารอ I/O

AIX Version 6.1 และใหม่กว่า มีการพัฒนาขึ้นในวิธีการที่ใช้คำนวณเปอร์เซ็นต์ของเวลาตัวประมวลผล ที่ใช้รอดิสก์ I/O (เวลา wio)

CPU ที่ว่างมีการทำเครื่องหมายเป็น wio ถ้า I/O ที่ค้างอยู่เริ่มต้น บน CPU นั้น

นอกจากนี้ การรอ I/O บนระบบไฟล์ที่ติดตั้ง NFS มีการรายงานเป็น เวลารอ I/O ด้วย

การประเมินค่าผลการทำงานดิสก์ด้วยคำสั่ง iostat

เริ่มต้นการประเมินผลโดยรันคำสั่ง **iostat** ด้วยพารามิเตอร์ช่วงเวลาในระหว่างที่มีเวิร์กโหลดสูงสุดของระบบ หรือขณะที่รันแอ็พพลิเคชันที่สำคัญซึ่งคุณต้องการลดเวลาหน่วงของ I/O

สคริปต์ shell ต่อไปนี้จะรันคำสั่ง **iostat** ในส่วนหลังขณะที่สำเนาของไฟล์ขนาดใหญ่จะรันอยู่ในพื้นหน้า ดังนั้น จึงมี I/O บางตัวที่ใช้วัด:

```
# iostat 5 3 >io.out &
# cp big1 /dev/null
```

ระบบปฏิบัติการ AIX เก็บรักษา ประวัติของกิจกรรมดิสก์ ถ้าประวัติดิสก์ I/O ปิดใช้งาน ข้อความต่อไปนี้จะแสดงขึ้นเมื่อคุณรันคำสั่ง **iostat**:

```
Disk history since boot not available.

The interval disk I/O statistics are unaffected by this.
```

เมื่อต้องการ เปิดใช้งานประวัติดิสก์ I/O จากบรรทัดรับคำสั่ง ให้ป้อน `smit chgsys` จากนั้นเลือก **true** จากฟิลด์ **เก็บรักษาประวัติดิสก์ I/O อย่างต่อเนื่อง**

ตัวอย่าง ต่อไปนี้จัดเก็บรายงานสามฉบับไว้ในไฟล์ `io.out`:

```
# iostat 5 3 >io.out &
# cp big /dev/null

System configuration: lcpu=4 drives=1 ent=0.50 paths=1 vdisks=1

tty:      tin         tout    avg-cpu: % user % sys % idle % iowait physc % entc
```

```
          0.0          0.0      0.1      0.6    99.2   0.1    0.0     1.2

Disks:        % tm_act     Kbps      tps    Kb_read   Kb_wrtn
hdisk0           0.4       5.6       0.6         28         0

tty:      tin         tout    avg-cpu: % user % sys % idle % iowait physc % entc
          0.0          0.0      0.1      1.4    98.4   0.2     0.0    2.4

Disks:        % tm_act     Kbps      tps    Kb_read   Kb_wrtn
hdisk0           6.0     123.2      10.6        4       612

tty:      tin         tout    avg-cpu: % user % sys % idle % iowait physc % entc
          2.6          2.6     0.1        0.5   99.4    0.0    0.0     1.1

Disks:        % tm_act     Kbps      tps    Kb_read   Kb_wrtn
hdisk0           0.0       0.0       0.0          0         0
```

รายงานแรกคือ รายงานสรุปตั้งแต่รีบูตครั้งล่าสุด และแสดงสมดุลโดยรวม (หรือ ในกรณีนี้ อาจไม่เกิดการสมดุล) ใน I/O กับฮาร์ดดิสก์แต่ละตัว hdisk1 จะไม่ได้ทำงานเป็นส่วนใหญ่ และ hdisk2 จะได้รับ 63 เปอร์เซ็นต์ของ I/O ทั้งหมด (จาก Kb_read และ Kb_wrtn)

รายงานฉบับที่สอง จะแสดงช่วงเวลา 5 วินาทีในระหว่างที่รัน **cp** เวลาที่ใช้ไปสำหรับ **cp** นี่คือ 2.6 วินาที ดังนั้น 2.5 วินาทีของการพึ่งพา I/O สูงสุดจะเป็นการเฉลี่ยด้วยเวลา 2.5 วินาทีของเวลาสูญเปล่ากับอัตราผลประโยชน์ 39.5 เปอร์เซ็นต์ตามที่รายงาน % iowait ช่วงเวลาที่สั้นที่สุดจะถูกกำหนดคุณสมบัติโดยละเอียดเพิ่มเติม ของตัวคำสั่งเอง แต่ในตัวอย่างนี้ยังคงสาธิตสิ่งที่คุณต้องพิจารณา เมื่อคุณกำลังมองหารายงานที่แสดงกิจกรรมโดยเฉลี่ย ระหว่างช่วงเวลา

หลักการที่เกี่ยวข้อง:

"คำสั่ง iostat" ในหน้า 115

คำสั่ง **iostat** จะเป็นวิธีที่เร็วที่สุด ในการขอรับความพึงพอใจในครั้งแรก ไม่ว่าระบบจะมีปัญหาเกี่ยวกับผลการทำงานในขอบเขตของดิสก์ I/O หรือไม่ก็ตาม

สิ่งอ้างอิงที่เกี่ยวข้อง:

"การมอนิเตอร์ผลการทำงานอย่างต่อเนื่องด้วยคำสั่ง iostat" ในหน้า 16

คำสั่ง **iostat** มีประโยชน์สำหรับการพิจารณาดิสก์ และการใช้ CPU

รายงาน TTY:

สองคอลัมน์ของข้อมูล TTY (*tin* และ *tout*) ในเอาต์พุต **iostat** แสดงจำนวนของอักขระที่อ่าน และที่บันทึกโดยอุปกรณ์ TTY ทั้งหมด

รวมทั้งอุปกรณ์ TTY จริงและ pseudo อุปกรณ์ TTY จริงคืออุปกรณ์ที่เชื่อมต่อกับอะซิงโครนัสพอร์ต อุปกรณ์ TTY แบบ pseudo คือ shells, **telnet** sessions, และหน้าต่าง **aixterm**

เนื่องจากการประมวลผลของอักขระอินพุตและเอาต์พุตใช้รีซอร์ส CPU ให้ดูความสัมพันธ์ระหว่างกิจกรรม TTY ที่เพิ่มขึ้นและการใช้ประโยชน์ CPU ถ้ามีความสัมพันธ์นั้นอยู่ให้ประเมินวิธีปรับปรุงประสิทธิภาพของระบบย่อย TTY ขั้นตอนที่สามารถใช้ได้รวมถึงการเปลี่ยนแอ็พพลิเคชันโปรแกรม การแก้ไข TTY พอร์ตพารามิเตอร์ในระหว่างการโอนย้ายไฟล์ หรือบางที การอัพเกรดเป็น อะแด็ปเตอร์การสื่อสารแบบอะซิงโครนัสที่เร็วกว่าหรือมีประสิทธิภาพมากกว่า

รายงานไมโครโพรเซสเซอร์:

คอลัมน์ข้อมูลสถิติไมโครโพรเซสเซอร์ (`% user`, `% sys`, `% idle` และ `% iowait`) จัดเตรียมการแตกของการใช้ไมโครโพรเซสเซอร์

ข้อมูลนี้ยังรายงานอยู่ในเอาต์พุตคำสั่ง **vmstat** ในคอลัมน์ที่มีป้าย `us`, `sy`, `id` และ `wa` สำหรับคำอธิบายโดยละเอียดสำหรับค่า โปรดดู "คำสั่ง vmstat" ในหน้า 112 และให้จดบันทึกการเปลี่ยนแปลงที่ทำไว้ `% iowait` ซึ่งกล่าวไว้ใน "การรายงานเวลารอ I/O" ในหน้า 196

บนระบบที่รันหนึ่งแอ็พพลิเคชัน เปอร์เซ็นต์รอ I/O ที่มีค่าสูงอาจเกี่ยวข้องกับ เวิร์กโหลด บนระบบที่มีการประมวลผลจำนวนมาก การประมวลผลบางตัวจะรันขณะที่การประมวลผลตัวอื่นจะรอ I/O ในกรณีนี้ `% iowait` สามารถมีขนาดเล็กหรือมีค่าศูนย์ เนื่องจากการประมวลผลที่รันจะ "ซ่อน" ช่วงเวลารอ แม้ว่า `% iowait` จะมีขนาดต่ำ ปัญหาเรื่องคอขวดยังสามารถจำกัดผลการทำงานของแอ็พพลิเคชันได้

ถ้าคำสั่ง **iostat** บ่งชี้ว่าสถานการณ์ที่โยงกับไมโครโพรเซสเซอร์ไม่มีอยู่ และเวลา `% iowait` มีค่ามากกว่า 20 เปอร์เซ็นต์ คุณอาจต้องมี I/O หรือสถานการณ์ที่โยงกับดิสก์ สถานการณ์นี้อาจเป็นสาเหตุของการเพจที่เกินปกติ เนื่องจากขาดแคลนหน่วยความจำที่ใช้จริง ซึ่งยังอาจมีสาเหตุมาจากโหลดดิสก์ไม่เกิดความสมดุล ข้อมูลการแตกแฟรกเมนต์ หรือรูปแบบการใช้ สำหรับโหลดดิสก์ที่ไม่เกิดความสมดุล รายงาน **iostat** บางตัวจัดเตรียมข้อมูล ที่จำเป็น แต่สำหรับข้อมูลที่เกี่ยวกับระบบไฟล์หรือโลจิคัลวอลุ่ม ซึ่งเป็นโลจิคัลรีซอร์ส คุณต้องใช้เครื่องมือ เช่น คำสั่ง **filemon** หรือ **fileplace**

รายงานไดร์ฟ:

รายงานไดร์ฟมีข้อมูลที่เกี่ยวข้องกับผลการทำงานสำหรับฟิสิคัลไดร์ฟ

เมื่อคุณสงสัยปัญหาเกี่ยวกับผลการทำงานดิสก์ I/O ให้ใช้คำสั่ง **iostat** หากต้องการหลีกเลี่ยงข้อมูลเกี่ยวกับ TTY และข้อมูลสถิติ CPU ให้ใช้อ็อพชัน **-d** นอกจากนี้ ข้อมูลสถิติดิสก์สามารถจำกัดความสำคัญของดิสก์ตามการระบุชื่อดิสก์

โปรดจำไว้ว่า ชุดแรกของข้อมูลแสดงกิจกรรมทั้งหมดตั้งแต่เริ่มต้นระบบ

ดิสก์: แสดงชื่อของฟิสิคัลวอลุ่ม ดิสก์คือ `hdisk` หรือ `cd` อย่างใดอย่างหนึ่ง แล้วตามด้วยหมายเลข ถ้าชื่อฟิสิคัลวอลุ่มระบุด้วยคำสั่ง **iostat** เฉพาะชื่อเหล่านั้นที่ระบุจะแสดงขึ้น

% tm_act
บ่งชี้เปอร์เซ็นต์เวลาที่ฟิสิคัลดิสก์แอ็คทีฟ (การใช้ประโยชน์จากแบนด์วิธ สำหรับไดร์ฟ) หรืออีกนัยหนึ่ง จำนวนคำร้องขอเวลาดิสก์ คงค้างอยู่ ไดร์ฟจะแอ็คทีฟในระหว่างที่ถ่ายโอนข้อมูลและการประมวลผลคำสั่ง เช่น การค้นหาตำแหน่งใหม่ เปอร์เซ็นต์ "เวลาที่ดิสก์แอ็คทีฟ" เป็นสัดส่วนโดยตรงกับ contention รีซอร์สและเป็นสัดส่วนต่อผลการทำงาน เนื่องจากดิสก์ใช้เพิ่มขึ้น ผลการทำงานจึงลดลงและเวลาตอบสนองเพิ่มขึ้น โดยทั่วไป เมื่อการใช้ประโยชน์มีค่าเกิน 70 เปอร์เซ็นต์ การประมวลผลจะรอนานกว่าสำหรับ I/O เพื่อเสร็จสิ้น เนื่องจากการประมวลผล UNIX บล็อก (หรือ sleep) ขณะที่รอคำร้องขอ I/O ให้เสร็จสิ้น มองหาเวลาที่ไม่ว่าง เปรียบเทียบกับเวลาที่ไม่ทำงานของไดร์ฟ การย้ายข้อมูลจากไดร์ฟที่ไม่ว่างไปยังไดร์ฟที่ไม่ทำงานสามารถช่วยบรรเทาอาการ คอขวดของดิสก์ การเพจไปและจากดิสก์จะสร้างโหลด I/O

Kbps บ่งชี้จำนวนของข้อมูลที่ถ่ายโอน (อ่านหรือเขียน) ลงในไดร์ฟ ในหน่วย KB ต่อวินาที ซึ่งเป็นผลรวมของ `Kb_read` plus `Kb_wrtn` หารด้วยวินาทีในช่วงของการรายงาน

tps บ่งชี้จำนวนของการถ่ายโอนต่อวินาทีซึ่งออกคำสั่งไปยังฟิสิคัลดิสก์ การถ่ายโอนคือคำร้องขอ I/O ผ่านระดับไดร์เวอร์อุปกรณ์ไปยังฟิสิคัลดิสก์ คำร้องขอโลจิคัลจำนวนมากสามารถรวมกันเป็นหนึ่งคำร้องขอ I/O ไปยังดิสก์ได้ การถ่ายโอนคือขนาดที่ไม่สามารถพิจารณาได้

Kb_read
รายงานข้อมูลทั้งหมด (ในหน่วย KB) ที่อ่านจากฟิสิคัลวอลุ่มในระหว่างช่วงเวลา ของการวัดค่า

Kb_wrtn
แสดงจำนวนข้อมูล (ในหน่วย KB) ที่เขียนไปยังฟิสิคัลวอลุ่ม ในระหว่างช่วงเวลาที่วัดค่า

ไม่มีค่าที่ยอมรับได้สำหรับฟิลด์ข้างต้นใดๆ เนื่องจากข้อมูลสถิติเกี่ยวข้องกับคุณสมบัติของแอ็พพลิเคชันมากเกินไป คอนฟิกูเรชันระบบ และชนิดของดิสก์ไดร์ฟและอะแด็ปเตอร์ ดังนั้น เมื่อคุณกำลังประมาณการข้อมูล ให้มองค่ารูปแบบและความสัมพันธ์ ความสัมพันธ์ส่วนใหญ่ จะอยู่ระหว่างการใช้ประโยชน์จากดิสก์ (`%tm_act`) และอัตราโอนย้ายข้อมูล (`tps`)

หากต้องการสรุปค่าที่ถูกต้องจากข้อมูลนี้ คุณต้องเข้าใจถึงรูปแบบการเข้าถึงข้อมูลดิสก์ของแอ็พพลิเคชัน เช่น ลำดับ สุ่ม หรือการรวม เช่นเดียวกับชนิดของฟิสิคัลดิสก์ไดร์ฟ และอะแด็ปเตอร์บนระบบ ตัวอย่างเช่น ถ้าแอ็พพลิเคชันอ่าน/เขียนตามลำดับ คุณควรคาดหวังอัตราการถ่ายโอนดิสก์สูง (`Kbps`) เมื่อคุณมีอัตราดิสก์ที่ไม่ว่างสูง (`%tm_act`) คอลัมน์ `Kb_read` และ `Kb_wrtn` สามารถยืนยันการเข้าใจของลักษณะการทำงานของอ่าน/เขียนแอ็พพลิเคชัน อย่างไรก็ตาม คอลัมน์เหล่านี้ไม่มีข้อมูลแสดงบนรูปแบบการเข้าถึงข้อมูล

โดยทั่วไป คุณไม่จำเป็นต้องรู้เกี่ยวกับอัตราการไม่ว่างของดิสก์สูง (`%tm_act`) ตราบเท่าที่อัตราการถ่ายโอนดิสก์ (`Kbps`) ยังคงสูงอยู่ อย่างไรก็ตาม ถ้าคุณได้รับอัตราการไม่ว่างของดิสก์สูงและอัตราการถ่ายโอนดิสก์ต่ำ คุณอาจมีโลจิคัลวอลุ่มที่แฟรกเมนต์ระบบไฟล์ หรือไฟล์เดี่ยวๆ

การกล่าวถึงดิสก์โลจิคัลวอลุ่ม และผลการทำงานของระบบไฟล์ในบางครั้งจะนำไปสู่ข้อสรุปที่ว่า มีไดร์ฟจำนวนมากที่คุณมีอยู่บนระบบของคุณ ผลการทำงานของดิสก์ I/O จะดีขึ้น ข้อสรุปนี้ไม่เป็นจริง เนื่องจาก มีข้อจำกัดเกี่ยวกับจำนวนของข้อมูลที่สามารถจัดการได้โดยดิสก์อะแด็ปเตอร์ ดิสก์อะแด็ปเตอร์ ยังสามารถทำให้เกิดคอขวดได้ ถ้าดิสก์ไดร์ฟของคุณทั้งหมดคือหนึ่งดิสก์อะแด็ปเตอร์ และระบบไฟล์ที่เป็นที่นิยมของคุณอยู่บนฟิสิคัลวอลุ่มที่แยกจากัน คุณอาจได้รับประโยชน์จากการใช้อะแด็ปเตอร์ดิสก์จำนวนมาก การปรับปรุงผลการทำงานจะขึ้นอยู่กับ ชนิดของการเข้าถึง

หากต้องการดูอะแด็ปเตอร์เฉพาะที่มีอยู่ทั่วไป ให้ใช้คำสั่ง **iostat** และเพิ่ม `Kbps` ซึ่งเป็นจำนวนสำหรับดิสก์ที่พ่วงกับดิสก์อะแด็ปเตอร์โดยเฉพาะ สำหรับผลการทำงานสูงสุดโดยรวม จำนวนของอัตราการถ่ายโอน (`Kbps`) ต้องต่ำกว่าอัตราทรูพุตดิสก์อะแด็ปเตอร์ในกรณีส่วนใหญ่ ให้ใช้ 70 เปอร์เซ็นต์ของอัตราทรูพุต ในระบบปฏิบัติการ AIX อ็อพชัน **-a** หรือ **-A** จะแสดงข้อมูลนี้

การประเมินค่าผลการทำงานดิสก์ด้วยคำสั่ง vmstat

หากต้องการพิสูจน์ว่า ระบบคือ I/O ที่เชื่อมโยงไว้จึงเป็นสิ่งที่ดีกว่าในการใช้คำสั่ง **iostat**

อย่างไรก็ตาม คำสั่ง **vmstat** จะสามารถชี้ไปยังทิศทางนั้น โดยดูที่คอลัมน์ *wa* ตามที่ได้กล่าวถึงใน “คำสั่ง vmstat” ในหน้า 112 ตัวบ่งชี้อื่นๆ สำหรับ I/O ที่โยงไว้คือ:

- ส่วนของ `disk xfer` ของเอาต์พุต **vmstat**
  หากต้องการแสดงข้อมูลสถิติ เกี่ยวกับโลจิคัลดิสก์ (อนุญาตให้ใช้มากสุดสี่ดิสก์) ให้ใช้คำสั่งต่อไปนี้:

```
# vmstat hdisk0 hdisk1 1 8
kthr     memory             page              faults        cpu     disk xfer
----  ----------  ------------------------ ------------ ----------- ------
```

```
r  b   avm   fre  re  pi  po  fr  sr  cy  in   sy  cs  us sy id wa  1 2 3 4
0  0  3456 27743   0   0   0   0   0   0 131  149  28   0  1 99  0  0 0
0  0  3456 27743   0   0   0   0   0   0 131   77  30   0  1 99  0  0 0
1  0  3498 27152   0   0   0   0   0   0 153 1088  35   1 10 87  2  0 11
0  1  3499 26543   0   0   0   0   0   0 199 1530  38   1 19  0 80  0 59
0  1  3499 25406   0   0   0   0   0   0 187 2472  38   2 26  0 72  0 53
0  0  3456 24329   0   0   0   0   0   0 178 1301  37   2 12 20 66  0 42
0  0  3456 24329   0   0   0   0   0   0 124   58  19   0  0 99  0  0 0
0  0  3456 24329   0   0   0   0   0   0 123   58  23   0  0 99  0  0 0
```

ส่วนของ *disk xfer* จะจัดเตรียมจำนวนของการถ่ายโอนต่อวินาทีเพื่อระบุฟิสิคัลวอลุ่ม ที่เกิดขึ้นในช่วงเวลาของกลุ่มตัวอย่าง คุณสามารถระบุหนึ่งในสี่ ชื่อฟิสิคัลวอลุ่มได้ ข้อมูลสถิติการถ่ายโอนจะกำหนดไว้สำหรับไดร์ฟเฉพาะแต่ละไดร์ฟ ตามที่ได้ระบุไว้ ซึ่งจำนวนนี้จะแสดงคำร้องขอฟิสิคัลอุปกรณ์ ซึ่งไม่ได้หมายความถึงจำนวนของข้อมูลที่จะถูกอ่านหรือเขียน คำร้องขอเชิงตรรกะทั้งหลาย สามารถรวมกันให้เป็นหนึ่งคำร้องขอแบบฟิสิคัล

- คอลัมน์ *in* ของเอาต์พุต **vmstat**

  คอลัมน์นี้ แสดงจำนวนของฮาร์ดแวร์หรืออุปกรณ์อินเตอร์รัปต์ (ต่อวินาที) ที่เฝ้าดูผ่านช่วงเวลาของการวัด ตัวอย่างของอินเตอร์รัปต์คือ ความสมบูรณ์ของคำร้องขอดิสก์ และอินเตอร์รัปต์ของนาฬิกาด้วยเวลา 10 มิลลิวินาที ในเวลาต่อมา 100 ครั้งต่อวินาที ฟิลด์ *in* จะยังคงมีค่ามากกว่า 100 แต่คำสั่ง **vmstat** ยังจัดเตรียมเอาต์พุตโดยละเอียดเพิ่มเติมเกี่ยวกับอินเตอร์รัปต์ของระบบ

- เอาต์พุต **vmstat −i**

  พารามิเตอร์ *−i* แสดงจำนวนของอินเตอร์รัปต์ที่ใช้โดยแต่ละอุปกรณ์เนื่องจากการเริ่มต้นทำงานของระบบ แต่ ด้วยการเพิ่มช่วงเวลา และพารามิเตอร์จำนวน ข้อมูลสถิติ ตั้งแต่การเริ่มต้นทำงานถูกแสดงอยู่ใน stanza แรก ทุกๆ stanza คือข้อมูลสถิติเกี่ยวกับช่วงเวลาที่สแกน

```
# vmstat -i 1 2
priority level    type   count module(handler)
    0       0   hardware     0 i_misc_pwr(a868c)
    0       1   hardware     0 i_scu(a8680)
    0       2   hardware     0 i_epow(954e0)
    0       2   hardware     0 /etc/drivers/ascsiddpin(189acd4)
    1       2   hardware   194 /etc/drivers/rsdd(1941354)
    3      10   hardware 10589024 /etc/drivers/mpsdd(1977a88)
    3      14   hardware 101947 /etc/drivers/ascsiddpin(189ab8c)
    5      62   hardware 61336129 clock(952c4)
   10      63   hardware 13769 i_softoff(9527c)
priority level    type   count module(handler)
    0       0   hardware     0 i_misc_pwr(a868c)
    0       1   hardware     0 i_scu(a8680)
    0       2   hardware     0 i_epow(954e0)
    0       2   hardware     0 /etc/drivers/ascsiddpin(189acd4)
    1       2   hardware     0 /etc/drivers/rsdd(1941354)
    3      10   hardware    25 /etc/drivers/mpsdd(1977a88)
    3      14   hardware     0 /etc/drivers/ascsiddpin(189ab8c)
    5      62   hardware   105 clock(952c4)
   10      63   hardware     0 i_softoff(9527c)
```

หมายเหตุ: เอาต์พุต จะแตกต่างจากระบบต่อระบบ ซึ่งขึ้นอยู่กับคอนฟิกูเรชันฮาร์ดแวร์ และซอฟต์แวร์ (ตัวอย่างเช่น การอินเตอร์รัปต์นาฬิกาอาจไม่แสดงอยู่ในเอาต์พุต **vmstat −i** แม้ว่าจะถูกอธิบายอยู่ใต้คอลัมน์ *in* ในเอาต์พุต **vmstat** ปกติ) ตรวจสอบจำนวนสูงสุดในคอลัมน์ *count* และการตรวจสอบสาเหตุที่โมดูลนี้ เรียกใช้งานอินเตอร์รัปต์จำนวนมาก

การประเมินค่าผลการทำงานดิสก์ด้วยคำสั่ง sar

คำสั่ง **sar** คือคำสั่ง UNIX มาตรฐานที่ใช้เพื่อเก็บรวบรวมข้อมูลสถิติเกี่ยวกับระบบ

ด้วยอ็อพชันที่มีมากมาย คำสั่ง **sar** จะจัดเตรียมการเคียวรี่ การเพจ TTY และข้อมูลสถิติอื่นๆ จำนวนมาก อ็อพชัน **sar –d** จะสร้างสถิติ I/O ดิสก์แบบเรียลไทม์

```
# sar -d 3 3

AIX konark 3 4 0002506F4C00    08/26/99

12:09:50     device    %busy    avque    r+w/s   blks/s   avwait   avserv

12:09:53     hdisk0      1      0.0        0        5      0.0      0.0
             hdisk1      0      0.0        0        1      0.0      0.0
                cd0      0      0.0        0        0      0.0      0.0

12:09:56     hdisk0      0      0.0        0        0      0.0      0.0
             hdisk1      0      0.0        0        1      0.0      0.0
                cd0      0      0.0        0        0      0.0      0.0

12:09:59     hdisk0      1      0.0        1        4      0.0      0.0
             hdisk1      0      0.0        0        1      0.0      0.0
                cd0      0      0.0        0        0      0.0      0.0


Average      hdisk0      0      0.0        0        3      0.0      0.0
             hdisk1      0      0.0        0        1      0.0      0.0
                cd0      0      0.0        0        0      0.0      0.0
```

ฟิลด์ที่แสดงด้วยคำสั่ง **sar –d** มีดังต่อไปนี้:

%busy ส่วนของเวลาที่อุปกรณ์ไม่ว่างให้บริการคำร้องขอที่ถ่ายโอน ซึ่งเป็นค่าเดียวกันกับคอลัมน์ `%tm_act` ในรายงานคำสั่ง **iostat**

avque จำนวนเฉลี่ยของคำร้องที่ค้างอยู่จากอะแด็ปเตอร์กับอุปกรณ์ในช่วงเวลานั้น ซึ่งอาจมีการดำเนินการ I/O เพิ่มเติมในคิวของไดร์เวอร์ อุปกรณ์ จำนวนนี้จะเป็นตัวบ่งชี้ที่ดี หากมีคอขวดของ I/O อยู่

r+w/s จำนวนของการถ่ายโอนอ่าน/เขียนจากหรือไปยังอุปกรณ์ ซึ่งเป็นค่าเดียวกันกับ `tps` ในรายงานคำสั่ง **iostat**

blks/s จำนวนไบต์ที่ถ่ายโอนในหน่วย 512 ไบต์

avwait จำนวนเฉลี่ยของการทำรายการที่รอการบริการ (ความยาวของคิว) เวลาเฉลี่ย (ในหน่วยมิลลิวินาที) ที่ถ่ายโอนคำร้องขอที่รออยู่บนคิว สำหรับอุปกรณ์ จำนวนนี้ไม่ได้รายงานอยู่ในปัจจุบันและแสดงค่า 0.0 ตามค่าดีฟอลต์

avserv จำนวนของมิลลิวินาทีต่อการค้นหาเฉลี่ย เวลาเฉลี่ย (ในหน่วยมิลลิวินาที) เพื่อให้บริการถ่ายโอนคำร้องขอแต่ละครั้ง (ประกอบด้วยการค้นหา เวลาแฝงโดยรอบ และเวลาในการถ่ายโอน) สำหรับอุปกรณ์ จำนวนนี้ไม่ได้รายงานอยู่ในปัจจุบัน และแสดงค่า 0.0 ตามค่าดีฟอลต์

การประเมินค่าการแตกแฟรกเมนต์โลจิคัลวอลุ่มด้วยคำสั่ง lslv

คำสั่ง **lslv** จะแสดงการแตกแฟรกเมนต์โลจิคัลวอลุ่ม ระหว่างข้อมูลอื่นๆ

หากต้องการตรวจสอบการแตกแฟรกเมนต์โลจิคัลวอลุ่ม ให้ใช้คำสั่ง **lslv –l** *lvname* ดังต่อไปนี้:

```
# lslv -l hd2
hd2:/usr
PV               COPIES        IN BAND       DISTRIBUTION
hdisk0           114:000:000   22%           000:042:026:000:046
```

เอาต์พุตของ COPIES จะแสดงโลจิคัลวอลุ่ม hd2 ที่มีเพียงสำเนาเดียวเท่านั้น IN BAND จะแสดงนโยบายภายในซึ่งเป็นแอ็ตทริบิวต์ของโลจิคัลวอลุ่ม ที่ตามมาภายหลัง เปอร์เซ็นต์ที่สูงกว่าจะมีการจัดสรรที่มีประสิทธิภาพ ที่ดีกว่า แต่ละโลจิคัลวอลุ่มจะมีนโยบายภายใน ถ้าระบบปฏิบัติการ ไม่สามารถปฏิบัติตามข้อกำหนดนี้ ระบบปฏิบัติการจะเลือกทางที่ดีที่สุดเพื่อให้ตรงกับข้อกำหนด สำหรับตัวอย่าง มีโลจิคัลพาร์ติชัน (LP) ทั้งหมด 114 พาร์ติชัน 42 LP จะตั้งอยู่บนจุดศูนย์กลาง 26 LP จะอยู่บนศูนย์กลาง และ 46 LP จะอยู่บนขอบภายใน เนื่องจากนโยบายภายในของ โลจิคัลวอลุ่มอยู่บนศูนย์กลาง ในกลุ่มคือ 22 เปอร์เซ็นต์ (26 / (42+26+46) DISTRIBUTION จะแสดงวิธีการที่ฟิสิคัลพาร์ติชันจะถูกวางอยู่ในส่วนของนโยบายภายในแต่ละส่วน นั่นคือ:

```
edge : middle : center : inner-middle : inner-edge
```

โปรดดู “ตำแหน่งบนฟิสิคัลวอลุ่ม” ในหน้า 221 สำหรับข้อมูลเพิ่มเติมเกี่ยวกับ การกำหนดตำแหน่งฟิสิคัลพาร์ติชัน

การประเมินผลการกำหนดตำแหน่งฟิสิคัลของข้อมูลด้วยคำสั่ง lslv

ถ้าเวิร์กโหลดแสดงระดับของการพึ่งพา I/O ที่สำคัญ คุณสามารถตรวจสอบการกำหนดตำแหน่งฟิสิคัลของไฟล์บนดิสก์ เพื่อพิจารณาว่า การจัดการใหม่ที่ระดับบางระดับจะต้องได้ผลตอบแทนในการปรับปรุง

หากต้องการดูการกำหนดตำแหน่งของพาร์ติชันของโลจิคัลวอลุ่ม hd11 ภายในฟิสิคัลวอลุ่ม hdisk0 ให้ใช้คำสั่งต่อไปนี้:

```
# lslv -p hdisk0 hd11
hdisk0:hd11:/home/op
USED  USED  USED  USED  USED  USED  USED  USED  USED  USED     1-10
USED  USED  USED  USED  USED  USED  USED                      11-17

USED  USED  USED  USED  USED  USED  USED  USED  USED  USED    18-27
USED  USED  USED  USED  USED  USED  USED                      28-34

USED  USED  USED  USED  USED  USED  USED  USED  USED  USED    35-44
USED  USED  USED  USED  USED  USED                            45-50

USED  USED  USED  USED  USED  USED  USED  USED  USED  USED    51-60
0052  0053  0054  0055  0056  0057  0058                      61-67

0059  0060  0061  0062  0063  0064  0065  0066  0067  0068    68-77
0069  0070  0071  0072  0073  0074  0075                      78-84
```

มองหาส่วนที่เหลือของ hd11 บน hdisk1 ด้วยคำสั่งต่อไปนี้:

```
# lslv -p hdisk1 hd11
hdisk1:hd11:/home/op
0035  0036  0037  0038  0039  0040  0041  0042  0043  0044    1-10
0045  0046  0047  0048  0049  0050  0051                     11-17

USED  USED  USED  USED  USED  USED  USED  USED  USED  USED   18-27
USED  USED  USED  USED  USED  USED  USED                     28-34

USED  USED  USED  USED  USED  USED  USED  USED  USED  USED   35-44
USED  USED  USED  USED  USED  USED                           45-50
```

```
0001  0002  0003  0004  0005  0006  0007  0008  0009  0010   51-60
0011  0012  0013  0014  0015  0016  0017                     61-67

0018  0019  0020  0021  0022  0023  0024  0025  0026  0027   68-77
0028  0029  0030  0031  0032  0033  0034                     78-84
```

จากด้านบนถึงด้านล่าง ห้าบล็อกที่แสดงขอบ กึ่งกลาง ศูนย์กลาง ภายในกึ่งกลาง และภายในขอบ ตามลำดับ

- USED บ่งชี้ว่า ฟิสิคัลพาร์ติชันที่ตำแหน่งนี้ ถูกใช้โดยโลจิคัลวอลุ่มที่นอกเหนือจากที่ระบุไว้ ตัวเลขบ่งชี้ถึง หมายเลขโลจิคัลพาร์ติชันของโลจิคัลวอลุ่มที่ระบุด้วยคำสั่ง **lslv -p**
- FREE บ่งชี้ว่า ฟิสิคัลพาร์ติชันนี้ไม่ได้ถูกใช้โดย โลจิคัลวอลุ่มใดๆ การแตกแฟรกเมนต์ของโลจิคัลวอลุ่มเกิดขึ้น หากโลจิคัลพาร์ติชันไม่ได้ต่อเนื่องกันระหว่างดิสก์
- ฟิสิคัลพาร์ติชัน STALE คือฟิสิคัลพาร์ติชันที่มีข้อมูล ที่คุณไม่สามารถใช้ได้ คุณยังสามารถมองเห็นฟิสิคัลพาร์ติชัน STALE ได้ด้วยคำสั่ง **lspv -m** ฟิสิคัลพาร์ติชันที่มีเครื่องหมาย STALE ต้องถูกอัพเดตให้มีข้อมูลเดียวกันกับฟิสิคัลพาร์ติชันที่ถูกต้อง กระบวนการนี้เรียกว่า การประสานเวลาใหม่ด้วยคำสั่ง **syncvg** ซึ่งสามารถกระทำได้ในเวลาที่ vary-on หรือสามารถเริ่มต้นได้ทุกเวลาที่ระบบกำลังรันอยู่ จนกว่าพาร์ติชัน STALE จะถูกบันทึกใหม่ด้วยข้อมูลที่ถูกต้อง ซึ่งไม่ได้ใช้เพื่อตอบกลับคำร้องขอการอ่าน หรือเขียนลงบนคำร้องขอการเขียน

ในตัวอย่างก่อนหน้านี้โลจิคัลวอลุ่ม hd11 คือแฟรกเมนต์ภายใน ฟิสิคัลวอลุ่ม hdisk1 ซึ่งมีโลจิคัลพาร์ติชันแรกอยู่ภายในกึ่งกลาง และส่วนภายในของ hdisk1 ขณะที่โลจิคัลพาร์ติชัน 35-51 จะอยู่นอกส่วนนั้น เวิร์กโหลดที่เข้าถึง hd11 โดยการสุ่มจะพบกับช่วงเวลารอ I/O ที่ไม่จำเป็นตราบนานเท่าที่การค้นหาอาจต้องการโลจิคัลวอลุ่ม hd11 รายงานเหล่านี้ ยังบ่งชี้ว่า ไม่มีฟิสิคัลพาร์ติชันที่ว่างใน hdisk0 หรือ hdisk1 อย่างใดอย่างหนึ่ง

การประเมินค่าไฟล์ที่กำหนดตำแหน่งด้วยคำสั่ง fileplace

หากต้องการดูวิธีที่ไฟล์ถูกคัดลอกไว้ก่อนหน้านี้ นั่นคือ big1 จะถูกเก็บอยู่บนดิสก์ เราสามารถใช้คำสั่ง **fileplace** ได้ คำสั่ง **fileplace** จะแสดงการกำหนดตำแหน่งของบล็อกของไฟล์ ภายในโลจิคัลวอลุ่ม หรือภายในฟิสิคัลวอลุ่มเพิ่มเติม

หากต้องการพิจารณาว่า คำสั่ง **fileplace** ได้ถูกติดตั้งและพร้อมใช้งานแล้ว ให้รันคำสั่งต่อไปนี้:

```
# lslpp -lI perfagent.tools
```

ใช้ คำสั่งต่อไปนี้:

```
# fileplace -pv big1

File: big1  Size: 3554273 bytes  Vol: /dev/hd10
Blk Size: 4096  Frag Size: 4096  Nfrags: 868   Compress: no
Inode: 19  Mode: -rwxr-xr-x  Owner: hoetzel  Group: system

  Physical Addresses (mirror copy 1)                          Logical Fragment
  ----------------------------------                          ----------------
  0001584-0001591  hdisk0     8 frags    32768 Bytes,   0.9%    0001040-0001047
  0001624-0001671  hdisk0    48 frags   196608 Bytes,   5.5%    0001080-0001127
  0001728-0002539  hdisk0   812 frags  3325952 Bytes,  93.5%    0001184-0001995

  868 frags over space of 956 frags:  space efficiency = 90.8%
  3 fragments out of 868 possible:  sequentiality = 99.8%
```

ตัวอย่างนี้ แสดงว่ามีการแตกแฟรกเมนต์จำนวนเล็กน้อยภายในฟิลด์ และเป็นช่องว่างขนาดเล็ก ดังนั้น เราจึงสามารถสรุปได้ว่า การจัดเรียงดิสก์ของ bigl ไม่ได้กระทบกับเวลาในการอ่านตามลำดับที่สำคัญ ยิ่งไปกว่านั้น การกำหนดว่า (ซึ่งได้สร้างขึ้นเมื่อเร็วๆ นี้)ไฟล์ขนาด 3.5 MB พบกับการแตกแฟรกเมนต์ขนาดเล็กนี้ ซึ่งปรากฏขึ้นที่ระบบไฟล์โดยทั่วไปจะไม่ถูกแตกแฟกเมนต์โดยเฉพาะ

ในบางครั้ง ส่วนของไฟล์อาจไม่ได้ถูกแม็พไว้กับบล็อกใดๆ ในวอลุ่ม พื้นที่เหล่านี้จะถูกเติมด้วยศูนย์ โดยระบบไฟล์ พื้นที่เหล่านี้จะแสดงเป็นบล็อกโลจิคัล unallocated ไฟล์ที่มีช่องเหล่านี้จะแสดงขนาดไฟล์ที่มีจำนวนไบต์ใหญ่กว่าที่จองไว้ (นั่นคือ คำสั่ง **ls –l** จะแสดงขนาดใหญ่สุด ซึ่งคำสั่ง **du** จะแสดงขนาดเล็กสุด หรือจำนวนของบล็อกไฟล์ที่จองไว้บนดิสก์)

คำสั่ง **fileplace** จะอ่านรายการของบล็อกของไฟล์จากโลจิคัลวอลุ่ม ถ้าไฟล์นั้นเป็นไฟล์ใหม่ ข้อมูลอาจไม่ได้อยู่บนดิสก์ ให้ใช้คำสั่ง **sync** เพื่อล้างข้อมูล และคำสั่ง **fileplace** จะไม่แสดงรีโมตไฟล์ NFS (ยกเว้นว่า คำสั่งจะรันอยู่บนเซิร์ฟเวอร์)

หมายเหตุ: ถ้าไฟล์ได้ถูกสร้างขึ้นโดยการค้หาตำแหน่งที่หลากหลาย และการเขียนกระจายเร็กคอร์ดแบบกว้างๆ เฉพาะเพจที่มีเร็กคอร์ดเท่านั้นที่จะสนใจพื้นที่บนดิสก์ และปรากฏอยู่บนรายงาน **fileplace** ระบบไฟล์ไม่ได้กรอกข้อมูลเพจ ที่เกิดขึ้นโดยอัตโนมัติเมื่อสร้างไฟล์ อย่างไรก็ตาม ถ้าไฟล์ถูกอ่านตามลำดับ (ตัวอย่างเช่น ด้วยคำสั่ง **cp** หรือ **tar**) พื้นที่ระหว่างเร็กคอร์ดจะถูกอ่านเป็นไบนารีศูนย์ ดังนั้น เอาต์พุตของคำสั่ง **cp** สามารถมีขนาดใหญ่กว่าไฟล์อินพุต ผ่านข้อมูลเดียวกัน

ประสิทธิผลของพื้นที่ว่างและการจัดลำดับได้:

ประสิทธิผลของพื้นที่ว่างยิ่งสูงหมายความว่าไฟล์มีการแบ่งเฟรกเมนต์น้อยลง และให้การเข้าถึงไฟล์ตามลำดับที่ดีขึ้น การจัดลำดับได้ยิ่งสูงบ่งชี้ว่า ไฟล์มีการจัดสรรต่อเนื่องกันมากขึ้น และอาจจะให้ การเข้าถึงไฟล์ตามลำดับที่ดีขึ้น

ประสิทธิผลของพื้นที่ว่าง =

จำนวนเฟรกเมนต์ทั้งหมดที่ใช้สำหรับการจัดเก็บไฟล์ / (ที่อยู่ฟิสิคัลของเฟรกเมนต์ ที่ใหญ่ที่สุด – ที่อยู่ฟิสิคัลของเฟรกเมนต์ที่เล็กที่สุด + 1)

การจัดลำดับได้ =

(จำนวนเฟรกเมนต์ทั้งหมด – จำนวนของเฟรกเมนต์ที่จัดกลุ่ม +1) / จำนวน เฟรกเมนต์ทั้งหมด

ถ้าคุณพบว่าค่าการจัดลำดับได้หรือประสิทธิผลของพื้นที่ว่างเริ่มต่ำ คุณสามารถใช้คำสั่ง **reorgvg** เพื่อปรับปรุงการใช้ประโยชน์และ ประสิทธิผลของโลจิคัลวอลุ่ม (โปรดดู “การจัดระเบียบโลจิคัลวอลุ่มใหม่” ในหน้า 227) เมื่อต้องการปรับปรุงการใช้ประโยชน์และประสิทธิผลของระบบไฟล์ ให้ดู “การจัดระเบียบระบบไฟล์ใหม่” ในหน้า 267

ในตัวอย่างนี้ ที่อยู่ฟิสิคัลของเฟรกเมนต์ที่ใหญ่ที่สุด – ที่อยู่ฟิสิคัลของเฟรกเมนต์ ที่เล็กที่สุด + 1 คือ: 0002539 – 0001584 + 1 = 956 เฟรกเมนต์; เฟรกเมนต์ ทั้งหมดที่ใช้คือ: 8 + 48 + 812 = 868; ประสิทธิผลของพื้นที่ว่างคือ 868 / 956 (90.8 เปอร์เซ็นต์); การจัดลำดับได้คือ (868 – 3 + 1) / 868 = 99.8 เปอร์เซ็นต์

เนื่องจากจำนวนทั้งหมดของเฟรกเมนต์ที่ใช้สำหรับการจัดเก็บไฟล์ไม่รวม ที่ตั้งบล็อกทางอ้อม แต่ที่อยู่ฟิสิคัลรวม ดังนั้น ประสิทธิผลของพื้นที่ว่าง จึงไม่มีทางเป็น 100 เปอร์เซ็นต์สำหรับไฟล์ที่ใหญ่กว่า 32 KB แม้ว่าไฟล์อยู่บน เฟรกเมนต์ที่ต่อเนื่องกัน

การประเมินผล I/O พื้นที่การเพจด้วยคำสั่ง vmstat

รายงาน **vmstat** บ่งชี้ถึงจำนวนของพื้นที่การเพจ I/O ที่เข้าแทนที่

I/O ไปและจากพื้นที่การเพจจะถูกสุ่ม โดยส่วใหญ่แล้วหนึ่งเพจในเวลาเดียวกัน ตัวอย่างต่อไปนี้แสดงกิจกรรมการเพจที่เกิดขึ้น ในระหว่างการคอมไพล์ C ในเครื่องที่ได้ยุบอยู่โดยใช้คำสั่ง **rmss** คอลัมน์ *pi* และ *po* (พื้นที่การเพจสำหรับ เพจเข้าและพื้นที่

การเพจสำหรับเพจออก) จะแสดงจำนวนของพื้นที่การเพจ I/O (ในรูปของเพจขนาด 4096 ไบต์) ระหว่างช่วงเวลาแต่ละช่วง ช่วงล่ะ 5 วินาที รายงานแรก (สรุปเนื่องจากระบบรีบูต) ได้ถูกลบออกแล้ว โปรดสังเกตว่า กิจกรรมการเพจจะเกิดขึ้นในการแยกเป็นแผ่น

```
# vmstat 5 8
kthr     memory             page              faults        cpu
----- ----------- ------------------------ ------------ -----------
 r  b   avm   fre  re  pi  po  fr   sr  cy  in   sy  cs us sy id wa
 0  1 72379   434   0   0   0   0    2   0 376  192 478  9  3 87  1
 0  1 72379   391   0   8   0   0    0   0 631 2967 775 10  1 83  6
 0  1 72379   391   0   0   0   0    0   0 625 2672 790  5  3 92  0
 0  1 72379   175   0   7   0   0    0   0 721 3215 868  8  4 72 16
 2  1 71384   877   0  12  13  44  150   0 662 3049 853  7 12 40 41
 0  2 71929   127   0  35  30 182  666   0 709 2838 977 15 13  0 71
 0  1 71938   122   0   0   8  32  122   0 608 3332 787 10  4 75 11
 0  1 71938   122   0   0   0   3   12   0 611 2834 733  5  3 75 17
```

รายงาน "ก่อนและหลัง" **vmstat -s** ต่อไปนี้ แสดงการรวมกันของกิจกรรมการเพจ โปรดจำไว้ว่า นี่คือพื้นที่การเพจสำหรับการเพจเข้า และพื้นที่การเพจสำหรับการเพจออก ซึ่งแสดง I/O ของพื้นที่การเพจที่เป็นจริง เพจเข้า (ไม่ผ่านการรับรอง) และเพจออกจะรายงาน I/O ทั้งหมด นั่นคือ I/O สำหรับพื้นที่การเพจ และไฟล์ I/O ปกติจะดำเนินการโดยกลไกการเพจ รายงานจะถูกแก้ไข เพื่อลบบรรทัดที่ไม่เกี่ยวข้องกับการอภิปรายนี้

# vmstat -s # before	**# vmstat -s # after**
6602 เพจเข้า 3948 เพจออก **544 เพจเข้าสำหรับพื้นที่การเพจ 1923 เพจออกสำหรับพื้นที่การเพจ** ทั้งหมด 0 ที่ต้องเรียกคืน	7022 เพจเข้า 4146 เพจออก **689 เพจเข้าสำหรับพื้นที่การเพจ 2032 เพจออกสำหรับพื้นที่การเพจ** ทั้งหมด 0 ที่ต้องเรียกคืน

ความเป็นจริงที่ว่า เพจเข้าสำหรับพื้นที่การเพจที่มากกว่าเพจออก ที่เกิดขึ้นในระหว่างการคอมไพล์จะถูกแนะนำว่า เราจะทำให้ระบบเล็กลงเท่ากับจุดที่ เกิดการแกว่งไปแกว่งมา เพจบางเพจจะถูกเพจใหม่ เนื่องจากกรอบได้ถูกนำไปใช้ ก่อนที่การใช้จะเสร็จสิ้น

การประเมินค่าดิสก์ I/O ทั้งหมดด้วยคำสั่ง vmstat

เทคนิคที่ได้กล่าวถึงยังสามารถนำมาใช้เพื่อประเมินค่าโหลดดิสก์ I/O ที่สร้างขึ้นโดยโปรแกรม

ถ้าระบบใช้งานอย่างอื่นตามลำดับต่อไปนี้:

```
# vmstat -s >statout
# testpgm
# sync
# vmstat -s >> statout
# egrep "ins|outs" statout
```

อัตราผลประโยชน์ก่อนและหลังรูปของการนับกิจกรรมดิสก์ที่สะสมไว้ เช่น:

```
     5698 page ins
     5012 page outs
        0 paging space page ins
       32 paging space page outs
     6671 page ins
     5268 page outs
        8 paging space page ins
      225 paging space page outs
```

ในระหว่างช่วงเวลา เมื่อรันคำสั่งนี้ (คอมไพล์ C ขนาดใหญ่) ระบบจะอ่านเพจทั้งหมด 981 เพจ (8 จากพื้นที่การเพจ) และเขียนจำนวนเพจทั้งหมด 449 เพจ (193 ไปยังพื้นที่การเพจ)

การวิเคราะห์ I/O โดยละเอียดด้วยคำสั่ง filemon

คำสั่ง **filemon** ใช้ตัวช่วยติดตาม เพื่อขอรับภาพของกิจกรรม I/O โดยละเอียดในระหว่างช่วงเวลาสำหรับเลเยอร์ของการใช้ประโยชน์จากระบบไฟล์ ซึ่งประกอบด้วยระบบไฟล์แบบโลจิคัล เซ็กเมนต์หน่วยความจำเสมือน LVM และเลเยอร์ฟิสิคัลดิสก์

คำสั่ง **filemon** สามารถนำมาใช้เพื่อเก็บรวบรวมข้อมูลเกี่ยวกับเลเยอร์ทั้งหมด หรือเลเยอร์ที่สามารถระบุด้วยอ็อพชันเลเยอร์ **-O** ค่าดีฟอลต์คือ การเก็บรวบรวมข้อมูลเกี่ยวกับ VM, LVM และฟิสิคัลเลเยอร์ ทั้งรายงานสรุป และรายงานโดยละเอียดจะถูกสร้างขึ้น เนื่องจากใช้ตัวช่วยการติดตาม คำสั่ง **filemon** สามารถรันได้โดยผู้ใช้รากหรือสมาชิกในกลุ่มระบบเท่านั้น

หากต้องการพิจารณาว่า คำสั่ง **filemon** ได้ถูกติดตั้งแล้ว และพร้อมใช้งาน ให้รันคำสั่งต่อไปนี้:

```
# lslpp -lI perfagent.tools
```

การติดตาม จะเริ่มต้นด้วยคำสั่ง **filemon** ซึ่งจะหยุดทำงานด้วยคำสั่งย่อย **trcoff** และกลับสู่การทำงานต่อด้วยคำสั่งย่อย **trcon** ในทันทีที่ยกเลิกการติดตาม คำสั่ง **filemon** จะเขียนรายงานลงใน stdout

หมายเหตุ: ซึ่งเฉพาะข้อมูลสำหรับไฟล์เหล่านั้นที่เปิดหลังจากเริ่มต้นคำสั่ง **filemon** จะถูกเก็บรวบรวมไว้ เว้นเสียแต่คุณจะระบุแฟล็ก **-u**

คำสั่ง **filemon** สามารถอ่านข้อมูลการติดตาม I/O ได้จากไฟล์ที่ระบุ แทนที่จะอ่านจากการประมวลผลการติดตาม แบบเรียลไทม์ ในกรณีนี้ รายงาน **filemon** จะสรุปกิจกรรม I/O สำหรับระบบและระยะเวลาที่แสดงโดยไฟล์การติดตาม เมธอดการประมวลผลแบบออฟไลน์นี้มีประโยชน์เมื่อมีความจำเป็นที่ต้องทำตามขั้นตอนหลังการประมวลผลไฟล์การติดตาม จากเครื่องรีโมต หรือดำเนินการเก็บรวบรวมข้อมูลการติดตามในหนึ่งครั้ง และเก็บรวบรวมข้อมูลหลังการประมวลผลในเวลาอื่น

คำสั่ง **trcrpt -r** ต้องเรียกใช้งานบนไฟล์บันทึกการติดตาม และเปลี่ยนทิศทางไปยังไฟล์อื่นได้ดังที่แสดงต่อไปนี้:

```
# gennames > gennames.out
# trcrpt -r   trace.out > trace.rpt
```

ณ จุดนี้ ไฟล์บันทึกการติดตามที่ปรับเปลี่ยนแล้วจะถูกป้อนลงในคำสั่ง **filemon** เพื่อออกรายงานเกี่ยวกับกิจกรรม I/O ที่ดักจับโดยเซสชันการติดตามที่บันทึกไว้ก่อนหน้านี้ ดังที่แสดงต่อไปนี้:

```
# filemon -i trace.rpt -n gennames.out | pg
```

สำหรับตัวอย่างนี้ คำสั่ง **filemon** จะอ่านเหตุการณ์การติดตามระบบไฟล์ จากอินพุตไฟล์ `trace.rpt` เนื่องจากข้อมูลการติดตามได้ถูกดักจับไว้แล้วบนไฟล์ คำสั่ง **filemon** จะไม่วางไว้ที่ส่วนหลังเพื่ออนุญาตให้แอ็พพลิเคชันโปรแกรมทำงานได้ หลังจากที่อ่านไฟล์ทั้งหมดแล้ว รายงานกิจกรรม I/O สำหรับหน่วยความจำเสมือน โลจิคัลวอลุ่ม และระดับของฟิสิคัลวอลุ่มจะถูกแสดงอยู่บนเอาต์พุตมาตรฐาน (ซึ่งในตัวอย่างนี้จะถูกส่งไปยังคำสั่ง **pg**)

ถ้าคำสั่ง **trace** รันพร้อมกับแฟล็ก **-C all** ให้รันคำสั่ง **trcrpt** ด้วยแฟล็ก **-C all** (โปรดดู “การจัดรูปแบบรายงานจากการติดตามเอาต์พุต -C” ในหน้า 439)

ลำดับของคำสั่งต่อไปนี้ แสดงตัวอย่างของการใช้คำสั่ง **filemon** :

```
# filemon -o fm.out -O all; cp /smit.log /dev/null ; trcstop
```

รายงานจะถูกสร้างตามลำดับนี้ในระบบที่ไม่ได้ทำงาน ดังต่อไปนี้:

```
Thu Aug 19 11:30:49 1999
System: AIX texmex Node: 4 Machine: 000691854C00

0.369 secs in measured interval
Cpu utilization:  9.0%

Most Active Files
------------------------------------------------------------------------
  #MBs  #opns   #rds   #wrs  file                volume:inode
------------------------------------------------------------------------
   0.1      1     14      0  smit.log            /dev/hd4:858
   0.0      1      0     13  null
   0.0      2      4      0  ksh.cat             /dev/hd2:16872
   0.0      1      2      0  cmdtrace.cat        /dev/hd2:16739

Most Active Segments
------------------------------------------------------------------------
  #MBs  #rpgs  #wpgs  segid  segtype             volume:inode
------------------------------------------------------------------------
   0.1     13      0   5e93  ???
   0.0      2      0   22ed  ???
   0.0      1      0   5c77  persistent

Most Active Logical Volumes
------------------------------------------------------------------------
  util  #rblk  #wblk   KB/s  volume              description
------------------------------------------------------------------------
  0.06    112      0  151.9  /dev/hd4            /
  0.04     16      0   21.7  /dev/hd2            /usr

Most Active Physical Volumes
------------------------------------------------------------------------
  util  #rblk  #wblk   KB/s  volume              description
------------------------------------------------------------------------
  0.10    128      0  173.6  /dev/hdisk0         N/A




------------------------------------------------------------------------
Detailed File Stats
------------------------------------------------------------------------

FILE: /smit.log  volume: /dev/hd4 (/)  inode: 858
opens:                  1
total bytes xfrd:       57344
reads:                  14      (0 errs)
  read sizes (bytes):   avg  4096.0 min    4096 max    4096 sdev     0.0
  read times (msec):    avg   1.709 min   0.002 max  19.996 sdev   5.092

FILE: /dev/null
opens:                  1
total bytes xfrd:       50600
writes:                 13      (0 errs)
  write sizes (bytes):  avg  3892.3 min    1448 max    4096 sdev   705.6
```

```
  write times (msec):   avg   0.007 min   0.003 max   0.022 sdev   0.006

FILE: /usr/lib/nls/msg/en_US/ksh.cat  volume: /dev/hd2 (/usr)  inode: 16872
opens:                  2
total bytes xfrd:       16384
reads:                  4       (0 errs)
  read sizes (bytes):   avg  4096.0 min    4096 max    4096 sdev     0.0
  read times (msec):    avg   0.042 min   0.015 max   0.070 sdev   0.025
lseeks:                 10

FILE: /usr/lib/nls/msg/en_US/cmdtrace.cat  volume: /dev/hd2 (/usr)  inode: 16739
opens:                  1
total bytes xfrd:       8192
reads:                  2       (0 errs)
  read sizes (bytes):   avg  4096.0 min    4096 max    4096 sdev     0.0
  read times (msec):    avg   0.062 min   0.049 max   0.075 sdev   0.013
lseeks:                 8

------------------------------------------------------------------------
Detailed VM Segment Stats   (4096 byte pages)
------------------------------------------------------------------------

SEGMENT: 5e93  segtype: ???
segment flags:
reads:                  13      (0 errs)
  read times (msec):    avg   1.979 min   0.957 max   5.970 sdev   1.310
  read sequences:       1
  read seq. lengths:    avg    13.0 min      13 max      13 sdev     0.0

SEGMENT: 22ed  segtype: ???
segment flags:          inode
reads:                  2       (0 errs)
  read times (msec):    avg   8.102 min   7.786 max   8.418 sdev   0.316
  read sequences:       2
  read seq. lengths:    avg     1.0 min       1 max       1 sdev     0.0

SEGMENT: 5c77  segtype: persistent
segment flags:          pers defer
reads:                  1       (0 errs)
  read times (msec):    avg  13.810 min  13.810 max  13.810 sdev   0.000
  read sequences:       1
  read seq. lengths:    avg     1.0 min       1 max       1 sdev     0.0

------------------------------------------------------------------------
Detailed Logical Volume Stats   (512 byte blocks)
------------------------------------------------------------------------

VOLUME: /dev/hd4  description: /
reads:                  5       (0 errs)
  read sizes (blks):    avg    22.4 min       8 max      40 sdev    12.8
  read times (msec):    avg   4.847 min   0.938 max  13.792 sdev   4.819
  read sequences:       3
  read seq. lengths:    avg    37.3 min       8 max      64 sdev    22.9
seeks:                  3       (60.0%)
  seek dist (blks):     init   6344,
```

```
                        avg    40.0 min       8 max      72 sdev    32.0
time to next req(msec): avg  70.473 min   0.224 max 331.020 sdev 130.364
throughput:             151.9 KB/sec
utilization:            0.06

VOLUME: /dev/hd2  description: /usr
reads:                  2       (0 errs)
  read sizes (blks):    avg     8.0 min       8 max       8 sdev     0.0
  read times (msec):    avg   8.078 min   7.769 max   8.387 sdev   0.309
  read sequences:       2
  read seq. lengths:    avg     8.0 min       8 max       8 sdev     0.0
seeks:                  2       (100.0%)
  seek dist (blks):     init 608672,
                        avg    16.0 min      16 max      16 sdev     0.0
time to next req(msec): avg 162.160 min   8.497 max 315.823 sdev 153.663
throughput:             21.7 KB/sec
utilization:            0.04

--------------------------------------------------------------------------
Detailed Physical Volume Stats   (512 byte blocks)
--------------------------------------------------------------------------

VOLUME: /dev/hdisk0  description: N/A
reads:                  7       (0 errs)
  read sizes (blks):    avg    18.3 min       8 max      40 sdev    12.6
  read times (msec):    avg   5.723 min   0.905 max  20.448 sdev   6.567
  read sequences:       5
  read seq. lengths:    avg    25.6 min       8 max      64 sdev    22.9
seeks:                  5       (71.4%)
  seek dist (blks):     init 4233888,
                        avg 171086.0 min       8 max  684248 sdev 296274.2
  seek dist (%tot blks):init 48.03665,
                        avg 1.94110 min 0.00009 max 7.76331 sdev 3.36145
time to next req(msec): avg  50.340 min   0.226 max 315.865 sdev 108.483
throughput:             173.6 KB/sec
utilization:            0.10
```

การใช้คำสั่ง **filemon** ในระบบด้วยเวิร์กโหลดที่เกิดขึ้นจริงจะส่งผลทำให้รายงานมีขนาดใหญ่มาก และอาจต้องการพื้นที่บัฟเฟอร์การติดตามเพิ่มมากขึ้น พื้นที่และการใช้เวลา CPU สำหรับคำสั่ง **filemon** สามารถลดระดับผลการทำงานของระบบเพื่อขยายบางส่วน ใช้ระบบที่ไม่ใช่สภาวะแวดล้อมจริง เพื่อทดสอบคำสั่ง **filemon** ก่อนที่จะเริ่มต้นใช้ในสภาวะแวดล้อมจริง และใช้การประมวลผลแบบออฟไลน์ และสำหรับระบบที่มี CPU จำนวนมาก ให้ใช้แฟล็ก **-C all** พร้อมกับคำสั่ง **trace**

หมายเหตุ: แม้ว่าคำสั่ง **filemon** จะรายงานค่าเฉลี่ย ค่าต่ำสุด ค่าสูงสุด และค่าความเบี่ยงเบนมาตรฐานในส่วนของข้อมูลสถิติโดยละเอียด ผลลัพธ์ไม่ควรจะนำมาใช้เพื่อพัฒนาช่วงความเชื่อมั่นหรือการอนุมานเชิงสถิติตามรูปแบบอื่นๆ โดยทั่วไป การกระจายจุดข้อมูลจะไม่ใช่การสุ่มหรือเป็นสัดส่วนกัน

รายงานโกลบอลของคำสั่ง filemon:

รายงานโกลบอลแสดงไฟล์แอ็คทีฟ เซ็กเมนต์โลจิคัลวอลุ่ม และฟิสิคัลวอลุ่มในระหว่างที่วัดค่า

รายงานจะแสดงอยู่ที่จุดเริ่มต้นของรายงาน **filemon** ตามค่าดีฟอลต์ รายงานโลจิคัลไฟล์และหน่วยความจำเสมือนจะถูกจำกัดไว้ที่ 20 ไฟล์และเซ็กเมนต์ที่แอ็คทีฟตามลำดับ ซึ่งจะวัดค่าด้วย จำนวนทั้งหมดของข้อมูลที่ถ่ายโอน ถ้าระบุแฟล็ก **-v** ไว้ กิจกรรมสำหรับไฟล์และเซ็กเมนต์ทั้งหมดจะถูกรายงาน ข้อมูลทั้งหมดในรายงานจะถูกแสดงจากด้านบนจนถึงด้านล่าง ตามที่ได้แอ็คทีฟมากที่สุดจนถึงแอ็คทีฟน้อยที่สุด

แอ็คทีฟไฟล์ส่วนใหญ่:

คำสั่ง **filemon** สามารถนำมาใช้เพื่อสร้างรายงาน ที่แสดงแอ็คทีฟไฟล์ส่วนใหญ่บนเลเยอร์ของการใช้ประโยชน์จากระบบไฟล์ที่หลากหลาย ซึ่งประกอบด้วยระบบโลจิคัลไฟล์ เซ็กเมนต์หน่วยความจำเสมือน LVM และเลเยอร์ฟิสิคัลดิสก์ ส่วนนี้อธิบายถึงส่วนหัวคอลัมน์ที่แสดงอยู่บน รายงาน

#MBs จำนวนทั้งหมดของ MB ที่ส่งผ่านช่วงที่วัดได้สำหรับไฟล์นี้ แถวจะถูกเรียงลำดับตามฟิลด์นี้จากมากไปหาน้อย

#opns จำนวนของการเปิดไฟล์ในระหว่างระยะเวลาการวัด

#rds จำนวนของการอ่านที่เรียกไฟล์

#wrs จำนวนของการเขียนที่เรียกไฟล์

ไฟล์ ชื่อไฟล์ (ชื่อพาธเต็มที่อยู่ในรายงานโดยละเอียด)

วอลุ่ม:inode

โลจิคัลวอลุ่มที่ไฟล์ตั้งอยู่ และจำนวน i-node ของไฟล์ในระบบไฟล์ที่เชื่อมโยง ฟิลด์นี้สามารถนำมาใช้เพื่อเชื่อมโยงไฟล์กับเซ็กเมนต์ ที่คงอยู่และสอดคล้องกันซึ่งแสดงอยู่ในรายงานเซ็กเมนต์ VM โดยละเอียด ฟิลด์นี้อาจมีค่าว่างสำหรับไฟล์ชั่วคราวที่สร้าง และลบในระหว่างการประมวลผล

แอ็คทีฟไฟล์ส่วนใหญ่คือ `smit.log` บนโลจิคัลวอลุ่ม hd4 และไฟล์ **null** แอ็พพลิเคชันจะใช้ฐานข้อมูล terminfo สำหรับการจัดการกับหน้าจอ ดังนั้น `ksh.cat` และ `cmdtrace.cat` จึงไม่ว่างด้วย ในทุกครั้งที่ shell ต้องการติดประกาศข้อความบนหน้าจอ shell จะใช้แคตาล็อกสำหรับแหล่งที่มาของข้อมูล

หากต้องการระบุไฟล์ที่ไม่รู้จัก คุณสามารถแปลชื่อโลจิคัลวอลุ่ม `/dev/hd1` ให้เป็นจุดประกอบของระบบไฟล์ `/home` และใช้คำสั่ง **find** หรือ **ncheck** :

```
# find / -inum 858 -print
/smit.log
```

หรือ

```
# ncheck -i 858 /
/:
858     /smit.log
```

เซกเมนต์ที่ใช้งานบ่อยที่สุด:

สามารถใช้คำสั่ง **filemon** เพื่อสร้าง รายงานที่แสดงรายการเซกเมนต์ที่ใช้งานบ่อยที่สุดบนชั้นต่างๆ ของ การใช้ระบบไฟล์ รวมถึงระบบไฟล์โลจิคัล เซกเมนต์หน่วยความจำ เสมือน, LVM, และชั้นฟิสิคัลดิสก์ ส่วนนี้อธิบายส่วนหัวคอลัมน์ ที่แสดงบนรายงาน

#MBs จำนวน MBs ทั้งหมดที่โอนย้ายในช่วงเวลาที่ประเมินสำหรับเซกเมนต์นี้ แถวมีการเรียงลำดับตามฟิลด์นี้ในลำดับจากมากไปน้อย

#rpgs จำนวนหน้า 4-KB ที่อ่านเข้าในเซกเมนต์จากดิสก์

#wpgs จำนวนหน้า 4-KB ที่บันทึกจากเซกเมนต์ลงในดิสก์ (page out)

#segid VMM ID ของเซกเมนต์หน่วยความจำ

segtype ชนิดของเซกเมนต์: เซกเมนต์ที่กำลังทำงาน, เซกเมนต์ถาวร (โลคัลไฟล์), ไคลเอ็นต์ เซกเมนต์ (รีโมตไฟล์), เซกเมนต์ตารางหน้า, เซกเมนต์ระบบ, หรือเซกเมนต์ถาวร พิเศษที่มีข้อมูลระบบไฟล์ (บันทึก, ไดเร็กทอรีราก, .inode, .inodemap, .inodex, .inodexmap, .indirect, .diskmap)

วอลุ่ม:inode

สำหรับเซกเมนต์ถาวร ชื่อของโลจิคัลวอลุ่มซึ่งมีไฟล์ที่เชื่อมโยง และหมายเลข i-node ของไฟล์ ฟิลด์นี้สามารถใช้เพื่อเชื่อมโยงเซกเมนต์ถาวร กับไฟล์ที่สอดคล้องกันของเซกเมนต์นั้น ดังที่แสดงในรายงานสถิติไฟล์โดยละเอียด ฟิลด์นี้ว่างเปล่าสำหรับเซกเมนต์ที่ไม่ถาวร

ถ้าคำสั่งยังคงใช้งานอยู่ สามารถใช้เครื่องมือการวิเคราะห์หน่วยความจำเสมือน **svmon** เพื่อแสดงข้อมูลเพิ่มเติมเกี่ยวกับเซกเมนต์ เมื่อกำหนด ID เซกเมนต์ของเซกเมนต์ นั้น (segid) ดังนี้: **svmon -D** *segid* โปรดดู คำสั่ง svmon สำหรับคำอธิบาย โดยละเอียด

ในตัวอย่างของเรา segtype ??? หมายความว่าระบบ ไม่สามารถระบุชนิดเซกเมนต์ได้ และคุณต้องใช้คำสั่ง **svmon** เพื่อรับข้อมูลเพิ่มเติม

โลจิคัลวอลุ่มที่ใช้งานบ่อยที่สุด:

สามารถใช้คำสั่ง **filemon** เพื่อสร้าง รายงานที่แสดงรายการโลจิคัลวอลุ่มที่ใช้งานบ่อยที่สุดบนชั้นต่างๆ ของ การใช้ระบบไฟล์ รวมถึงระบบไฟล์โลจิคัล เซกเมนต์หน่วยความจำ เสมือน, LVM, และชั้นฟิสิคัลดิสก์ ส่วนนี้อธิบายส่วนหัวคอลัมน์ ที่แสดงบนรายงาน

util การใช้ประโยชน์โลจิคัลวอลุ่ม

#rblk จำนวนบล็อก 512-ไบต์ที่อ่านจากโลจิคัลวอลุ่ม

#wblk จำนวนบล็อก 512-ไบต์ที่บันทึกลงในโลจิคัลวอลุ่ม

KB/s อัตราการโอนย้ายข้อมูลเฉลี่ยในหน่วย KB ต่อวินาที

วอลุ่ม ชื่อโลจิคัลวอลุ่ม

คำอธิบาย

อาจเป็นจุดติดตั้งระบบไฟล์หรือชนิดของโลจิคัลวอลุ่ม (การเพจ, jfslog, บูต หรือ sysdump) อย่างใดอย่างหนึ่ง ตัวอย่างเช่น โลจิคัลวอลุ่ม /dev/hd2 คือ /usr; /dev/hd6 คือการเพจ และ /dev/hd8 คือ jfslog และยังอาจมี คำว่า *บีบอัด* ด้วย ซึ่งหมายความว่าข้อมูลทั้งหมดถูกบีบอัดโดยอัตโนมัติ โดยใช้การบีบอัด Lempel-Zev (LZ) ก่อนที่จะบันทึกข้อมูลนั้นลงในดิสก์ และข้อมูลทั้งหมด มีการคลายการบีบอัดโดยอัตโนมัติเมื่ออ่านจากดิสก์ (โปรดดู "การบีบอัด JFS" ในหน้า 266 สำหรับรายละเอียด)

การใช้ประโยชน์มีการแสดงเป็นเปอร์เซ็นต์ โดย 0.06 หมายถึงมีการใช้งาน 6 เปอร์เซ็นต์ในระหว่างช่วงเวลาที่ประเมิน

ฟิสิคัลวอลุ่มที่ใช้งานบ่อยที่สุด:

สามารถใช้คำสั่ง **filemon** เพื่อสร้าง รายงานที่แสดงรายการฟิสิคัลวอลุ่มที่ใช้งานบ่อยที่สุดบนชั้นต่างๆ ของ การใช้ระบบไฟล์ รวมถึงระบบไฟล์โลจิคัล เซกเมนต์หน่วยความจำ เสมือน, LVM, และชั้นฟิสิคัลดิสก์ ส่วนนี้อธิบายส่วนหัวคอลัมน์ ที่แสดงบนรายงาน

util การใช้ประโยชน์ฟิสิคัลวอลุ่ม

หมายเหตุ: การร้องขอ I/O โลจิคัลวอลุ่มเริ่มต้น ก่อนหน้าและสิ้นสุดหลังจากการร้องขอ I/O ฟิสิคัลวอลุ่ม ด้วยเหตุนี้ การใช้โลจิคัลวอลุ่มทั้งหมด จึงจะปรากฏขึ้นสูงกว่าการใช้ฟิสิคัลวอลุ่มทั้งหมด

#rblk จำนวนบล็อก 512-ไบต์ที่อ่านจากฟิสิคัลวอลุ่ม

#wblk จำนวนบล็อก 512-ไบต์ที่บันทึกลงในฟิสิคัลวอลุ่ม

KB/s อัตราการโอนย้ายข้อมูลเฉลี่ยในหน่วย KB ต่อวินาที

วอลุ่ม ชื่อฟิสิคัลวอลุ่ม

คำอธิบาย
คำอธิบายแบบง่ายของชนิดฟิสิคัลวอลุ่ม ตัวอย่างเช่น SCSI Multimedia ซีดีรอมไดร์ฟ หรือ 16 บิต SCSI ดิสก์ไดร์ฟ

การใช้ประโยชน์มีการแสดงเป็นเปอร์เซ็นต์โดย 0.10 หมายถึงมีการใช้งาน 10 เปอร์เซ็นต์ในระหว่างช่วงเวลาที่ประเมิน

ไฟล์ที่ใช้งานบ่อยที่สุด เรียงลำดับตามกระบวนการ:

สามารถใช้คำสั่ง **filemon** เพื่อสร้าง รายงานที่แสดงรายการไฟล์ที่ใช้งานบ่อยที่สุดบนชั้นต่างๆ ของ ระบบไฟล์ที่ใช้อยู่ รวมถึงระบบไฟล์โลจิคัล เซกเมนต์หน่วยความจำ เสมือน, LVM, และชั้นฟิสิคัลดิสก์ โดยเรียงลำดับตามกระบวนการ

#MBS จำนวนเมกะไบต์ทั้งหมดที่โอนย้ายเข้าและออกจากไฟล์ แถวมีการเรียงลำดับตามฟิลด์นี้ในลำดับจากมากไปน้อย

#opns จำนวนครั้งที่เปิดไฟล์ในระหว่างรอบระยะเวลา การประเมิน

#rds จำนวนของการเรียกระบบการอ่านที่ดำเนินการไปยังไฟล์

#wrs จำนวนของการเรียกระบบการบันทึกที่ดำเนินการไปยังไฟล์

ไฟล์ ชื่อของไฟล์ ชื่อพาธแบบเต็มอยู่ในรายงานรายละเอียด

PID ID ของกระบวนการที่เปิดไฟล์

กระบวนการ
ชื่อของกระบวนการที่เปิดไฟล์

TID ID ของเธรดที่เปิดไฟล์

ไฟล์ที่ใช้งานบ่อยที่สุด เรียงลำดับตามเธรด:

สามารถใช้คำสั่ง **filemon** เพื่อสร้าง รายงานที่แสดงรายการไฟล์ที่ใช้งานบ่อยที่สุดบนชั้นต่างๆ ของ ระบบไฟล์ที่ใช้อยู่ รวมถึงระบบไฟล์โลจิคัล เซกเมนต์หน่วยความจำ เสมือน, LVM, และชั้นฟิสิคัลดิสก์ โดยเรียงลำดับตามเธรด

#MBS จำนวนเมกะไบต์ทั้งหมดที่โอนย้ายเข้าและออกจากไฟล์ แถวมีการเรียงลำดับตามฟิลด์นี้ในลำดับจากมากไปน้อย

#opns จำนวนครั้งที่เปิดไฟล์ในระหว่างรอบระยะเวลา การประเมิน

#rds จำนวนของการเรียกระบบการอ่านที่ดำเนินการไปยังไฟล์

#wrs จำนวนของการเรียกระบบการบันทึกที่ดำเนินการไปยังไฟล์

ไฟล์ ชื่อของไฟล์ชื่อพาธแบบเต็มอยู่ในรายงานรายละเอียด

PID ID ของกระบวนการที่เปิดไฟล์

กระบวนการ
ชื่อของกระบวนการที่เปิดไฟล์

TID ID ของเธรดที่เปิดไฟล์

รายงานที่แสดงรายละเอียดของคำสั่ง filemon:

รายงานที่แสดงรายละเอียดจะกำหนดข้อมูลเพิ่มเติมสำหรับ รายงานโกลบอล

มีเพียงรายการเดียวสำหรับไฟล์ที่รายงาน เซ็กเมนต์ หรือวอลุ่ม ที่อยู่ในรายงานที่แสดงรายละเอียด ฟิลด์ในแต่ละรายการจะถูกกล่าวถึงด้านล่าง สำหรับรายงานที่แสดงรายละเอียดทั้งหมดสี่ฉบับ ฟิลด์บางฟิลด์จะรายงานค่าเดี่ยว ข้อมูลสถิติรายงานที่แสดงคุณสมบัติการแจกแจงค่าต่างๆ ตัวอย่างเช่น ข้อมูลสถิติเกี่ยวกับเวลาตอบสนองจะเก็บสำหรับคำร้องขออ่านหรือเขียนทั้งหมด ซึ่งถูกมอนิเตอร์ เวลาตอบสนองโดยเฉลี่ย ต่ำสุด และสูงสุดจะถูกรายงาน และค่าความเบี่ยงเบนมาตรฐานของเวลาการตอบสนอง ค่าความเบี่ยงเบนมาตรฐานจะถูกใช้ เพื่อแสดงจำนวนของเวลาตอบสนองแต่ละครั้งที่เบี่ยงเบนจากค่าเฉลี่ย สองในสามของเวลาตอบสนองที่ทำเป็นตัวอย่างจะอยู่ระหว่าง ค่าเฉลี่ยลบค่าความเบี่ยงเบนมาตรฐาน (`avg - sdev`) กับค่าเฉลี่ยบวกกับค่าความเบี่ยงเบนมาตรฐาน (`avg + sdev`) ถ้าการแจกแจงของเวลาตอบสนอง ถูกกระจายเหนือช่วงขนาดใหญ่ ค่าความเบี่ยงเบนมาตรฐานจะมีขนาดใหญ่กว่า เมื่อเปรียบเทียบกับเวลาตอบสนองโดยเฉลี่ย

ข้อมูลสถิติไฟล์โดยละเอียด:

ข้อมูลสถิติไฟล์โดยละเอียดถูกจัดเตรียมไว้สำหรับแต่ละไฟล์ที่แสดงอยู่ในรายงาน *Most Active Files*

stanza ของรายงาน *ไฟล์ที่แอ็คทีฟส่วนใหญ่* สามารถนำมาใช้ เพื่อพิจารณาการเข้าถึงที่ได้ทำไว้แล้วกับไฟล์ นอกจากจำนวนไบต์ทั้งหมดที่ถ่ายโอน เปิด อ่าน เขียร และ lseeks แล้ว ผู้ใช้ยังสามารถพิจารณาถึง ขนาดของการอ่าน/เขียนและเวลา

FILE ชื่อของไฟล์ชื่อพาธเต็มที่กำหนดไว้ หากเป็นไปได้

วอลุ่ม ชื่อของโลจิคัลวอลุ่ม/ระบบไฟล์ที่มีไฟล์

inode หมายเลข I-node สำหรับไฟล์ภายในระบบไฟล์

เปิด จำนวนครั้งที่ไฟล์นั้นถูกเปิดขณะที่มอนิเตอร์

จำนวนไบต์ xfrd
จำนวนไบต์ทั้งหมดของการอ่าน/เขียนจาก/ไปยังไฟล์

อ่าน จำนวนของการเรียกการอ่านพร้อมกับไฟล์

ขนาดของการอ่าน (ไบต์)
ข้อมูลสถิติเกี่ยวกับขนาดที่ถ่ายโอน (`avg/min/max/sdev`) ในหน่วยไบต์

เวลาที่อ่าน (มิลลิวินาที)
ข้อมูลสถิติเวลาตอบสนองการอ่าน (`avg/min/max/sdev`) ในหน่วยมิลลิวินาที

เขียน จำนวนการเรียกการเขียนพร้อมกับไฟล์

ขนาดของการเขียน (ไบต์)
ข้อมูลสถิติเกี่ยวกับขนาดของการโอนย้ายการเขียน

เวลาที่เขียน (มิลลิวินาที)
ข้อมูลสถิติเกี่ยวกับเวลาตอบสนองการเขียน

lseeks จำนวนของการเรียกรูทีน **lseek( )**

ขนาดของการอ่านและขนาดของการเขียนจะแสดงแนวคิดของวิธีที่แอ็พพลิเคชันของคุณ อ่านและเขียนข้อมูลได้อย่างมีประสิทธิภาพ ใช้เพจขนาด 4 KB จำนวนมากสำหรับผลลัพธ์ที่ดีที่สุด

ข้อมูลสถิติเกี่ยวกับเซ็กเมนต์ VM โดยละเอียด:

รายงาน เซ็กเมนต์ที่แอ็คทีฟส่วนใหญ่ จะแสดงรายละเอียดแบบสแตติก สำหรับเซ็กเมนต์ VM ทั้งหมด

องค์ประกอบแต่ละตัวที่แสดงอยู่รายงาน เซ็กเมนต์แอ็คทีฟส่วนใหญ่ มี stanza ที่สอดคล้องกันที่แสดงข้อมูลโดยละเอียดเกี่ยวกับ I/O จากและไปยังหน่วยความจำ

SEGMENT
ID เซ็กเมนต์ของระบบปฏิบัติการภายใน

segtype ชนิดของเนื้อหาของเซ็กเมนต์

แฟล็กเซ็กเมนต์
แอ็ตทริบิวต์เซ็กเมนต์ที่หลากหลาย

วอลุ่ม สำหรับเซ็กเมนต์ที่ยังคงอยู่ ชื่อของโลจิคัลวอลุ่มที่มี ไฟล์ที่สอดคล้องกัน

inode สำหรับเซ็กเมนต์ที่ยังคงอยู่ หมายเลข i-node สำหรับไฟล์ที่สอดคล้องกัน

อ่าน จำนวนของเพจขนาด 4096 ไบต์ที่อ่านภายในเซ็กเมนต์ (นั่นคือ เพจขาเข้า)

เวลาที่อ่าน (มิลลิวินาที)
ข้อมูลสถิติเวลาตอบสนองการอ่าน (avg/min/max/sdev) ในหน่วยมิลลิวินาที

ลำดับของการอ่าน
จำนวนของลำดับของการอ่าน ลำดับคือสตริงของเพจที่อ่านอย่างต่อเนื่อง (เพจขาเข้า) จำนวนของลำดับการอ่านคือตัวบ่งชี้ของจำนวนของ การแอ็กเซสตามลำดับ

ความยาวของลำดับของการอ่าน
ข้อมูลสถิติที่อธิบายถึงความยาวของลำดับของการอ่านในหน่วยเพจ

เขียน จำนวนของเพจที่เขียนจากเซ็กเมนต์ลงในดิสก์ (นั่นคือ เพจขาออก)

เวลาที่เขียน (มิลลิวินาที)
ข้อมูลสถิติเกี่ยวกับเวลาตอบสนองการเขียน

ลำดับของการเขียน
จำนวนของลำดับของการเขียน ลำดับคือสตริงของเพจที่เขียนอย่างต่อเนื่อง (เพจขาออก)

ความยาวของลำดับของการเขียน
ข้อมูลสถิติที่อธิบายถึงความยาวของลำดับของการเขียนในหน่วยเพจ

ด้วยการตรวจสอบจำนวนของลำดับการอ่าน คุณสามารถพิจารณาถึงการเข้าถึงตามลำดับหรือตามการสุ่ม ตัวอย่างเช่น ถ้าจำนวนลำดับของการอ่าน เข้าใกล้จำนวนการอ่านแล้ว ไฟล์ที่เข้าถึงจะถูกสุ่มเพิ่มเติม หรืออีกนัยหนึ่ง ถ้าจำนวนของลำดับการอ่านมีขนาดเล็กกว่าจำนวนของการอ่าน และความยาวของลำดับการอ่านมีค่าสูง ไฟล์ที่เข้าถึงจะมีลำดับที่มากกว่า ตรรกะเดียวกันนี้จะใช้กับลำดับการเขียน

ข้อมูลสถิติเกี่ยวกับโลจิคัลหรือฟิสิคัลวอลุ่มโดยละเอียด:

องค์ประกอบแต่ละตัวแสดงอยู่ในรายงาน โลจิคัลวอลุ่มที่แอ็คทีฟส่วนใหญ่ / ฟิสิคัลวอลุ่มที่แอ็คทีฟส่วนใหญ่ จะมี stanza ที่สอดคล้องกัน ซึ่งแสดงข้อมูลโดยละเอียดเกี่ยวกับโลจิคัลหรือฟิสิคัลวอลุ่ม

นอกจากจำนวนของการอ่านและเขียนแล้ว ผู้ใช้ยังสามารถพิจารณาเวลาในการอ่าน และเขียนรวมถึงขนาดด้วย เช่นเดียวกับระยะทางในการค้นหาเริ่มต้นและค่าเฉลี่ย สำหรับโลจิคัลหรือฟิสิคัลวอลุ่ม

VOLUME
ชื่อของวอลุ่ม

คำอธิบาย
คอธิบายของวอลุ่ม (อธิบายถึงเนื้อหา หากทำงานกับโลจิคัลวอลุ่ม อธิบายถึงชนิด หากทำงานกับฟิสิคัลวอลุ่ม)

อ่าน จำนวนของคำร้องขอการอ่านที่สร้างขึ้นพร้อมกับวอลุ่ม

ขนาดของการอ่าน (blks)
ข้อมูลเกี่ยวกับขนาดที่ถ่ายโอนการอ่าน (avg/min/max/sdev) ในหน่วย 512 ไบต์ต่อบล็อก

เวลาที่อ่าน (มิลลิวินาที)
ข้อมูลสถิติเวลาตอบสนองการอ่าน (avg/min/max/sdev) ในหน่วยมิลลิวินาที

ลำดับของการอ่าน
จำนวนของลำดับของการอ่าน ลำดับคือสตริงที่มีขนาด 512 ไบต์ต่อบล็อก ซึ่งจะถูกอ่านอย่างต่อเนื่อง และบ่งชี้จำนวนของการแอ็กเซสตามลำดับ

ความยาวของลำดับของการอ่าน
ข้อมูลสถิติที่อธิบายถึงความยาวของลำดับของการอ่านในหน่วยบล็อก

เขียน จำนวนของคำร้องขอการเขียนที่สร้างขึ้นพร้อมกับวอลุ่ม.

ขนาดของการเขียน (blks)
ข้อมูลสถิติเกี่ยวกับขนาดของการโอนย้ายการเขียน

เวลาที่เขียน (มิลลิวินาที)
ข้อมูลสถิติเกี่ยวกับเวลาตอบสนองการเขียน

ลำดับของการเขียน
จำนวนของลำดับของการเขียน ลำดับคือสตริงที่มีขนาด 512 ไบต์ต่อบล็อก ซึ่งถูกเขียนอย่างต่อเนื่อง

ความยาวของลำดับของการเขียน
ข้อมูลสถิติที่อธิบายถึงความยาวของลำดับของการเขียนในหน่วยบล็อก

ค้นหา จำนวนของการค้นหาที่มาก่อนหน้าคำร้องของให้อ่านหรือเขียน และยังแสดงเป็นเปอร์เซ็นต์ของการอ่านและการเขียนทั้งหมดที่ต้องการค้นหา

ระยะทางการค้นหา (blks)
ข้อมูลสถิติเกี่ยวกับระยะทางการค้นหาในหน่วย 512 ไบต์ต่อบล็อก นอกจากข้อมูลปกติแล้ว (avg/min/max/sdev) ระยะทางของการดำเนินการค้นหาเริ่มต้น (สมมุติว่า บล็อก 0 คือตำแหน่งเริ่มต้น) จะถูกรายงานแยกจากกัน ระยะทางการค้นหานี้ในบางครั้งจะใหญ่มาก ซึ่งจะถูกรายงานแยกจากกันเพื่อหลีกเลี่ยงการบิดเบือนของข้อมูลสถิติอื่นๆ

ระยะทางการค้นหา (cyls)
(ฟิสิคัลวอลุ่มเท่านั้น) ข้อมูลสถิติเกี่ยวกับระยะทางการค้นหาในหน่วยของดิสก์ไซลิเดอร์

เวลาที่อยู่ถัดจากคำร้องขอ
ข้อมูลสถิติ (avg/min/max/sdev) ที่อธิบายถึงระยะเวลาในหน่วยมิลลิวินาทีระหว่างคำร้องขอการอ่านหรือเขียนอย่างต่อเนื่อง ในวอลุ่ม คอลัมน์นี้บ่งชี้ถึงอัตราที่วอลุ่มได้ถูกเข้าถึง

ทรูพุต ทรูพุตของวอลุ่มทั้งหมดในหน่วย KB ต่อวินาที

การใช้ประโยชน์
ส่วนของเวลาที่วอลุ่มไม่ว่าง รายการในรายงานนี้ จะเรียงลำดับตามฟิลด์นี้จากมากไปหาน้อย

เวลาในการค้นหาที่ยาวนานสามารถเพิ่มเวลาตอบสนอง I/O และส่งผลให้ผลการทำงานของแอ็พพลิเคชันลดน้อยลง ด้วยการตรวจสอบจำนวนของลำดับการอ่าน คุณสามารถพิจารณาถึงการเข้าถึงตามลำดับหรือตามการสุ่ม ตรรกะเดียวกันนี้ใช้กับลำดับการเขียน

ข้อมูลสถิติไฟล์โดยละเอียด เรียงลำดับตามการประมวลผล:

ข้อมูลสถิติไฟล์โดยละเอียดถูกจัดเตรียมไว้สำหรับแต่ละไฟล์ ที่แสดงอยู่ในรายงาน Most Active Files ที่เรียงลำดับตามการประมวลผล

ID การประมวลผล
ID ของการประมวลผลที่เปิดไฟล์

ชื่อ ชื่อของไฟล์ที่เปิด ซึ่งรวมถึงพาธ

ID เธรด
ID ของเธรดที่เปิดไฟล์

 # ของการค้นหา
จำนวนของการค้นหา

 # ของการอ่าน
จำนวนของการดำเนินการอ่าน

ข้อผิดพลาดเกี่ยวกับการเขียน
จำนวนของข้อผิดพลาดในการอ่าน

 # ของการเขียน
จำนวนของการดำเนินการเขียน

ข้อผิดพลาดในการเขียน
จำนวนของข้อผิดพลาดในการเขียน

ไบต์ที่อ่าน
จำนวนของไบต์ที่อ่าน

min จำนวนต่ำสุดของไบต์ที่อ่านในแต่ละครั้ง

avr จำนวนเฉลี่ยของไบต์ที่อ่านในแต่ละครั้ง

max จำนวนสูงสุดของไบต์ที่อ่านในแต่ละครั้ง

ไบต์ที่เขียน
จำนวนของไบต์ที่อ่าน

min จำนวนต่ำสุดของไบต์ที่เขียนในแต่ละครั้ง

avr จำนวนเฉลี่ยของไบต์ที่เขียนในแต่ละครั้ง

max จำนวนสูงสุดของไบต์ที่เขียนในแต่ละครั้ง

เวลาที่อ่าน
เวลาที่ใช้ในการดำเนินการอ่าน

เวลาที่เขียน
เวลาที่ใช้ในการดำเนินการเขียน

ข้อมูลสถิติไฟล์โดยละเอียด เรียงลำดับตามเธรด:

ข้อมูลสถิติไฟล์โดยละเอียดถูกจัดเตรียมไว้สำหรับแต่ละไฟล์ ที่แสดงอยู่ในรายงาน `Most Active Files` ที่เรียงลำดับตามเธรด

ID เธรด
ID ของเธรดที่เปิดไฟล์

ชื่อ ชื่อของไฟล์ที่เปิด ซึ่งรวมถึงพาธ

ID การประมวลผล
ID ของการประมวลผลที่เปิดไฟล์

 # ของการค้นหา
จำนวนของการค้นหา

 # ของการอ่าน
จำนวนของการดำเนินการอ่าน

ข้อผิดพลาดเกี่ยวกับการเขียน
จำนวนของข้อผิดพลาดในการอ่าน

 # ของการเขียน
จำนวนของการดำเนินการเขียน

ข้อผิดพลาดในการเขียน
จำนวนของข้อผิดพลาดในการเขียน

ไบต์ที่อ่าน
จำนวนของไบต์ที่อ่าน

min จำนวนต่ำสุดของไบต์ที่อ่านในแต่ละครั้ง

avr จำนวนเฉลี่ยของไบต์ที่อ่านในแต่ละครั้ง

max จำนวนสูงสุดของไบต์ที่อ่านในแต่ละครั้ง

ไบต์ที่เขียน
จำนวนของไบต์ที่อ่าน

min จำนวนต่ำสุดของไบต์ที่เขียนในแต่ละครั้ง

avr จำนวนเฉลี่ยของไบต์ที่เขียนในแต่ละครั้ง

max จำนวนสูงสุดของไบต์ที่เขียนในแต่ละครั้ง

เวลาที่อ่าน
เวลาที่ใช้ในการดำเนินการอ่าน

เวลาที่เขียน
เวลาที่ใช้ในการดำเนินการเขียน

คำแนะนำสำหรับการใช้คำสั่ง filemon:

มีหลายแนวทางสำหรับการใช้คำสั่ง **filemon**

- ไฟล์ /etc/inittab จะแอ็คทีฟอยู่เสมอ Daemons ระบุใน /etc/inittab จะถูกตรวจสอบเป็นประจำ เพื่อพิจารณาถึง daemon ที่ต้องการสร้างขึ้นใหม่
- ไฟล์ /etc/passwd จะแอ็คทีฟอยู่เสมอ เนื่องจากสิทธิการเข้าถึงไฟล์และไดเร็กทอรีจะถูกตรวจสอบ
- เวลาในการค้นหาที่ยาวนานจะเพิ่มเวลาตอบสนอง I/O และลดระดับผลการทำงาน
- ถ้าการอ่านและการเขียนส่วนใหญ่ต้องการค้นหา คุณต้องแฟรกเมนต์ไฟล์และระบบไฟล์ที่แอ็คทีฟบนฟิสิคัลดิสก์เดียวกัน อย่างไรก็ตาม สำหรับการประมวลผลด้วยรายการแบบออนไลน์ (OLTP) หรือระบบฐานข้อมูลที่ลักษณะการทำงานอาจเป็นปกติ
- ถ้าจำนวนของการอ่านและเขียนเข้าใกล้จำนวนของลำดับ การเข้าถึงฟิสิคัลดิสก์จะเป็นการสุ่มมากกว่าลำดับ ลำดับคือสตริงของเพจ ที่อ่าน (เพจขาเข้า) หรือเขียน (เพจขาออก) อย่างต่อเนื่อง seq. lengths คือความยาวในหน่วยเพจของลำดับ การเข้าถึงไฟล์สุ่ม ยังสามารถเกี่ยวข้องกับการค้นหาต่างๆ ในกรณีนี้ คุณไม่สามารถแยกออกจากเอาต์พุต **filemon** ได้ ถ้าการเข้าถึงไฟล์คือการสุ่มหรือถ้าไฟล์ถูกแฟรกเมนต์ ใช้คำสั่ง **fileplace** เพื่อตรวจสอบเพิ่มเติม
- รีโมตไฟล์จะถูกแสดงอยู่ในคอลัมน์ volume:inode พร้อมกับชื่อระบบรีโมต

เนื่องจากคำสั่ง **filemon** สามารถใช้กำลัง CPU บางส่วนได้ ให้ใช้เครื่องมือด้วยการตัดสินใจ และวิเคราะห์ผลการทำงานของระบบ ขณะที่พิจารณาถึงข้อควรพิจารณาการใช้ที่เกี่ยวข้องกับการรันเครื่องมือ การทดสอบจะแสดงถึงสภาวะแวดล้อมที่เต็มไปด้วย CPU:

- ด้วย I/O เพียงเล็กน้อย คำสั่ง **filemon** ทำให้การคอมไพล์ขนาดใหญ่ช้าลงประมาณหนึ่งเปอร์เซ็นต์
- ด้วยอัตราเอาต์พุตดิสก์ที่มีค่าสูง คำสั่ง **filemon** จะเขียนโปรแกรมช้าลงประมาณห้าเปอร์เซ็นต์

บทสรุปสำหรับการมอนิเตอร์ดิสก์ I/O

โดยทั่วไป %iowait สูงบ่งชี้ว่าระบบมีปัญหาเกี่ยวกับ แอ็พพลิเคชัน การขาดแคลนหน่วยความจำ หรือการตั้งค่าคอนฟิกระบบย่อย I/O ที่ไม่มีประสิทธิภาพ ตัวอย่างเช่น ปัญหาแอ็พพลิเคชันอาจเกิดขึ้นเนื่องจากการร้องขอ I/O จำนวนมาก แต่ไม่ได้ดำเนินการกับข้อมูลมากนัก การทำความเข้าใจกับปัญหาคอขวด I/O และการปรับปรุงประสิทธิผลของระบบย่อย I/O คือหลักสำคัญในการแก้ไข ปัญหาคอขวดนี้

ความไวของดิสก์อาจมาในหลายรูปแบบ โดยมีความละเอียดแตกต่างกัน โซลูชันปกติบางรายการอาจรวมถึง:

- การจำกัดจำนวนของโลจิคัลวอลุ่มที่ใช้งานอยู่และระบบไฟล์ที่วางบน ฟิสิคัลดิสก์เฉพาะ แนวคิดนี้คือการปรับสมดุลของไฟล์ I/O ให้เท่าเทียมกันบน ฟิสิคัลดิสก์ไดร์ฟทั้งหมด
- การกระจายโลจิคัลวอลุ่มบนหลายฟิสิคัลดิสก์ มีประโยชน์เป็นพิเศษ เมื่อเข้าถึงไฟล์ที่แตกต่างกันจำนวนมาก
- การสร้างบันทึก Journaled File Systems (JFS) หลายรายการสำหรับกลุ่ม วอลุ่มหนึ่ง และการกำหนดบันทึกนั้นให้กับระบบไฟล์เฉพาะ (ถ้าเป็นไปได้ บน อุปกรณ์แคชการบันทึกด่วน) มีประโยชน์สำหรับแอ็พพลิเคชันที่สร้าง ลบ หรือแก้ไขไฟล์จำนวนมาก โดยเฉพาะไฟล์ชั่วคราว
- ถ้าเอาต์พุต **iostat** บ่งชี้ว่ากิจกรรมเวิร์กโหลด I/O ของคุณไม่ได้แจกจ่ายอย่างเท่าเทียมกันระหว่างดิสก์ไดร์ฟระบบต่างๆ และการใช้ ประโยชน์ของดิสก์ไดร์ฟหนึ่งตัวขึ้นไปเป็น 70–80 เปอร์เซ็นต์หรือมากกว่าอยู่บ่อยๆ ให้ลองพิจารณาการจัดระเบียบระบบไฟล์ใหม่ เช่น การแบ็คอัพและการเรียกคืนระบบไฟล์เพื่อลดการแบ่งเฟรกเมนต์ การแบ่งเฟรกเมนต์ทำให้ไดร์ฟค้นหามากเกินไป และอาจเป็นส่วนประกอบสำคัญ ของเวลาการตอบกลับโดยรวม
- ถ้างานพื้นหลังขนาดใหญ่ที่เน้น I/O มีส่วนเกี่ยวข้องกับเวลาตอบกลับเชิงโต้ตอบ คุณอาจต้องการเรียกใช้ I/O pacing
- ถ้าปรากฏว่ามีการอ่านไฟล์จำนวนเล็กน้อยซ้ำแล้วซ้ำเล่า ให้พิจารณาว่า หน่วยความจำจริงเพิ่มเติมจะช่วยให้สามารถบัฟเฟอร์ไฟล์เหล่านี้ ได้อย่างมีประสิทธิภาพมากขึ้นหรือไม่
- ถ้ารูปแบบการเข้าถึงของเวิร์กโหลดส่วนใหญ่เป็นแบบสุ่ม คุณอาจพิจารณา การเพิ่มดิสก์และการแจกจ่ายไฟล์ที่เข้าถึงแบบสุ่มบนไดร์ฟเพิ่มเติม
- ถ้ารูปแบบการเข้าถึงของเวิร์กโหลดส่วนใหญ่เป็นแบบตามลำดับและเกี่ยวข้องกับ หลายดิสก์ไดร์ฟ คุณอาจพิจารณาการเพิ่มดิสก์อะแด็ปเตอร์ตั้งแต่หนึ่งตัวขึ้นไป นอกจากนี้ ยังควรพิจารณาการสร้าง striped โลจิคัลวอลุ่มเพื่อ สนับสนุนไฟล์ตามลำดับขนาดใหญ่ที่มีความสำคัญต่อประสิทธิภาพ
- การใช้อุปกรณ์แคชการบันทึกด่วน
- การใช้ asynchronous I/O

การมอนิเตอร์ผลการทำงาน LVM ด้วยคำสั่ง lvmstat

คุณสามารถใช้คำสั่ง **lvmstat** เพื่อตรวจพบ พื้นที่หรือพาร์ติชันของโลจิคัลวอลุ่มถูกประเมินผลด้วยความถี่เพิ่มขึ้นจากเดิม

เพื่อแสดงข้อมูลสถิติของพื้นที่ที่เข้าถึงด้วยคำสั่ง **lvmstat** คุณต้องเปิดใช้งานข้อมูลสถิติ เพื่อรันต่อโลจิคัลวอลุ่มหรือกลุ่มวอลุ่ม

หากต้องการเปิดใช้งานข้อมูลสถิติสำหรับคำสั่ง **lvmstat** สำหรับโลจิคัลวอลุ่มเฉพาะ ให้ใช้คำสั่งต่อไปนี้:

```
# lvmstat -l lvname -e
```

หาก ต้องการปิดใช้งานข้อมูลสถิติสำหรับคำสั่ง **lvmstat** สำหรับโลจิคัลวอลุ่ม ให้ใช้คำสั่งต่อไปนี้:

```
# lvmstat -l lvname -d
```

หาก ต้องการเปิดใช้งานข้อมูลสถิติสำหรับคำสั่ง **lvmstat** สำหรับโลจิคัลวอลุ่มทั้งหมด ให้ใช้คำสั่งต่อไปนี้:

```
# lvmstat -v vgname -e
```

หาก ต้องการปิดใช้งานข้อมูลสถิติสำหรับคำสั่ง **lvmstat** สำหรับโลจิคัลวอลุ่มทั้งหมดในกลุ่มวอลุ่ม ให้ใช้คำสั่งต่อไปนี้:

```
# lvmstat -v vgname -d
```

เมื่อใช้คำสั่ง **lvmstat** ถ้าคุณไม่ได้ระบุช่วงของค่า เอาต์พุตจะแสดงข้อมูลสถิติสำหรับพาร์ติชันทุกพาร์ติชัน ในโลจิคัลวอลุ่ม เมื่อคุณระบุค่าของช่วงเวลาในหน่วยวินาที เอาต์พุตคำสั่ง **lvmstat** จะแสดงข้อมูลสถิติสำหรับพาร์ติชันโดยเฉพาะที่มีการเข้าถึงในช่วงเวลาที่ระบุ ต่อไปนี้คือตัวอย่างของคำสั่ง **lvmstat** :

```
# lvmstat -l lv00 1

Log_part  mirror#  iocnt   Kb_read   Kb_wrtn      Kbps
       1       1   65536     32768         0      0.02
       2       1   53718     26859         0      0.01

Log_part  mirror#  iocnt   Kb_read   Kb_wrtn      Kbps
       2       1    5420      2710         0  14263.16

Log_part  mirror#  iocnt   Kb_read   Kb_wrtn      Kbps
       2       1    5419      2709         0  15052.78

Log_part  mirror#  iocnt   Kb_read   Kb_wrtn      Kbps
       3       1    4449      2224         0  13903.12
       2       1     979       489         0   3059.38

Log_part  mirror#  iocnt   Kb_read   Kb_wrtn      Kbps
       3       1    5424      2712         0  12914
```

คุณสามารถใช้แฟล็ก **–c** เพื่อจำกัดจำนวนของข้อมูลสถิติที่คำสั่ง **lvmstat** แสดง แฟล็ก **–c** จะระบุจำนวนของพาร์ติชันที่มีกิจกรรม I/O ที่คุณต้องการแสดง ต่อไปนี้คือตัวอย่างของการใช้คำสั่ง **lvmstat** พร้อมกับแฟล็ก **–c** :

```
# lvmstat -l lv00 -c 5
```

คำสั่ง ข้างต้นจะแสดงข้อมูลสถิติสำหรับ 5 พาร์ติชันที่มีกิจกรรม I/O มากที่สุด

คุณไม่ต้องระบุพารามิเตอร์การวนซ้ำ คำสั่ง **lvmstat** จะยังคงสร้างเอาต์พุตจนกระทั่งคุณอินเตอร์รัปต์คำสั่ง ไม่เช่นนั้น คำสั่ง **lvmstat** จะแสดงข้อมูลสถิติสำหรับจำนวนของการวนซ้ำที่ระบุ

ในการใช้คำสั่ง **lvmstat** ถ้าคุณพบว่า มีพาร์ติชันเพียงเล็กน้อยที่ถูกใช้งานหนัก คุณอาจต้องการแยกพาร์ติชันเหล่านั้น ด้วยฮาร์ดดิสก์ที่ต่างกันโดยใช้คำสั่ง **lvmstat** คำสั่ง **lvmstat** อนุญาตให้คุณถ่ายโอนพาร์ติชัน ออกฮาร์ดดิสก์หนึ่งไปยังฮาร์ดดิสก์ตัวอื่น สำหรับรายละเอียดเกี่ยวกับการใช้คำสั่ง **lvmstat** โปรดดู คำสั่ง migratelp ใน *Commands Reference, Volume 3*

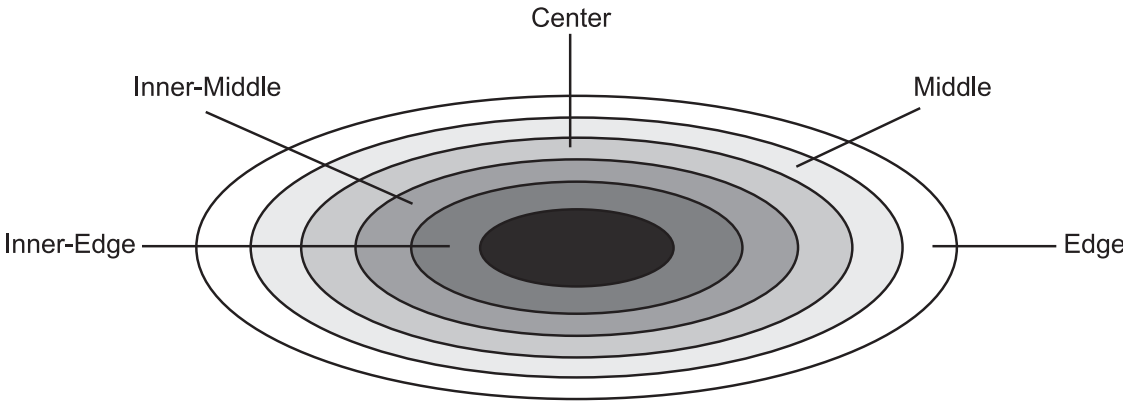
สำหรับอ็อพชันและข้อมูลเพิ่มเติมเกี่ยวกับคำสั่ง **lvmstat** โปรดดู คำสั่ง lvmstat ใน *Commands Reference, Volume 3*

แอ็ตทริบิวต์โลจิคัลวอลุ่มที่กระทบต่อผลการทำงาน

มีหลายปัจจัยที่เกี่ยวข้องกับผลการทำงานซึ่งสามารถควบคุมได้ ขณะสร้างโลจิคัลวอลุ่ม อ็อพชันเหล่านี้จะปรากฏขึ้นเป็นพร้อมต์ให้ระบุค่าบนหน้าจอ **smitty mklv**

ตำแหน่งบนฟิสิคัลวอลุ่ม

นโยบายการจัดสรรระหว่างฟิสิคัลวอลุ่ม ระบุ กลยุทธ์ที่ควรจะใช้สำหรับการเลือกฟิสิคัลพาร์ติชันบนฟิสิคัล วอลุ่ม กลยุทธ์ทั่วไปมีอยู่ห้าแบบคือขอบ ขอบด้านใน กึ่งกลาง กึ่งกลางด้านใน และศูนย์กลาง



รูปที่ 17. นโยบายการจัดสรรระหว่างฟิสิคัลวอลุ่ม. รูปภาพนี้ แสดงตำแหน่งหน่วยเก็บบนฟิสิคัลวอลุ่มหรือดิสก์ ดิสก์มีการจัดรูปแบบเป็นแทร็กจำนวนหลายร้อยแทร็ก โดยเริ่มต้นที่ขอบด้านนอก ของดิสก์และเลื่อนเข้าไปยังศูนย์กลางของดิสก์ เนื่องจากวิธีการอ่าน ดิสก์ (แทร็คหมุนอยู่ภายใต้หัวการอ่าน/การบันทึกที่เคลื่อนย้ายได้) ข้อมูลที่ถูกบันทึกใกล้กับศูนย์กลางของดิสก์จะสามารถค้นหาได้เร็วกว่าข้อมูลที่ถูกบันทึกบนขอบด้านนอก ในบางส่วน เป็นผลมาจากการดำเนินการกลไกของหัวการอ่าน/การบันทึก และส่วนของแต่ละแทร็คที่ต้องผ่านภายใต้หัว ข้อมูลมี ความหนาแน่นมากขึ้นเมื่อเลื่อนเข้าใกล้ศูนย์กลาง ส่งผลให้หัวเคลื่อนที่ ช้าลง ซึ่งส่งผลให้ผลผลิตโดยรวมเร็วขึ้น

ฟิสิคัลพาร์ติชันมีการกำหนดหมายเลขอย่างต่อเนื่อง โดยเริ่มต้นด้วยหมายเลข หนึ่ง จากขอบด้านนอกสุดไปยังขอบด้านในสุด

กลยุทธ์ขอบและขอบด้านในระบุการจัดสรรพาร์ติชันไปยังขอบของ ฟิสิคัลวอลุ่ม พาร์ติชันเหล่านี้มีเวลาการค้นหาเฉลี่ย ช้าที่สุด ซึ่งโดยทั่วไปแล้ว ส่งผลให้เวลาการตอบกลับนานขึ้นสำหรับแอ็พพลิเคชัน ใดๆ ที่ใช้พาร์ติชันเหล่านี้ ขอบบนดิสก์ที่มีการผลิตตั้งแต่ช่วงกลางทศวรรษ 1990s สามารถรองรับส่วนต่อแทร็คได้มากขึ้น ดังนั้นขอบจึงทำงานได้เร็วขึ้นสำหรับ I/O ตามลำดับ

กลยุทธ์กึ่งกลางและกึ่งกลางด้านในระบุเพื่อหลีกเลี่ยงขอบของ ฟิสิคัลวอลุ่ม และออกจากศูนย์กลางเมื่อจัดสรรพาร์ติชัน กลยุทธ์เหล่านี้จัดสรรที่ตั้งที่ดีและเหมาะสมสำหรับพาร์ติชัน พร้อมกับเวลาค้นหาเฉลี่ยที่เหมาะสม พาร์ติชันส่วนใหญ่บน ฟิสิคัลวอลุ่มมีการจัดสรรโดยใช้กลยุทธ์นี้

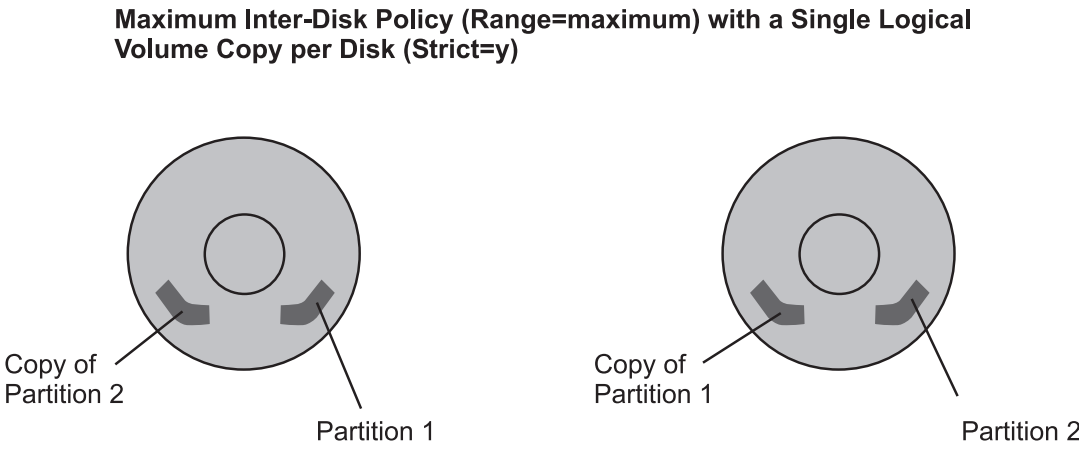
กลยุทธ์ศูนย์กลางระบุการจัดสรรพาร์ติชันไปยังส่วนศูนย์กลางของแต่ละ ฟิสิคัลวอลุ่ม พาร์ติชันเหล่านี้มีเวลาการค้นหาเฉลี่ยเร็วที่สุด ซึ่งโดยทั่วไปแล้ว ส่งผลให้เวลาการตอบกลับดีที่สุดสำหรับแอ็พพลิเคชัน ใดๆ ที่ใช้พาร์ติชันเหล่านี้ พาร์ติชันบนฟิสิคัลวอลุ่มที่จัดสรรโดยกลยุทธ์ ศูนย์กลางมีจำนวนน้อยกว่าพาร์ติชันที่จัดสรรโดยกลยุทธ์ทั่วไปอื่นๆ

โลจิคัลวอลุ่มพื้นที่ว่างการเพจเป็นตัวเลือกที่ดีสำหรับการจัดสรรที่ศูนย์กลางของ ฟิสิคัลวอลุ่ม ถ้ามีกิจกรรมการเพจมาก ในทางตรงกันข้าม ดัมพ์และบูตโลจิคัลวอลุ่มมีการใช้ไม่บ่อยกัน ดังนั้นจึงควร จัดสรรที่ตอนต้นหรือตอนท้ายของ ฟิสิคัลวอลุ่ม

กฎทั่วไปคือ I/Os ยิ่งมาก โดยตลอดหรือในช่วงเวลาการรันแอ็พพลิเคชัน ที่สำคัญ ควรจัดสรรฟิสิคัลพาร์ติชันของโลจิคัลวอลุ่มให้ยิ่งใกล้ กับศูนย์กลางของฟิสิคัลวอลุ่ม

ช่วงของฟิสิคัลวอลุ่ม

นโยบายการจัดสรรระหว่างฟิสิคัลวอลุ่ม ระบุ กลยุทธ์ที่ควรจะใช้สำหรับการเลือกอุปกรณ์ฟิสิคัลที่จะจัดสรรฟิสิคัล พาร์ติชัน ของโลจิคัลวอลุ่ม ตัวเลือกคืออ็อพชันต่ำสุดและ สูงสุด



รูปที่ 18. นโยบายการจัดสรรระหว่างฟิสิคัลวอลุ่ม. ภาพสาธิตนี้ แสดงฟิสิคัลวอลุ่ม 2 รายการ รายการหนึ่งประกอบด้วยพาร์ติชัน 1 และสำเนาของพาร์ติชัน 2 ฟิสิคัลวอลุ่มอีกรายการหนึ่งประกอบด้วยพาร์ติชัน 2 ที่มีสำเนา ของพาร์ติชัน 1 สูตรสำหรับการจัดสรรคือ Maximum Inter-Disk Policy (ช่วง=สูงสุด) ที่มีหนึ่งสำเนาโลจิคัลวอลุ่มต่อดิสก์ (Strict=y)

อ็อพชันต่ำสุดบ่งชี้จำนวนของฟิสิคัลวอลุ่มที่ใช้เพื่อ จัดสรรฟิสิคัลพาร์ติชันที่ต้องการ โดยทั่วไป นี่เป็นนโยบาย ที่ใช้เพื่อนำเสนอความน่าเชื่อถือและการมีอยู่ที่เยี่ยมที่สุด โดยไม่มีสำเนาที่โลจิคัลวอลุ่ม ตัวเลือกสองตัวที่มีอยู่ เมื่อใช้อ็อพชันต่ำสุดโดยมีและไม่มีสำเนา มีดังนี้:

- ไม่มีสำเนา: อ็อพชันต่ำสุดบ่งชี้ว่าฟิสิคัลวอลุ่มหนึ่ง ควรมีฟิสิคัลพาร์ติชันทั้งหมดของโลจิคัลวอลุ่มนี้ ถ้าโปรแกรมการจัดสรรต้องใช้ฟิสิคัลวอลุ่มตั้งแต่สองรายการขึ้นไป โปรแกรมจะใช้ จำนวนต่ำสุดที่เป็นไปได้ เพื่อยังคงให้สอดคล้องกับพารามิเตอร์ อื่น
- มีสำเนา: อ็อพชันต่ำสุดบ่งชี้ว่าควรจะใช้ฟิสิคัลวอลุ่มมาก ตามจำนวนสำเนา ถ้าโปรแกรมการจัดสรรต้องใช้ฟิสิคัลวอลุ่มตั้งแต่ สองรายการขึ้นไป โปรแกรมจะใช้จำนวนต่ำสุดของฟิสิคัลวอลุ่มที่เป็นไปได้ เพื่อรองรับฟิสิคัลพาร์ติชันทั้งหมด มีการสังเกต ข้อจำกัดที่บังคับใช้โดยพารามิเตอร์อื่นๆ เช่น strict อ็อพชันตลอดเวลา

คำนิยามเหล่านี้ใช้ได้เมื่อขยายหรือคัดลอกโลจิคัลวอลุ่ม ที่มีอยู่ มีการนับการจัดสรรที่มีอยู่เพื่อกำหนด ตัวอย่างเช่น จำนวนของฟิสิคัลวอลุ่มที่จะใช้ในกรณีของสำเนา ต่ำสุด

อ็อพชันสูงสุดบ่งชี้จำนวนของฟิสิคัลวอลุ่มที่ใช้เพื่อ จัดสรรฟิสิคัลพาร์ติชันที่ต้องการ อ็อพชันสูงสุดมีไว้เพื่อขยาย ฟิสิคัลพาร์ติชันของโลจิคัลวอลุ่มนี้บนฟิสิคัลวอลุ่มจำนวนมากที่สุดเท่าที่จะเป็นไปได้ โดยพิจารณาถึงข้อจำกัดอื่นๆ อ็อพชัน นี้เป็นอ็อพชันที่มุ่งเน้นประสิทธิภาพและควรใช้กับสำเนาเพื่อพัฒนา การมีอยู่ให้มากขึ้น ถ้าโลจิคัลวอลุ่มที่ไม่มีสำเนาขยายบนหลาย ฟิสิคัลวอลุม การสูญเสียฟิสิคัลวอลุ่มใดๆ ที่มีฟิสิคัลพาร์ติชัน จากโลจิคัลวอลุ่มนั้นอาจเพียงพอทำให้โลจิคัลวอลุ่ม ไม่สมบูรณ์ได้

จำนวนสูงสุดของฟิสิคัลวอลุ่มที่ใช้สำหรับการจัดสรร

ตั้งค่าจำนวนสูงสุดของฟิสิคัลวอลุ่มสำหรับการจัดสรรใหม่

ค่าควรอยู่ระหว่างหนึ่งจนถึงจำนวนทั้งหมดของฟิสิคัลวอลุ่ม ที่อยู่ในกลุ่มวอลุ่ม อ็อพชันนี้จะเกี่ยวข้องกับ “ช่วงของฟิสิคัลวอลุ่ม”

ความสอดคล้องกันของมิเรอร์การเขียน

LVM มั่นใจว่า ความสอดคล้องกันของข้อมูลระหว่างสำเนาของโลจิคัลวอลุ่มที่มิเรอร์ ระหว่างการประมวลผล I/O ปกติ

สำหรับทุกๆ การเขียนลงในโลจิคัลวอลุ่ม LVM จะสร้างคำร้องขอการเขียน สำหรับสำเนามิเรอร์ ปัญหาเกิดขึ้นหากระบบพังในส่วนกลางของการประมวลผลที่มิเรอร์การเขียน (ก่อนสำเนาทั้งหมดจะถูกเขียน) ถ้าการกู้คืนความสอดคล้องกันของมิเรอร์การเขียน ถูกร้องขอสำหรับโลจิคัลวอลุ่ม LVM จะเก็บข้อมูลเพิ่มเติมเพื่ออนุญาตให้กู้คืนมิเรอร์ที่สอดคล้องกันเหล่านี้ การกู้คืนความสอดคล้องกันของมิเรอร์การเขียน ควรดำเนินการสำหรับโลจิคัลวอลุ่มที่มิเรอร์ส่วนใหญ่ โลจิคัลวอลุ่ม เช่น พื้นที่เพจที่ไม่ได้ใช้ข้อมูลที่มีอยู่ เมื่อกลุ่มวอลุ่มถูก vary on ห้ามทำการปกป้อง

เร็กคอร์ด Mirror Write Consistency (MWC) ประกอบด้วยหนึ่งเซ็กเตอร์ ซึ่งระบุ โลจิคัลพาร์ติชันที่อาจไม่สอดคล้องกัน หากระบบไม่ได้ปิดระบบ อย่างถูกต้อง เมื่อกลุ่มวอลุ่มถูก vary on กลับมาออนไลน์ ข้อมูลนี้จะถูกใช้เพื่อสร้างโลจิคัลพาร์ติชันที่สอดคล้องกันอีกครั้ง

หมายเหตุ: ด้วย Mirror Write Consistency LVs เนื่องจากเซ็กเตอร์การควบคุม MWC อยู่บนขอบของดิสก์ ผลการทำงานอาจถูกปรับปรุง ถ้าโลจิคัลวอลุ่มที่ทำมิเรอร์ยังคงอยู่บนขอบ

จุดเริ่มต้นใน AIX อ็อพชันความสอดคล้องกันของมิเรอร์การเขียนที่เรียกว่า Passive Mirror Write Consistency จะพร้อมใช้งาน กลไกดีฟอลต์สำหรับการทำให้มั่นใจว่า ความสอดคล้องกับการเขียนที่ทำมิเตอร์คือ Active MWC MWC ที่แอ็คทีฟจะจัดเตรียมการกู้คืนแบบเร็วที่เวลาของการรีบูต หลังจากที่เกิดพัง อย่างไรก็ตาม ข้อดีนี้มาจากค่าใช้จ่ายของการเขียนการลดระดับผลการทำงานการเขียน โดยเฉพาะในเคสของการเขียนแบบสุ่ม การปิดใช้งาน Active MWC จะจำกัดข้อเสียของผลการทำงานการเขียนนี้ แต่ขึ้นอยู่กับการรีบูตหลังจากที่ระบบพัง คุณต้องใช้คำสั่ง **syncvg -f** เพื่อซิงโครไนต์กลุ่มวอลุ่มทั้งหมด ก่อนที่ผู้ใช้จะสามารถเข้าถึงกลุ่มวอลุ่มนั้นได้ หากต้องการบรรลุการ vary on แบบอัตโนมัติ ของกลุ่มวอลุ่มต้องถูกปิดใช้งาน

การปิดใช้งาน Passive MWC ไม่ได้จำกัดข้อเสียเกี่ยวกับผลการทำงานของการเขียนเท่านั้น ที่เชื่อมโยงกับ Active MWC แต่โลจิคัลพาร์ติชันจะถูกซิงโครไนซ์ใหม่แบบอัตโนมัติ ตามพาร์ติชันที่กำลังเข้าถึง นั่นหมายความว่า ผู้ดูแลระบบไม่ซิงโครไนซ์โลจิคัลวอลุ่มแบบแมนวน หรือปิดใช้งาน vary on แบบอัตโนมัติ ผลเสียของ Passive MWC คือการดำเนินการอ่านช้าซึ่งอาจเกิดขึ้นจนกว่าพาร์ติชันทั้งหมดจะถูกซิงโครไนซ์ใหม่

คุณสามารถเลือกอ็อพชันความสอดคล้องกันกับมิเรอร์การเขียนภายใน SMIT เมื่อสร้างหรือเปลี่ยนโลจิคัลวอลุ่ม อ็อพชันการเลือกจะมีผลบังคับใช้ เมื่อโลจิคัลวอลุ่มจะถูกมิเรอร์ (สำเนา > 1)

ข้อมูลที่เกี่ยวข้อง:

นโยบายความต้องกันในการเขียนสำเนาสำหรับโลจิคัลวอลุ่ม

จัดสรรแต่ละโลจิคัลพาร์ติชันที่คัดลอกบน PV ที่แยกจากกัน

ระบุว่า ให้ปฏิบัติตามนโยบายการจัดสรรอย่างเคร่งครัด

นโยบายการจัดสรรที่เคร่งครัดจะจัดสรรสำเนาของโลจิคัลพาร์ติชันแต่ละชุด บนฟิสิคัลวอบุ่มที่แยกจากกัน อ็อพชันนี้จะเกี่ยวข้องกับ “ช่วงของฟิสิคัลวอลุ่ม” ในหน้า 222

เปลี่ยนตำแหน่งโลจิคัลวอลุ่มในระหว่างการจัดระเบียบใหม่หรือไม่?

ระบุว่าจะอนุญาตให้เปลี่ยนตำแหน่งของโลจิคัลวอลุ่มในระหว่าง การจัดระเบียบใหม่หรือไม่

สำหรับ striped โลจิคัลวอลุ่ม ต้องตั้งค่าพารามิเตอร์การเปลี่ยนตำแหน่งเป็น no (ค่าดีฟอลต์สำหรับ striped โลจิคัลวอลุ่ม) ขึ้นอยู่กับการติดตั้งของคุณ คุณอาจต้องการเปลี่ยนตำแหน่งโลจิคัลวอลุ่ม

นโยบายการจัดตารางเวลาสำหรับการอ่านและการบันทึกสำเนาโลจิคัล พาร์ติชัน

สามารถตั้งค่านโยบายการจัดตารางเวลาที่แตกต่างอื่นสำหรับโลจิคัลวอลุ่มได้

มีการใช้นโยบายการจัดตารางเวลาหลายชนิดสำหรับโลจิคัลวอลุ่มที่มี หลายสำเนา ดังนี้:

- นโยบาย *ขนาน* ปรับสมดุลการอ่านระหว่างดิสก์ต่างๆ ในแต่ละการอ่าน ระบบจะตรวจสอบว่าสำเนาหลักยุ่งอยู่หรือไม่ ถ้าไม่ยุ่ง การอ่านจะมีการเริ่มต้น บนสำเนาหลัก ถ้าสำเนาหลักยุ่ง ระบบจะตรวจสอบสำเนา รอง ถ้าไม่ยุ่ง การอ่านจะมีการเริ่มต้นบนสำเนารอง ถ้าสำเนารองยุ่ง การอ่านจะมีการเริ่มต้นบนสำเนาที่มีจำนวน I/Os ค้างอยู่ น้อยที่สุด การบันทึกมีการเริ่มต้นขึ้นพร้อมกัน
- นโยบาย *ขนาน/ตามลำดับ* เริ่มต้นการอ่านบนสำเนาหลัก เสมอ การบันทึกมีการเริ่มต้นขึ้นพร้อมกัน
- นโยบาย *ขนาน/round robin* คล้ายกับนโยบายขนานยกเว้นว่า แทนที่จะตรวจสอบสำเนาหลักก่อนเสมอ นโยบายนี้จะเปลี่ยนไปมา ระหว่างสำเนาต่างๆ ซึ่งส่งผลให้มีการใช้ประโยชน์อย่างเท่าเทียมกันสำหรับการอ่าน แม้ว่า จะไม่มีทางมี I/O ที่ค้างอยู่มากกว่าหนึ่งรายการก็ตาม การบันทึกมีการเริ่มต้นขึ้น พร้อมกัน
- นโยบาย *ตามลำดับ* ส่งผลให้ออกใช้การอ่านทั้งหมดไปยัง สำเนาหลัก การบันทึกเกิดขึ้นตามลำดับ โดยเริ่มต้นบันทึกลงในดิสก์หลักก่อน เฉพาะเมื่อการ บันทึกในดิสก์หลักเสร็จสมบูรณ์แล้วเท่านั้นจึงจะมีการเริ่มต้นการบันทึกที่สองลงในดิสก์รอง

สำหรับข้อมูลที่มีสำเนาฟิสิคัลเพียงชุดเดียว ไดรเวอร์อุปกรณ์ของโลจิคัลวอลุ่ม จะแปลที่อยู่การร้องขอการอ่านหรือการบันทึกโลจิคัลเป็นที่อยู่ฟิสิคัล และเรียกไดรเวอร์อุปกรณ์ฟิสิคัลที่เหมาะสมเพื่อให้บริการการร้องขอ นโยบายสำเนาเดียวนี้จัดการ Bad Block Relocation สำหรับการร้องขอการบันทึก และส่งคืน ข้อผิดพลาดการอ่านทั้งหมดไปยังกระบวนการที่เรียก

นโยบายการทำมิเรอร์การจัดตารางเวลา เช่น ขนาน และขนาน/round-robin ช่วยให้ประสิทธิภาพของการตั้งค่าคอนฟิกแบบเน้นการอ่านที่มิเรอร์เท่าเทียมกับ ประสิทธิภาพของที่ไม่ได้มิเรอร์ โดยปกติ ประสิทธิภาพของการตั้งค่าคอนฟิกแบบเน้นการบันทึก ที่มิเรอร์ต่ำกว่าแบบที่ไม่ได้มิเรอร์ ยกเว้นว่ามีการใช้ดิสก์มากขึ้น

เปิดใช้งานการตรวจสอบการเขียน

ระบุว่า ให้ตรวจสอบการเขียนทั้งหมดลงในโลจิคัลวอลุ่ม ด้วยการติดตามการอ่าน

การตั้งค่านี้ให้มีค่าเป็น *On* จะกระทบต่อผลการทำงาน

Stripe size

ขนาด strip ที่คูณด้วยจำนวนดิสก์ในอาร์เรย์ เท่ากับ *ขนาด stripe* ขนาด strip อาจเป็นจำนวนยกกำลัง 2 ใดๆ ตั้งแต่ 4 KB ถึง 128 MB

เมื่อกำหนด striped โลจิคัลวอลุ่ม ต้องใช้ฟิสิคัลไดร์ฟอย่างน้อยสอง รายการ ขนาดของโลจิคัลวอลุ่มในหน่วยพาร์ติชันต้องเป็นผลคูณเลขจำนวนเต็มของจำนวนดิสก์ไดร์ฟที่ใช้ โปรดดู “การปรับ striping โลจิคัลวอลุ่ม” ในหน้า 228 สำหรับคำอธิบายโดยละเอียด

การปรับผลการทำงาน LVM ด้วยคำสั่ง lvmo

คุณสามารถใช้คำสั่ง **lvmo** เพื่อจัดการกับจำนวนของ LVM pbufs ต่อกลุ่มวอลุ่ม

พารามิเตอร์ที่สามารถปรับแต่งได้สำหรับคำสั่ง **lvmo** มีดังต่อไปนี้:

pv_pbuf_count

จำนวนของ pbufs ที่จะเพิ่มเมื่อเพิ่มฟิสิคัลวอลุ่ม ให้กับกลุ่มวอลุ่ม

max_vg_pbuf_count

จำนวนสูงสุดของ pbufs ที่สามารถจัดสรรไว้สำหรับกลุ่มวอลุ่ม สำหรับค่านี้เพื่อทำให้มีผลบังคับใช้ กลุ่มวอลุ่มต้องถูก vary off และ vary on อีกครั้ง

global_pbuf_count

จำนวนต่ำสุดของ pbufs ที่จะเพิ่มเมื่อเพิ่มฟิสิคัลวอลุ่ม ให้กับกลุ่มวอลุ่มใดๆ หากต้องการเปลี่ยนค่านี้ ให้ใช้คำสั่ง **ioo**

aio_cache_pbuf_count

จำนวน pbufs ทั้งหมดในปัจจุบันที่พร้อมใช้งานสำหรับโลจิคัลวอลุ่ม aio_cache ในกลุ่มวอลุ่ม จำนวนสูงสุดของ `aio_cache_pbuf_count` ที่ สามารถจัดสรรให้กับกลุ่มวอลุ่มมีการระบุไว้โดยพารามิเตอร์ `max_vg_pbuf_count`

ในตัวอย่างต่อไปนี้ คำสั่ง **lvmo –a** แสดงค่าปัจจุบันสำหรับพารามิเตอร์ที่สามารถปรับได้ในกลุ่มวอลุ่ม `rootvg`

```
# lvmo -a

vgname = rootvg
pv_pbuf_count = 256
total_vg_pbufs = 768
max_vg_pbuf_count = 8192
pervg_blocked_io_count = 0
global_pbuf_count = 256
global_blocked_io_count = 20
aio_cache_pbuf_count = 512
```

ถ้าคุณต้องการแสดงค่าปัจจุบันสำหรับกลุ่มวอลุ่มอื่น ให้ใช้คำสั่งต่อไปนี้:

```
lvmo -v <vg_name> -a
```

หากต้องการตั้งค่าสำหรับความสามารถในการปรับแต่งด้วยคำสั่ง **lvmo** ให้ใช้เครื่องหมายเท่ากับ ดังที่แสดงในตัวอย่างต่อไปนี้:

หมายเหตุ: ในตัวอย่าง ต่อไปนี้ ความสามารถในการปรับ *pv_pbuf_count* มีการตั้งค่า เป็น 257 ในกลุ่มวอลุ่ม `redvg`

```
# lvmo -v redvg -o pv_pbuf_count=257

vgname = redvg
pv_pbuf_count = 257
total_vg_pbufs = 257
max_vg_pbuf_count = 263168
pervg_blocked_io_count = 0
global_pbuf_count = 256
global_blocked_io_count = 20
```

หมายเหตุ: ถ้าคุณเพิ่มค่า pbuf มากขึ้น คุณจะมองเห็นการลดระดับในผลการทำงาน หรือลักษณะการทำงานของระบบที่คาดไม่ถึง

ข้อมูลที่เกี่ยวข้อง:

คำสั่ง lvmo

ข้อควรพิจารณาเกี่ยวกับฟิสิคัลวอลุ่ม

มีสองสามเรื่องที่ควรพิจารณาเกี่ยวกับฟิสิคัลวอลุ่ม

ประเด็นประสิทธิภาพที่สำคัญสำหรับดิสก์ไดร์ฟเกี่ยวข้องกับแอ็พพลิเคชัน นั่นคือ จะทำการเข้าถึงขนาดเล็กจำนวนมาก (แบบสุ่ม) หรือการเข้าถึงขนาดใหญ่ จำนวนน้อยกว่า (ตามลำดับ) สำหรับการเข้าถึงแบบสุ่ม โดยทั่วไปแล้ว ประสิทธิภาพจะดีกว่าการใช้ไดร์ฟความจุขนาดเล็กจำนวนมากกว่า ตรงกันข้ามสำหรับ การเข้าถึงตามลำดับ (การใช้ไดร์ฟที่เร็วกว่าหรือการใช้ striping ที่มีจำนวนไดร์ฟมากกว่า)

ข้อแนะนำเกี่ยวกับกลุ่มวอลุ่ม

ถ้าเป็นไปได้ เพื่อให้ได้การจัดการระบบที่ง่ายขึ้นและประสิทธิภาพที่ดีขึ้น กลุ่มวอลุ่มดีฟอลต์, rootvg, ควรประกอบด้วยฟิสิคัลวอลุ่มที่ ระบบปฏิบัติการมีการติดตั้งเป็นครั้งแรกเท่านั้น

การเก็บระบบปฏิบัติการเพียงอย่างเดียวไว้ใน rootvg เป็นการตัดสินใจที่ได้ เนื่องจากการอัพเดตระบบปฏิบัติการ การติดตั้งใหม่ และการกู้คืนสามารถดำเนินการ ได้โดยไม่เป็นอันตรายต่อข้อมูลผู้ใช้ อัพเดตและการติดตั้งใหม่สามารถทำได้เร็วขึ้นมาก เนื่องจากมีเฉพาะระบบปฏิบัติการเท่านั้นรวมอยู่ในการเปลี่ยนแปลง

ควรกำหนดกลุ่มวอลุ่มอื่นหนึ่งรายการขึ้นไปสำหรับฟิสิคัลวอลุ่มอื่น ที่จัดเก็บข้อมูลผู้ใช้ การมีข้อมูลผู้ใช้ในกลุ่มวอลุ่มอื่น ช่วยให้สามารถส่งออกข้อมูลนั้นไปยังระบบอื่นได้ง่ายขึ้น

วางระบบไฟล์ที่แอ็คทีฟสูงบนดิสก์หนึ่ง และวางล็อกของระบบไฟล์นั้น บนอีกดิสก์หนึ่ง ถ้ากิจกรรมอาจสร้างธุรกรรมล็อกจำนวนมาก อุปกรณ์ที่แคช (เช่น solid-state disks หรือดิสก์อาร์เรย์ที่มีแคชการบันทึก) ให้ ประสิทธิภาพที่ดีกว่ามากสำหรับล็อกโลจิคัลวอลุ่ม (ล็อก JFS หรือล็อกของฐานข้อมูล)

หลักการที่เกี่ยวข้อง:

“การจัดระเบียบบันทึกระบบไฟล์และโลจิคัลวอลุ่มบันทึกใหม่” ในหน้า 278

Journaled File System (JFS) และ Enhanced Journaled File System (JFS2) ใช้เทคนิคการเจอร์นัลฐานข้อมูลเพื่อรักษาโครงสร้างระบบไฟล์ ที่สอดคล้องกัน เทคนิคนี้เกี่ยวข้องกับการทำซ้ำธุรกรรมที่ทำใน metadata ระบบไฟล์ลงในบันทึกระบบไฟล์ หมุนเวียน Metadata ระบบไฟล์ ประกอบด้วย superblock, i-nodes, ตัวชี้ข้อมูลทางอ้อม และไดเร็กทอรี

ผลกระทบต่อประสิทธิภาพจากการทำมิเรอร์ rootvg

ในการทำมิเรอร์ เมื่อเกิดการบันทึกขึ้น การบันทึกต้องเกิดขึ้นในสำเนา โลจิคัลวอลุ่มทั้งหมด โดยปกติ การเขียนนี้ใช้เวลานานกว่า โลจิคัลวอลุ่มที่ไม่ได้มิรเรอร์

แม้ว่าการทำมิเรอร์เป็นสิ่งที่ทำกันทั่วไปสำหรับข้อมูลลูกค้า โดยเฉพาะในสภาพ แวดล้อมฐานข้อมูล แต่ไม่ค่อยมีการใช้กันบ่อยนักสำหรับวอลุ่มระบบ

การทำมิรเรอร์ยังอาจทำให้โอเวอร์เฮดตัวประมวลผลเพิ่มขึ้นด้วย เนื่องจากดิสก์ I/Os สองตัวใช้คำสั่งในการทำให้สมบูรณ์มากกว่าตัวเดียว สิ่งสำคัญคือการทำความเข้าใจกับ โครงร่างของ rootvg โลจิคัลวอลุ่ม เพื่อให้สามารถคาดเดาตำแหน่งที่อาจมีปัญหา เมื่อทำมิเรอร์ rootvg โลจิคัลวอลุ่มได้

โลจิคัลวอลุ่มที่พบใน rootvg ซึ่งมี ไฟล์ใน / และไฟล์ /usr/bin ที่ใช้มาก ซึ่งมีโปรแกรมปฏิบัติการจำนวนมาก ต้องเป็นข้อมูลที่อ่านโดยส่วนใหญ่ พื้นที่ว่างการเพจ ต้องมีการบันทึกเฉพาะถ้าจำนวนของหน่วยความจำฟิสิคัลในระบบไม่เพียงพอ ที่จะรองรับ

กิจกรรมระดับปัจจุบันเท่านั้น ถือเป็นเรื่องปกติที่ระบบจะ เพจเป็นครั้งคราว แต่โดยปกติแล้ว การเพจอย่างหนักอย่างต่อเนื่อง นำไปสู่เวลา การตอบกลับที่ไม่ดี โดยทั่วไป การเพิ่มหน่วยความจำฟิสิคัลสามารถช่วยแก้ไขปัญหานี้ได้

ระบบไฟล์ /tmp และ /var ทำให้เห็น กิจกรรมการบันทึกไฟล์ของแอ็พพลิเคชันหลายชนิด แอ็พพลิเคชัน เช่น คอมไพเลอร์ มักจะสร้างและบันทึกไฟล์ชั่วคราวไว้ในไดเร็กทอรี /tmp ไดเร็กทอรี /var ได้รับไฟล์ที่มีการกำหนดสำหรับเมลและคิวเครื่องพิมพ์ มีการเขียน **jfslog** ระหว่างการดำเนินการปกติเท่านั้น ในระบบไฟล์ที่เหลือ เฉพาะ ไดเร็กทอรี /home เท่านั้นที่แอ็คทีฟระหว่างการดำเนินการปกติ บ่อยครั้งที่ไดเร็กทอรีโฮมของผู้ใช้ถูกนำไปวางไว้ในระบบไฟล์อื่น ซึ่งทำให้การจัดการ rootvg ง่ายขึ้น

Rootvg สามารถมีการมิรเรอร์โดยการมิรเรอร์แต่ละโลจิคัล วอลุ่มใน rootvg โดยใช้คำสั่ง **mklvcopy** หรือโดยการ มิรเรอร์ทั้งกลุ่มวอลุ่มโดยใช้คำสั่ง **mirrorvg**

โดยค่าดีฟอลต์ คำสั่ง **mirrorvg** ใช้นโยบายการจัดตารางเวลาแบบ ขนานและปิดการตรวจสอบการบันทึกสำหรับโลจิคัลวอลุ่มทั้งหมด คำสั่งไม่ได้เปิดใช้งานความสม่ำเสมอ ของ mirror-write สำหรับพื้นที่ว่างหน้า แต่เปิดใช้งานความสม่ำเสมอของ mirror-write สำหรับโลจิคัลวอลุ่มอื่นทั้งหมด ควรวาง โลจิคัลวอลุ่มที่มีการเขียนบ่อยครั้งไว้ใกล้กับขอบ ด้านนอกของดิสก์ เพื่อลดระยะทางการค้นหาระหว่างโลจิคัลวอลุ่ม และแคชความสม่ำเสมอของ mirror-write ให้เหลือน้อยที่สุด

คำเตือนคือการทำมิรเรอร์ rootvg ไม่ส่งผลกระทบต่อประสิทธิภาพมากนัก นั่นคือ ถ้ามีการทำมิรเรอร์พื้นที่ว่างการเพจ ประสิทธิภาพที่ช้าลงอาจเกี่ยวข้องโดย ตรงกับอัตราการเพจ ดังนั้น ระบบที่มีการกำหนดคอนฟิกให้สนับสนุน อัตราการเพจสูง ซึ่งมีพื้นที่ว่างการเพจบน rootvg อาจไม่ต้องการทำการมิรเรอร์ rootvg

โดยสรุป rootvg ที่มิรเรอร์อาจมีประโยชน์ถ้าเวิร์กโหลดของคุณ ไม่ได้มีอัตราการเพจสูงอย่างต่อเนื่อง

การจัดระเบียบโลจิคัลวอลุ่มใหม่

ถ้าคุณพบว่าวอลุ่มมีการแบ่งเฟรกเมนต์มากจนต้องจัดระเบียบใหม่ คุณสามารถใช้คำสั่ง **reorgvg** (หรือ **smitty reorgvg**) เพื่อจัดระเบียบโลจิคัลวอลุ่มใหม่และทำให้เป็นไปตาม นโยบายที่ระบุ

คำสั่งนี้จะจัดระเบียบการวางตำแหน่งของฟิสิคัลพาร์ติชันใหม่ภายใน กลุ่มวอลุ่ม ตามลักษณะของโลจิคัลวอลุ่ม ถ้ามีการระบุชื่อโลจิคัลวอลุ่มพร้อมกับคำสั่ง จะมีการกำหนดระดับความสำคัญสูงสุดให้กับ โลจิคัลวอลุ่มแรกในรายการ เพื่อใช้คำสั่งนี้ กลุ่มวอลุ่ม ต้องมีการเปลี่ยนแปลงและมีพาร์ติชันที่ว่าง แฟล็กที่เปลี่ยนตำแหน่งได้ของแต่ละโลจิคัลวอลุ่ม ต้องมีการตั้งค่าเป็น yes เพื่อให้การจัดระเบียบใหม่เกิดขึ้น มิฉะนั้น โลจิคัลวอลุ่มจะถูกละเว้นไป

โดยทราบรูปแบบการใช้งานของโลจิคัลวอลุ่ม คุณสามารถทำการตัดสินใจได้ดีขึ้น เกี่ยวกับการตั้งค่านโยบายสำหรับแต่ละวอลุ่ม คำแนะนำมีดังนี้:

- จัดสรร LVs ที่ใช้บ่อยให้กับ PVs อื่น
- กระจาย LV ที่ใช้บ่อยบนหลาย PVs
- วาง LVs ที่ใช้บ่อยที่สุดในศูนย์กลางของ PVs ยกเว้นสำหรับ LVs ที่เปิด Mirror Write Consistency Check
- วาง LVs ที่ใช้น้อยที่สุดบนขอบของ PVs (ยกเว้นเมื่อมีการเข้าถึงตามลำดับ)
- ทำให้ LV ต่อเนื่องกัน
- กำหนด LV เป็นขนาดสูงสุดที่คุณจะต้องการ
- วางโลจิคัลวอลุ่มที่ใช้บ่อยไว้ใกล้กัน
- วางไฟล์ตามลำดับบนขอบ

ข้อแนะนำสำหรับประสิทธิภาพที่ดีที่สุด

เมื่อใดก็ตามที่ตั้งค่าคอนฟิกโลจิคัลวอลุ่มสำหรับประสิทธิภาพที่ดีขึ้น การมีอยู่อาจได้รับผลกระทบ เลือกว่าระหว่างประสิทธิภาพและการมีอยู่ เรื่องใดที่สำคัญมากกว่ากันในสภาพแวดล้อมของคุณ

ใช้คำแนะนำเหล่านี้เมื่อตั้งค่าคอนฟิกประสิทธิภาพสูงสุดโดยใช้ คำสั่ง SMIT:

- ถ้าระบบทำการอ่านเป็นส่วนใหญ่ การทำมิเรอร์ด้วยนโยบายการจัดตารางเวลาที่ตั้ง ค่าเป็นแบบขนานสามารถช่วยให้ประสิทธิภาพดีขึ้น เนื่องจาก I/Os การอ่านจะตรง ไปยังสำเนาที่ยุ่งน้อยที่สุด ถ้าระบบทำการบันทึก การทำมิเรอร์ จะทำให้เกิด penalty ประสิทธิภาพ เนื่องจากจะมีหลายสำเนาที่จะบันทึก และเร็กคอร์ด Mirror Write Consistency ที่จะอัพเดต คุณอาจยังต้องการ ตั้งค่านโยบายการจัดสรรเป็น Strict เพื่อให้มีสำเนาแต่ละชุดบนฟิสิคัล วอลุ่มแยกต่างหาก
- ตั้งค่านโยบายตรวจสอบการบันทึกเป็น ไม่ และถ้าจำนวนสำเนามากกว่า หนึ่ง ให้ตั้งค่า Mirror Write Consistency เป็นปิด
- โดยทั่วไป โลจิคัลวอลุ่มที่เข้าถึงบ่อยที่สุดควรอยู่ในศูนย์กลางเพื่อลดระยะการค้นหา ให้เหลือน้อยที่สุด อย่างไรก็ตาม มีข้อยกเว้นบางอย่าง:
  - ดิสก์เก็บข้อมูลมากขึ้นต่อแทร็คไว้บนขอบของดิสก์โลจิคัลวอลุ่ม ที่เข้าถึงในลักษณะตามลำดับควรมีการวางไว้บนขอบเพื่อให้ประสิทธิภาพ ดีขึ้น
  - ข้อยกเว้นอีกอย่างหนึ่งใช้สำหรับโลจิคัลวอลุ่มที่มีการเปิด Mirror Write Consistency Check (MWCC) เนื่องจากส่วน MWCC อยู่บนขอบของดิสก์ ประสิทธิภาพอาจพัฒนาขึ้นถ้าโลจิคัลวอลุ่มที่มิเรอร์อยู่บนขอบ ด้วย
- โลจิคัลวอลุ่มที่จะเข้าถึงบ่อยหรือพร้อมกันควรมีการวางไว้ ใกล้กันบนดิสก์ Locality ของการอ้างอิงมีความสำคัญ มากกว่าการวางการอ้างอิงในศูนย์กลาง
- วางโลจิคัลวอลุ่มที่มีการใช้งานปานกลางไว้ในกึ่งกลาง และวางโลจิคัลวอลุ่ม ที่ไม่ค่อยได้ใช้ไว้บนขอบ
- โดยการตั้งค่านโยบายการจัดสรรระหว่างฟิสิคัลวอลุ่มเป็นค่าสูงสุด คุณยังมั่นใจได้ว่า การอ่านและการบันทึกมีการแบ่งใช้ระหว่าง PVs

ข้อแนะนำสำหรับการมีอยู่สูงสุด

เมื่อต้องการตั้งค่าคอนฟิกระบบให้มีการมีอยู่สูงสุด (โดยใช้คำสั่ง SMIT) ให้ปฏิบัติตามคำแนะนำเหล่านี้:

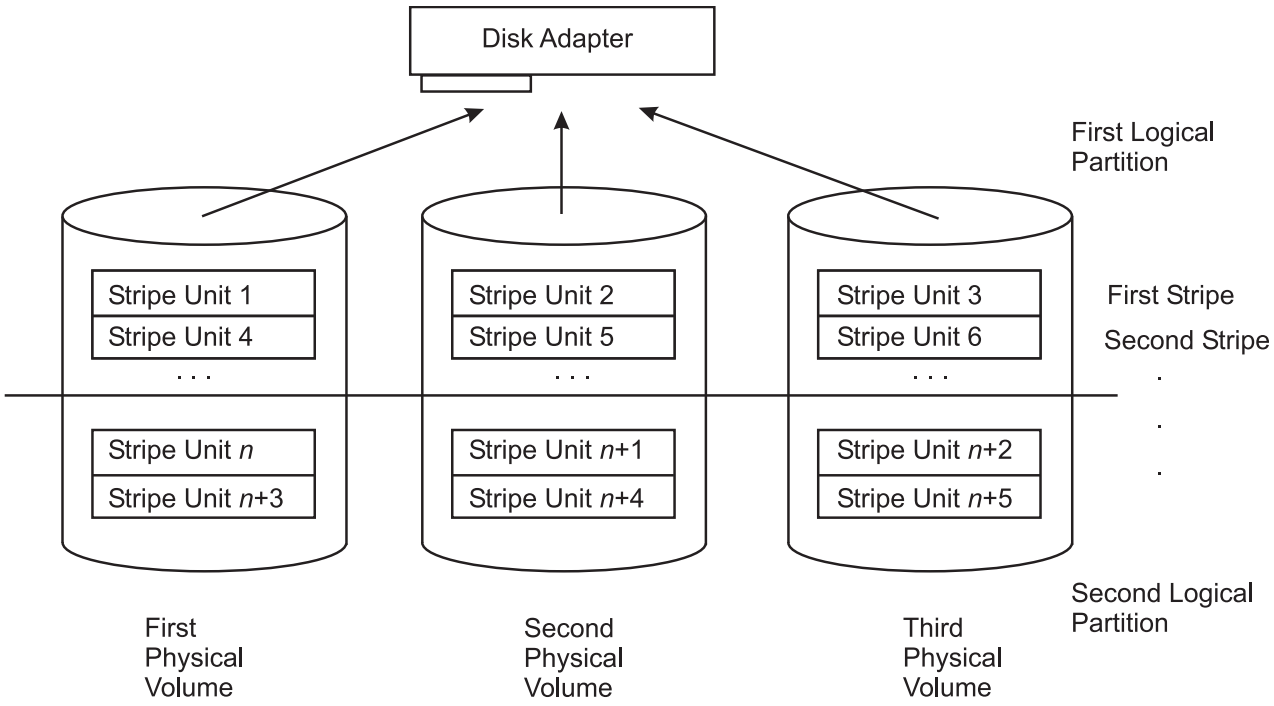
- ใช้สำเนา LP สามชุด (การทำมิเรอร์สองครั้ง)
- ตั้งค่าตรวจสอบการบันทึกเป็น Yes
- ตั้งค่านโยบาย inter เป็น Minimum (การทำมิเรอร์สำเนา = # ของ PVs)
- ตั้งค่านโยบายการจัดตารางเวลาเป็น Sequential
- ตั้งค่านโยบายการจัดสรรเป็น Strict (ไม่มีการทำมิเรอร์บน PV เดียวกัน)
- กลุ่มวอลุ่มต้องมีฟิสิคัลวอลุ่มอย่างน้อยสามรายการ
- ทำมิเรอร์สำเนาบนฟิสิคัลวอลุ่มที่แนบกับบัส อะแด็ปเตอร์ และแหล่งจ่ายไฟแยกต่างหาก

การมีฟิสิคัลวอลุ่มอย่างน้อยสามรายการช่วยให้ส่วนประกอบครบสมบูรณ์ในกรณีที่ฟิสิคัลวอลุ่มหนึ่งไม่พร้อมใช้งาน การใช้บัส อะแด็ปเตอร์ และแหล่งจ่ายไฟแยกต่างหากช่วยให้สามารถใช้สำเนาที่ไม่ได้แนบกับอุปกรณ์ที่ล้มเหลวได้

การปรับ striping โลจิคัลวอลุ่ม

Striping คือเทคนิคสำหรับการกระจายข้อมูลในโลจิคัลวอลุ่ม บนหลายดิสก์ไดร์ฟ ในวิธีที่ความสามารถ I/O ของดิสก์ไดร์ฟสามารถมีการใช้ใน แบบขนานควบคู่ไปกับการเข้าถึงข้อมูลบน โลจิคัลวอลุ่มได้ วัตถุประสงค์หลักของ striping คือการอ่านและการบันทึก ประสิทธิภาพสูงมากของไฟล์ตามลำดับขนาดใหญ่ แต่ยังมีประโยชน์ สำหรับการเข้าถึงแบบสุ่มด้วย

ภาพสาธิตต่อไปนี้แสดงตัวอย่าง



รูปที่ 19. Striped โลจิคัลวอลุ่ม /dev/lvs0. ภาพสาธิตนี้ แสดงฟิสิคัลวอลุ่มหรือไดร์ฟสามรายการ แต่ละฟิสิคัลไดร์ฟ มีการแบ่งพาร์ติชันเป็นสองโลจิคัลวอลุ่ม พาร์ติชันหรือโลจิคัลวอลุ่ม แรกของไดร์ฟ 1 มีหน่วย stripe 1 และ 4 พาร์ติชัน 1 ของไดร์ฟ 2 มีหน่วย stripe 2 และ 5 และพาร์ติชัน 1 ของไดร์ฟ 3 มีหน่วย stripe 3 และ 6 พาร์ติชันที่สองของไดร์ฟหนึ่ง มีหน่วย stripe n และ n+3 พาร์ติชันที่สองของไดร์ฟ 2 มีหน่วย stripe n+1 และ n+4 พาร์ติชัน 2 ของไดร์ฟ 3 มีหน่วย stripe n+2 และ n+5

ในโลจิคัลวอลุ่มปกติ ที่อยู่ข้อมูลตรงกับลำดับของบล็อก ในฟิสิคัลพาร์ติชันที่สัมพันธ์กัน ใน striped โลจิคัลวอลุ่ม ที่อยู่ข้อมูลเป็นไปตามลำดับของ หน่วย stripe Stripe ที่สมบูรณ์ประกอบด้วยหนึ่งหน่วย stripe บนแต่ละรายการของ อุปกรณ์ฟิสิคัลที่มีส่วนประกอบของ striped โลจิคัล วอลุ่ม LVM กำหนดฟิสิคัลบล็อกซึ่งมีฟิสิคัลไดร์ฟตรงกับบล็อกที่จะอ่าน หรือบันทึก ถ้ามีไดร์ฟเกี่ยวข้องมากกว่า หนึ่งไดร์ฟ การดำเนินงาน I/O ที่จำเป็นจะมีการจัดตารางเวลาสำหรับ ไดร์ฟทั้งหมดพร้อมกัน

ดังตัวอย่าง lvs0 ที่สมมุติขึ้นมีขนาด stripe-unit เป็น 64 KB ประกอบด้วยพาร์ติชัน 2 MB หกรายการ และมี journaled file system (JFS) ถ้าแอ็พพลิเคชันกำลังอ่านไฟล์ตามลำดับขนาดใหญ่และ read-ahead อยู่ในสภาวะคงที่ การอ่านแต่ละรายการจะส่งผลให้มีการจัดตารางเวลา สองหรือสาม I/Os ให้กับแต่ละดิสก์ไดร์ฟเพื่ออ่านหน้าทั้งหมด แปดหน้า (สมมุติว่าไฟล์อยู่บนบล็อกที่ต่อเนื่องกัน ในโลจิคัลวอลุ่ม) การดำเนินงานอ่านทำในลำดับที่กำหนดโดย ไดรเวอร์อุปกรณ์ดิสก์ ข้อมูลที่ร้องขอประกอบขึ้นจาก ชิ้นของอินพุตต่างๆ และถูกส่งคืนไปยังแอ็พพลิเคชัน

แม้ว่าแต่ละอุปกรณ์ดิสก์จะมีเวลาแฝงแรกเริ่มที่แตกต่างกัน ขึ้นอยู่กับว่า accessor อยู่ที่ใดเมื่อเริ่มต้นการดำเนินงาน หลังจากกระบวนการเข้าสู่สภาวะคงที่แล้ว ทั้งสามดิสก์ควรจะมีการอ่าน ใกล้เคียงกับความเร็วสูงสุด

การออกแบบโลจิคัลวอลุ่มที่ถูกถอดออก

คุณต้องจัดเตรียมข้อมูลบางส่วนเพื่อกำหนดโลจิคัลวอลุ่ม ที่ถูกถอดออก

เมื่อกำหนดโลจิคัลวอลุ่มที่ถูกถอดออก คุณต้องระบุ:

ไดร์ฟ ฟิสิคัลไดร์ฟอย่างน้อยสองตัว ไดร์ฟที่ใช้ควรมีกิจกรรมอื่นๆ เพียงเล็กน้อย เมื่อ I/O ที่ต่อเนื่องกันซึ่งมีผลการทำงานที่สำคัญ เข้าแทนที่ การรวมกันของดิสก์อะแด็ปเตอร์และดิสก์ไดร์ฟบางส่วนยังกำหนดให้แบ่งเวิร์กโหลดของโลจิคัลวอลุ่มที่ถอดออกระหว่างอะแด็ปเตอร์สองตัวขึ้นไป

ขนาดหน่วยที่ถอดออก
แม้ว่าไดร์ฟสามารถมีได้ 2 ตัวซึ่งมีขนาดตั้งแต่ 4 KB ถึง 128 KB ให้พิจารณาการอ่านตามลำดับก่อนหน้า เนื่องจากการดำเนินการแบบนั้นคือกลไก ที่ออกคำสั่งให้อ่านโดยส่วนใหญ่ วัตถุประสงค์คือ มีการดำเนินการอ่านก่อนหน้านี้แต่ละกระบวนการ ที่ส่งผลอย่างน้อยหนึ่ง I/O ซึ่งมีจำนวนที่เท่ากันกับดิสก์ไดร์ฟแต่ละตัว (โปรดดูรูปภาพก่อนหน้า)

ขนาด จำนวนของฟิสิคัลพาร์ติชันที่จัดสรรให้กับโลจิคัลวอลุ่ม ต้องเป็นจำนวนของดิสก์ไดร์ฟที่ใช้อย่างครบถ้วน

แอ็ตทริบิวต์
โลจิคัลวอลุ่มที่ถอดออกสามารถมิรเรอร์ และสามารถตั้งค่าจำนวนสำเนา มากกว่า 1

การปรับสำหรับ striped logical volume I/O

ดิสก์ I/Os ตามลำดับและแบบสุ่มได้รับประโยชน์จาก disk striping

เทคนิคต่อไปนี้ให้ระดับสูงสุดของผลผลิต I/O ตามลำดับ:

- กระจายโลจิคัลวอลุ่มบนฟิสิคัลวอลุ่มมากที่สุดเท่าที่เป็นไปได้
- ใช้อะแด็ปเตอร์ให้มากที่สุดเท่าที่เป็นไปได้สำหรับฟิสิคัลวอลุ่ม
- สร้างกลุ่มวอลุ่มแยกต่างหากสำหรับ striped โลจิคัลวอลุ่ม
- ตั้งค่าขนาด stripe-unit เป็น 64 KB
- ตั้งค่าคำสั่ง *minpgahead* เป็น 2 โดยใช้คำสั่ง **ioo** โปรดดู "การปรับประสิทธิภาพการอ่านตามลำดับ" ในหน้า 269
- ตั้งค่า *maxpgahead* เป็น 16 เท่าของจำนวนดิสก์ไดร์ฟ โดยใช้คำสั่ง **ioo** ซึ่งส่งผลให้ทำ page-ahead ในหน่วยของขนาด stripe-unit (64 KB) คูณจำนวนของดิสก์ไดร์ฟ ส่งผลให้เกิดการอ่านของหนึ่งหน่วย stripe จากแต่ละดิสก์ไดร์ฟสำหรับแต่ละ การดำเนินงาน read-ahead
- ร้องขอ I/Os ในจำนวน 64 KB คูณจำนวนของดิสก์ไดร์ฟ ค่านี้เท่ากับค่า *maxpgahead*
- แก้ไข *maxfree* โดยใช้คำสั่ง **ioo** เพื่อสนับสนุนการเปลี่ยนแปลงใน *maxpgahead* (*maxfree* = *minfree* + *maxpgahead*) โปรดดู "ค่าสำหรับพารามิเตอร์ minfree และ maxfree" ในหน้า 167
- ใช้ I/O บัฟเฟอร์ 64 ไบต์ ถ้าโลจิคัลวอลุ่มจะครอบครองฟิสิคัลไดร์ฟ ที่เชื่อมต่ออยู่กับดิสก์อะแด็ปเตอร์สองรายการขึ้นไป I/O บัฟเฟอร์ที่ใช้ ควรมีการจัดสรรแบบ 64 ไบต์ การทำเช่นนี้หลีกเลี่ยงการที่ LVM จัดลำดับ I/Os ไปยังดิสก์อื่น รหัสต่อไปนี้จะให้ตัวชี้บัฟเฟอร์แบบ 64 ไบต์:

```
char *buffer;
buffer = malloc(MAXBLKSIZE+64);
buffer = ((int)buffer + 64) & ~0x3f;
```

ถ้า striped โลจิคัลวอลุ่มอยู่บน raw logical volumes และมีการทำการบันทึก ใหญ่กว่า 1.125 MB ใน striped raw logical volumes เหล่านี้ การเพิ่ม พารามิเตอร์ *lvm_bufcnt* ด้วยคำสั่ง **ioo** อาจเพิ่มผลผลิตของกิจกรรมการบันทึก โปรดดู "การปรับบัฟเฟอร์ของระบบไฟล์" ในหน้า 275

ตัวอย่างข้างบนใช้สำหรับ JFS striped logical volume เทคนิคเดียวกันใช้กับ enhanced JFS ยกเว้นว่าพารามิเตอร์ **ioo** ที่ใช้จะเป็น enhanced JFS equivalents

นอกจากนี้ ยังเป็นแนวคิดที่ดีที่จะผสม striped และ non-striped logical volumes ในฟิสิคัลวอลุ่มเดียวกัน ฟิสิคัลวอลุ่มทั้งหมดควรจะมีขนาดเดียวกัน ภายในชุดของ striped โลจิคัลวอลุ่ม

ความเกี่ยวข้องกับผลการทำงานของโลจิคัลวอลุ่มที่มิเรอร์

AIX อนุญาตให้ striping และมิรเรอร์ไปพร้อมกันบน โลจิคัลวอลุ่มเดียวกัน

คุณลักษณะนี้จะจัดเตรียมกลไกที่สะดวกสำหรับหน่วยเก็บ ที่มีผลการทำงานสูง การวัดค่าบ่งชี้ว่า ผลการทำงานของการอ่านและการเขียนระบบไฟล์ของ striping และการมอนิเตอร์มีค่าเท่ากันในกรณีที่ไม่มีมิเรอร์ ซึ่งคุณมีเป็นสองเท่าของดิสก์จำนวนมาก

ระบบไฟล์ที่เขียนจะมีข้อได้เปรียบจากการทำแคชในระบบไฟล์ ซึ่งอนุญาตให้ใช้ความสามารถในการพิจารณาให้ซ้อนทับการเขียนลงบนดิสก์ด้วยโปรแกรมที่กำหนดค่าเริ่มต้นการเขียนไว้ ผลการทำงานสำหรับการเขียนข้อมูลดิบอาจต้องมีความอดทน เนื่องจากการเขียนเป็นแบบซิงโครนัส การเขียนทั้งสองแบบต้องเสร็จสิ้นก่อนที่การควบคุมจะส่งคืนไปยังโปรแกรมที่เริ่มต้น การดำเนินการเขียนที่ใหญ่กว่า จะเพิ่มทรูพุตการเขียนข้อมูลดิบ และ Mirror Write Consistency (MWC) จะส่งผลถึงผลการทำงานในกรณีนี้

สรุป striping และการมอนิเตอร์จะอนุญาตให้ใช้หน่วยเก็บที่มีมากเกินไปสำหรับการเข้าถึงผลการทำงาน ที่สูงมาก

การใช้ raw disk I/O

บางแอ็พพลิเคชัน เช่น ฐานข้อมูล ไม่ต้องการระบบไฟล์ เนื่องจาก แอ็พพลิเคชันทำฟังก์ชันต่างๆ เช่น การบันทึก การเก็บประวัติของข้อมูล และ การแคชเอง ประสิทธิภาพของแอ็พพลิเคชันประเภทนี้โดยปกติแล้ว ดีกว่าเมื่อใช้ raw I/O แทนการใช้ไฟล์ I/O เนื่องจากไม่ต้องมีภาระงานเพิ่มเติมจากสำเนา หน่วยความจำ การบันทึก และ inode locks

เมื่อใช้ raw I/O แอ็พพลิเคชันควรใช้ไฟล์พิเศษอักขระ `/dev/rlv*` ไม่ควรใช้ไฟล์พิเศษบล็อก `/dev/lv*` เนื่องจากไฟล์นี้จะแบ่ง I/Os ขนาดใหญ่เป็นหลาย 4K I/Os ไม่ควรใช้ `/dev/rhdisk*` และ `/dev/hdisk*` raw disk interface เนื่องจากทำให้ประสิทธิภาพด้อยลงและยังอาจส่งผลให้ เกิดปัญหาความไม่สอดคล้องกันของข้อมูลอีกด้วย

การใช้การเรียก sync และ fsync

ถ้าเปิดไฟล์ด้วย O_SYNC หรือ O_DSYNC แต่ละการบันทึกจะทำให้ข้อมูล สำหรับการบันทึกนั้นถูกฟลัชไปยังดิสก์ก่อนการส่งคืนการบันทึก ถ้าการบันทึกส่งผลให้เกิดการจัดสรรดิสก์ใหม่ (การขยายไฟล์แทนการบันทึกทับ หน้าที่มีอยู่) การบันทึกยังจะทำให้เกิดการบันทึกไฟล์บันทึก JFS ที่ตรงกัน ด้วย

การซิงโครไนซ์ที่บังคับใช้ของเนื้อหาของหน่วยความจำจริงและดิสก์เกิดขึ้นได้ หลายวิธีดังนี้:

- แอ็พพลิเคชันโปรแกรมทำการเรียก **fsync( )** สำหรับไฟล์ที่ระบุ ซึ่งส่งผลให้หน้าทุกหน้าซึ่งมีข้อมูลที่แก้ไขสำหรับไฟล์นั้นถูกบันทึกลงใน ดิสก์ การบันทึกเสร็จสมบูรณ์เมื่อการเรียก **fsync( )** กลับคืน ไปยังโปรแกรม
- แอ็พพลิเคชันโปรแกรมทำการเรียก **sync( )** ซึ่งส่งผลให้หน้าไฟล์ทั้งหมด ในหน่วยความจำซึ่งมีข้อมูลที่แก้ไขถูกจัดตารางเวลาสำหรับการบันทึก ลงในดิสก์ การบันทึกไม่จำเป็นต้องเสร็จสมบูรณ์เมื่อการเรียก **sync( )** กลับคืน ไปยังโปรแกรม
- ผู้ใช้สามารถป้อนคำสั่ง **sync** ซึ่งจะออกใช้การเรียก **sync( )** ได้ อีกครั้งที่การบันทึกบางรายการอาจไม่เสร็จสมบูรณ์เมื่อมีการพร้อมต์ให้ผู้ใช้ ป้อนข้อมูล (หรือเมื่อประมวลผลคำสั่งถัดไปใน shell script)
- `/usr/sbin/syncd` daemon ออกใช้การเรียก **sync( )** ในช่วงเวลาที่สม่ำเสมอ โดยปกติคือทุก 60 วินาที การทำเช่นนี้ช่วยให้มั่นใจว่าระบบ ไม่ได้สะสมข้อมูลจำนวนมากที่มีอยู่เฉพาะใน RAM ที่ลบเลือนได้

การดำเนินงาน sync มีผลกระทบหลายอย่าง นอกเหนือจากการใช้ CPU เพียงเล็กน้อย:

- ทำให้การบันทึกเกาะกลุ่มกัน แทนที่จะกระจายออก
- ทำให้มีข้อมูลระบบที่จะบันทึกอย่างน้อย 28 KB แม้ว่าไม่มีกิจกรรม I/O ตั้งแต่ การดำเนินงาน **sync** ก่อนหน้านี้
- เร่งการบันทึกข้อมูลลงในดิสก์ ยกเลิกขั้นตอนวิธี write-behind ผลกระทบนี้พบได้มากในโปรแกรมที่ออกใช้การเรียก **fsync** ( ) หลังจากทุกการบันทึก
- เมื่อการเรียก **sync**( ) หรือ **fsync**( ) เกิดขึ้น เร็กคอร์ดไฟล์บันทึกจะถูก บันทึกลงในอุปกรณ์บันทึก JFS เพื่อบ่งชี้ว่าข้อมูลที่แก้ไขมีการ committed บนดิสก์แล้ว

การตั้งค่าขีดจำกัดคิวของ SCSI อะแด็ปเตอร์และอุปกรณ์ดิสก์

ระบบปฏิบัติการสามารถบังคับใช้ขีดจำกัดจำนวนของการร้องขอ I/O ที่สามารถคงค้างจาก SCSI อะแด็ปเตอร์ไปยัง SCSI บัสหรือดิสก์ไดร์ฟที่กำหนด ขีดจำกัดเหล่านี้มีไว้เพื่อใช้ความสามารถของฮาร์ดแวร์ ในการจัดการหลายคำร้องขอ ในขณะที่ทำให้มั่นใจว่าขั้นตอนวิธีค้นหาการใช้ ประโยชน์สูงสุดในไดรเวอร์อุปกรณ์สามารถทำงานได้อย่างมีประสิทธิภาพ

สำหรับอุปกรณ์ที่ไม่ใช่ของ IBM บางครั้งอาจเหมาะสมที่จะแก้ไข ค่าขีดจำกัดคิวดีฟอลต์ที่เลือกไว้แล้วเพื่อจัดการกรณีเลวร้ายที่สุดที่อาจเกิดขึ้นได้ ส่วนต่อไปนี้อธิบายสถานการณ์ซึ่งควรจะเปลี่ยนค่าดีฟอลต์ และแนะนำค่าใหม่

ดิสก์ไดร์ฟที่ไม่ใช่ของ IBM

สำหรับดิสก์ไดร์ฟ IBM จำนวนดีฟอลต์ของคำร้องขอที่สามารถคงค้างอยู่ ณ เวลาที่กำหนดใดๆ คือ 3 ค่านี้ได้มาจากข้อควรพิจารณาเกี่ยวกับประสิทธิภาพที่ซับซ้อน และไม่มีการจัดเตรียมอินเตอร์เฟสโดยตรงสำหรับการเปลี่ยนค่านี้ ความลึกของคิวฮาร์ดแวร์ ดีฟอลต์สำหรับดิสก์ไดร์ฟที่ไม่ใช่ของ IBM คือ 1 ถ้าดิสก์ไดร์ฟที่ไม่ใช่ของ IBM เฉพาะสามารถบัฟเฟอร์หลายการร้องขอได้ คำอธิบายของระบบของอุปกรณ์นั้น ควรจะเปลี่ยนตามนั้น

ดังตัวอย่าง ลักษณะดีฟอลต์ของดิสก์ไดร์ฟที่ไม่ใช่ของ IBM แสดงขึ้นโดยใช้คำสั่ง **lsattr** ดังนี้:

```
# lsattr -D -c disk -s scsi -t osdisk
pvid          none Physical volume identifier      False
clr_q         no   Device CLEARS its Queue on error
q_err         yes  Use QERR bit
q_type        none Queuing TYPE
queue_depth   1    Queue DEPTH
reassign_to   120  REASSIGN time out value
rw_timeout    30   READ/WRITE time out value
start_timeout 60   START unit time out value
```

คุณสามารถใช้คำสั่ง SMIT (พาธด่วนคือ **smitty chgdsk**) หรือ **chdev** เพื่อเปลี่ยนพารามิเตอร์เหล่านี้ ตัวอย่างเช่น ถ้าระบบของคุณมี SCSI ดิสก์ไดร์ฟที่ไม่ใช่ของ IBM hdisk5 คำสั่งต่อไปนี้จะเปิดใช้งานการจัดคิวสำหรับอุปกรณ์นั้น และตั้งค่าความลึกของคิวเป็น 3:

```
# chdev -l hdisk5 -a q_type=simple -a queue_depth=3
```

ดิสก์อาร์เรย์ที่ไม่ใช่ของ IBM

ดิสก์อาร์เรย์ที่ไม่ใช่ของ IBM เช่นเดียวกับดิสก์ไดร์ฟที่ไม่ใช่ของ IBM เป็นคลาสดิสก์ คลาสย่อย SCSI, ชนิด `osdisk` (SCSI ดิสก์ไดร์ฟอื่น)

ดิสก์อาร์เรย์ปรากฏขึ้นในระบบปฏิบัติการเป็นดิสก์ไดร์ฟเดียว แทนที่จะเป็น ดิสก์ไดร์ฟขนาดใหญ่ เนื่องจากดิสก์อาร์เรย์ประกอบด้วยฟิสิคัลดิสก์ไดร์ฟจำนวนมากอย่างแท้จริง และแต่ละฟิสิคัลดิสก์ไดร์ฟสามารถจัดการร้องขอได้หลายรายการ ดัง

นั้นจึงต้องตั้งค่า ความลึกคิวของอุปกรณ์ดิสก์อาร์เรย์เป็นค่าที่สูงเพียงพอให้สามารถใช้งานอุปกรณ์ ฟิสิคัลทั้งหมดได้อย่างมีประสิทธิภาพ ตัวอย่างเช่น ถ้า hdisk7 เป็นดิสก์อาร์เรย์ที่ ไม่ใช่ของ IBM ซึ่งมีแปดดิสก์ การเปลี่ยนแปลงที่เหมาะสมอาจเป็นดังนี้:

```
# chdev -l hdisk7 -a q_type=simple -a queue_depth=24
```

ถ้าดิสก์อาร์เรย์มีการติดตั้งผ่านทางอะแด็ปเตอร์บัส SCSI-2 Fast/Wide SCSI ยังอาจจำเป็นต้องเปลี่ยนขีดจำกัดการร้องขอที่ค้างอยู่สำหรับ บัสนั้นด้วย

การขยายคอนฟิกูเรชัน

น่าเสียดายที่ความพยายามในการปรับผลการทำงานทุกครั้งจะเข้าใกล้ จุดของการส่งคืนที่ลดน้อยลงในที่สุด ดังนั้นจึงมีคำถามที่ว่า "ฮาร์ดแวร์ใดที่ฉันต้องการ ฮาร์ดแวร์นั้นราคาเท่าไหร่ และวิธีที่ดีที่สุดที่ฉันสามารถใช้ฮาร์ดแวร์นั้นได้คือวิธีใด?" คำถามนั้น เป็นคำถามที่ถามกันบ่อยพร้อมกับเวิร์กโหลดที่จำกัดดิสก์ เนื่องจาก จำนวนของตัวแปรที่มีขนาดใหญ่

การเปลี่ยนแปลงที่อาจปรับปรุงผลการทำงานของเวิร์กโหลดที่จำกัดดิสก์ ประกอบด้วย:

- การเพิ่มดิสก์ไดร์ฟ และการกระจายข้อมูลที่มีอยู่ระหว่างดิสก์ไดร์ฟเหล่านั้น ซึ่งจะแบ่งโหลด I/O ระหว่างตัวประเมินผลจำนวนมาก
- การรับดิสก์ไดร์ฟที่เร็วกว่าเพื่อเสริมหรือแทนที่ดิสก์ไดร์ฟที่มีอยู่เดิม สำหรับข้อมูลที่มีการใช้ในระดับสูง
- การเพิ่มดิสก์อะแด็ปเตอร์หนึ่งตัวขึ้นไปเพื่อพ่วงต่อกับดิสก์ไดร์ฟปัจจุบันหรือดิสก์ไดร์ฟใหม่
- การเพิ่ม RAM ให้กับระบบและการเพิ่มพารามิเตอร์ *minperm* และ *maxperm* ของ VMM เพื่อปรับปรุงการทำแคชในหน่วยความจำของข้อมูลที่มีการใช้ในระดับสูง

สำหรับแนวทางที่เน้นไปที่คอนฟิกูเรชันและเวิร์กโหลดของคุณ คุณสามารถใช้เครื่องจำลองแบบผลักดันให้เกิดการวัดค่า เช่น BEST/1

การใช้ RAID

Redundant Array of Independent Disks (RAID) คือคำศัพท์ที่ใช้ อธิบายเทคนิคของการปรับปรุงการมีอยู่ของข้อมูล โดยใช้อาร์เรย์ของดิสก์และ ระเบียบวิธี data-striping ต่างๆ

ดิสก์อาร์เรย์คือกลุ่มของดิสก์ไดร์ฟที่ทำงานร่วมกัน เพื่อให้ได้อัตราการโอนย้ายข้อมูลและ I/O ที่สูงขึ้นกว่าอัตราที่ได้จาก ไดร์ฟขนาดใหญ่เครื่องเดียว อาร์เรย์คือชุดของหลายดิสก์ไดร์ฟบวกกับคอนโทรลเลอร์เฉพาะงาน (อาร์เรย์คอนโทรลเลอร์) ที่เก็บประวัติของวิธีการแจกจ่ายข้อมูลบน ไดร์ฟต่างๆ ข้อมูลของไฟล์เฉพาะถูกบันทึกลงในเซกเมนต์ในไดร์ฟต่างๆ ที่แตกต่างกันในอาร์เรย์แทนที่จะบันทึกลงในไดร์ฟเดียว

อาร์เรย์ ยังสามารถให้การทำซ้ำข้อมูล ดังนั้นจึงไม่มีข้อมูลสูญหายถ้าไดร์ฟหนึ่ง (ฟิสิคัลไดร์ฟ) ในอาร์เรย์ล้มเหลว ขึ้นอยู่กับระดับ RAID ข้อมูลจะมีการ ทำมิเรอร์หรือ striped อย่างใดอย่างหนึ่ง

อาร์เรย์ย่อยมีอยู่ภายในระบบย่อย อาร์เรย์ ขึ้นอยู่กับวิธีการตั้งค่าคอนฟิกของคุณ ระบบย่อยอาร์เรย์อาจมีอาร์เรย์ย่อยหนึ่งรายการขึ้นไป ซึ่งเรียกอีกอย่างว่า Logical Units (LUN) แต่ละ LUN มีลักษณะเฉพาะของตัวเอง (ตัวอย่างเช่น ระดับ RAID, ขนาดโลจิคัลบล็อกและ ขนาดโลจิคัลยูนิต) จากระบบปฏิบัติการ จะมองเห็นแต่ละอาร์เรย์ย่อยเป็น hdisk เดี่ยวที่มีชื่อเฉพาะของตนเอง

ขั้นตอนวิธี RAID สามารถนำไปใช้เป็น ส่วนประกอบหนึ่งของซอฟต์แวร์ระบบไฟล์ของระบบปฏิบัติการ หรือใช้เป็นส่วนประกอบหนึ่งของไดรเวอร์อุปกรณ์ดิสก์ (ทั่วไปสำหรับ RAID 0 และ RAID 1) ขั้นตอนวิธีเหล่านี้สามารถทำโดยตัวประมวลผลที่ฝังอยู่แบบโลคัลบนฮาร์ดแวร์ RAID อะแด็ปเตอร์ โดยทั่วไป ฮาร์ดแวร์ RAID อะแด็ปเตอร์ให้ประสิทธิภาพที่ดีกว่าซอฟต์แวร์ RAID เนื่องจากตัวประมวลผลที่ฝังไว้ offload ตัวประมวลผลระบบหลักโดยการทำขั้นตอนวิธีซับซ้อน บางครั้งใช้วงจรเฉพาะงานสำหรับการโอนย้ายข้อมูลและการจัดวาง

อ็อพชัน RAID ที่สนับสนุนโดย LVM

AIX LVM สนับสนุนอ็อพชัน RAID สามอ็อพชัน

Item	Descriptor
RAID 0	Striping
RAID 1	การมอนิเตอร์
RAID 10 หรือ 0+1	การมอนิเตอร์และ striping

การใช้ fast write cache

Fast write cache (FWC) คือแคชแบบไม่ลบเลือนที่เป็นอ็อพชันซึ่ง ให้การทำซ้ำกับอะแด็ปเตอร์แคชมาตรฐาน FWC ติดตามการบันทึกที่ยังไม่ได้ committed ในดิสก์

Fast write cache สามารถพัฒนาเวลาการตอบกลับสำหรับการดำเนินงาน บันทึกได้เป็นอย่างมาก อย่างไรก็ตาม ต้องระวังไม่ให้แคชเต็มไปด้วยการร้องขอการบันทึก เร็วกว่าอัตราที่แคชสามารถ destage ข้อมูล FWC ยังอาจส่งผลเสียต่ออัตรา I/O ต่ำสุด เนื่องจากต้องการการประมวลผลเพิ่มเติม ใน adapter card เพื่อกำหนดว่าข้อมูลที่กำลังโอนย้ายอยู่ในแคช หรือไม่

โดยปกติแล้ว fast write cache มีข้อดีอย่างมากในเวิร์กโหลดเฉพาะงาน ตัวอย่างเช่น การคัดลอกฐานข้อมูลบนดิสก์ชุดใหม่ ถ้ากระจาย fast write cache บนหลายอะแด็ปเตอร์ ประโยชน์ยิ่งมากขึ้นเป็นทวีคูณ

FWC ยังช่วยลดปัญหาคอขวดบันทึก JFS เนื่องจากคุณสมบัติต่อไปนี้ของ บันทึก JFS:

1. บันทึก JFS เน้นการบันทึก FWC ไม่ได้แคชข้อมูลที่ไม่ได้แก้ไข
2. การบันทึกมีขนาดเล็กและเกิดขึ้นบ่อย เนื่องจากความจุแคชไม่มากนัก แคชจึงทำงานได้ดีที่สุดสำหรับ I/Os ขนาดเล็กอัตราสูง ซึ่งมีการรวบรวมเข้าด้วยกัน เป็นฟิสิคัล I/O ขนาดใหญ่ขึ้นในอะแด็ปเตอร์ I/Os ที่ใหญ่ขึ้นมีแนวโน้มจะมีประสิทธิภาพที่ดีขึ้นเนื่องจาก โดยปกติ มีการหมุนเวียนดิสก์ที่ต้องบันทึกข้อมูลน้อยลง
3. โดยปกติแล้ว ไฟล์บันทึกไม่ใหญ่มากเมื่อเปรียบเทียบกับขนาดแคช ดังนั้นไฟล์ บันทึกจึงไม่มีแนวโน้มจะ "ล้าง" แคชบ่อยนัก ด้วยเหตุนี้ ไฟล์บันทึกจึงไม่ได้รับประโยชน์ จากการบันทึกข้อมูลแคชที่มีอยู่ใหม่ แม้ว่าตัวควบคุมอาร์เรย์อื่นๆ เกี่ยวกับแคชการบันทึกสามารถใช้กับไฟล์บันทึกได้อย่างมีประสิทธิภาพ ข้อความนี้อธิบาย ประสิทธิภาพของไฟล์บันทึกที่เกี่ยวข้องกับ FWC เท่านั้น

เมื่อแบนด์วิดท์ดิสก์เดียวมีประสิทธิภาพจำกัด วิธีการแก้ไขอย่างหนึ่งคือ strip หลายอุปกรณ์ RAID 5 เข้าในโลจิคัลวอลุ่ม ขนาด strip คือ 64 KB คูณด้วย จำนวนของดิสก์ข้อมูลใน RAID 5 เมื่ออะแด็ปเตอร์มีการตั้งค่าคอนฟิก สำหรับ RAID 5 การบันทึกที่เท่ากับหรือใหญ่กว่าขนาด strip จะ เลี่ยงแคช นี่เป็นเหตุผลว่าเพราะอะไรการบันทึก 128 KB ใน 2+p อาร์เรย์ที่มี FWC จึงช้ากว่าการบันทึก 127 KB และเท่ากับการบันทึก 128 KB ใน 2+p ที่ไม่มี FWC ที่เป็นเช่นนี้เพื่อป้องกันไม่ให้ I/O ตามลำดับจำนวนมาก "ล้าง" แคช

ความล้มเหลวของ I/O อย่างรวดเร็วสำหรับอุปกรณ์ไฟเบอร์แชนเนล

AIX สนับสนุนความล้มเหลวของ I/O อย่างรวดเร็วสำหรับอุปกรณ์ไฟเบอร์แชนเนล (FC) หลังจากที่ลิงก์เหตุการณ์ในสภาวะแวดล้อมแบบสวิตช์

ถ้าไดรเวอร์อะแด็ปเตอร์ FC ตรวจพบลิงก์เหตุการณ์ เช่น ลิงก์ที่หายไประหว่างอุปกรณ์หน่วยเก็บ ข้อมูลและสวิตช์ ไดรเวอร์อะแด็ปเตอร์ FC จะรอระยะเวลาสั้นๆ ประมาณ 15 วินาที เพื่อให้ fabric มั่นคง ที่จุดนี้ ถ้าไดร์เวอร์อะแด็ปเตอร์ FC ตรวจพบว่า อุปกรณ์ไม่ได้อยู่บนโครงสร้าง ไดร์เวอร์อะแด็ปเตอร์จะเกิดความล้มเหลวสำหรับ I/O ทั้งหมดที่อะแด็ปเตอร์ไดร์เวอร์ การลองซ้ำ I/O ใหม่หรือในอนาคตของ I/Os ที่ล้มเหลวจะล้มเหลวในทันที ตามอะแด็ปเตอร์ จนกว่าไดรเวอร์อะแด็ปเตอร์ตรวจพบว่าอุปกรณ์เชื่อมต่อกับ fabric อีกครั้งแล้ว

ความล้มเหลวของ I/O แบบเร็วจะถูกควบคุมโดยแอ็ตทริบิวต์อุปกรณ์ **fscsi fc_err_recov** ค่าติดตั้งดีฟอลต์สำหรับแอ็ตทริบิวต์นี้ คือ `delayed_fail` ซึ่งเป็นลักษณะการทำงานที่ล้มเหลวของ I/O ซึ่งพบในเวอร์ชันก่อนหน้านี้ของ AIX หากต้องการเปิดใช้งานความล้มเหลวของ I/O แบบเร็ว ให้ตั้งค่าแอ็ตทริบิวต์นี้เป็น `fast_fail` ตามที่แสดงในตัวอย่าง:

```
chdev -l fscsi0 -a fc_err_recov=fast_fail
```

สำหรับตัวอย่างนี้ อินสแตนซ์ของอุปกรณ์ **fscsi** คือ `fscsi0` ตรรกะการล้มเหลวอย่างรวดเร็วมีการเรียกใช้เมื่อไดรเวอร์อะแด็ปเตอร์ได้รับการบ่งชี้ จากสวิตช์ว่า มีเหตุการณ์ลิงก์กับพอร์ตของอุปกรณ์หน่วยเก็บข้อมูลแบบรีโมต ตามเส้นทางของ Registered State Change Notification (RSCN) จาก สวิตช์

ความล้มเหลวของ I/O แบบเร็วจะมีประโยชน์ในสถานการณ์ที่ซอฟต์แวร์แบบมัลติพาธ ถูกนำมาใช้ การตั้งค่าแอ็ตทริบิวต์ **fc_err_recov** ให้มีค่า `fast_fail` สามารถลดจำนวนครั้งของความล้มเหลวของ I/O เนื่องจากลิงก์หายไประหว่างอุปกรณ์หน่วยเก็บ และสวิตช์ ซึ่งจะสนับสนุนความล้มเหลวแบบเร็วสำหรับพาธสำรอง

ในคอนฟิกูเรชันของพาธเดี่ยว คอนฟิกูเรชันที่มีพาธเดี่ยวกับอุปกรณ์การเพจ ค่ากำหนดดีฟอลต์คือ `delayed_fail`

ความล้มเหลวของ I/O แบบเร็วต้องการข้อมูลต่อไปนี้:

- สภาวะแวดล้อมที่ทำสวิตช์แล้ว ซึ่งไม่สนับสนุนในสภาวะแวดล้อมแบบลูป ซึ่งประกอบด้วยพับลิกลูป
- อะแด็ปเตอร์เฟิร์มแวร์ FC 6227 ระดับ 3.22A 1 หรือสูงกว่า
- อะแด็ปเตอร์เฟิร์มแวร์ FC 6228 ระดับ 3.82A 1 หรือสูงกว่า
- อะแด็ปเตอร์เฟิร์มแวร์ FC 6239 ทุกระดับ
- รีลีสอะแด็ปเตอร์ FC ถัดมาทั้งหมดสนับสนุน Fast I/O Failure

ถ้าไม่ตรงกับข้อกำหนดใดๆ เหล่านี้ อุปกรณ์ **fscsi** จะบันทึกล็อกข้อผิดพลาดชนิด `INFO` เพื่อบ่งชี้ว่า ไม่ตรงกับข้อกำหนดอย่างใดอย่างหนึ่งเหล่านี้ และไม่ได้เปิดใช้งาน ความล้มเหลว I/O อย่างรวดเร็ว

อุปกรณ์ FC บางรายการสนับสนุนการเปิดใช้งานและการปิดใช้งาน ความล้มเหลว I/O อย่างรวดเร็วขณะอุปกรณ์อยู่ในสภาวะ **พร้อมใช้งาน** เมื่อต้องการตรวจสอบว่าอุปกรณ์สนับสนุนฟังก์ชันการติดตามแบบไดนามิกหรือไม่ ให้ใช้คำสั่ง **lsattr** ความล้มเหลว I/O อย่างรวดเร็วสามารถ เปลี่ยนได้สำหรับอุปกรณ์ที่สนับสนุน โดยไม่ต้องยกเลิกการกำหนดคอนฟิกและ กำหนดคอนฟิกอุปกรณ์อีกครั้ง หรือ cycling ลิงก์ การเปลี่ยนแปลงต้องมีการร้องขอ เมื่อ storage area network (SAN) fabric มั่นคง คำร้องขอจะล้มเหลว ถ้าการกู้คืนข้อผิดพลาดแอ็คทีฟอยู่ใน SAN ระหว่างเวลาของคำร้องขอ

ข้อมูลที่เกี่ยวข้อง:

คำสั่ง lsattr

การติดตามแบบไดนามิกของอุปกรณ์ไฟเบอร์แชนเนล

AIX สนับสนุนการติดตามแบบไดนามิกของอุปกรณ์ (FC)

เวอร์ชัน AIX ก่อนหน้า ผู้ใช้ต้องยกเลิกการกำหนดคอนฟิกอุปกรณ์หน่วยเก็บข้อมูล FC และอินสแตนซ์ของอุปกรณ์ อะแด็ปเตอร์ก่อนปรับเปลี่ยนค่าติดตั้ง system area network (SAN) ที่อาจส่งผลให้มีการเปลี่ยนแปลง N_Port ID (SCSI ID) ของพอร์ตหน่วยเก็บข้อมูล รีโมต

ถ้าเปิดใช้งานการติดตามแบบไดนามิกของอุปกรณ์ FC อุปกรณ์อะแด็ปเตอร์ FC จะตรวจพบเมื่อ Fibre Channel N_Port ID ของอุปกรณ์เปลี่ยนแปลงไป ไดร์เวอร์อะแด็ปเตอร์ FC จะเราต์ทราฟฟิกใหม่ซึ่งได้ถูกกำหนดไว้แล้วสำหรับอุปกรณ์นั้นให้มีแอดเดรสใหม่ ขณะที่อุปกรณ์ยังคงออนไลน์อยู่ เหตุการณ์ ที่สามารถทำให้ N_Port ID เปลี่ยนแปลงรวมถึงสถานการณ์จำลองอย่างใดอย่างหนึ่ง ต่อไปนี้:

- การย้ายสายเคเบิลระหว่างอุปกรณ์สวิตช์ และหน่วยเก็บข้อมูลจากสวิตช์พอร์ต หนึ่งไปอีกพอร์ตหนึ่ง
- การเชื่อมต่อสวิตช์แยกสองตัวโดยใช้ inter-switch link (ISL)
- การรีบูตสวิตช์

การติดตามแบบไดนามิกของอุปกรณ์ FC จะถูกควบคุมโดยแอ็ตทริบิวต์ **fscsi** ตัวใหม่ นั่นคือ `dyntrk` ค่าติดตั้งดีฟอลต์สำหรับแอ็ตทริบิวต์นี้ คือ `no` เมื่อต้องการเปิดใช้งานการติดตามแบบไดนามิกของอุปกรณ์ FC ให้ตั้งค่าแอ็ตทริบิวต์นี้เป็น `dyntrk=yes` ดังแสดงใน ตัวอย่าง

```
chdev -l fscsi0 -a dyntrk=yes
```

สำหรับตัวอย่างนี้ อินสแตนซ์ของอุปกรณ์ **fscsi** คือ `fscsi0` ตรรกะการติดตามแบบไดนามิกถูกเรียกใช้เมื่ออะแด็ปเตอร์ไดรเวอร์ได้รับ การระบุจากสวิตช์ว่ามีเหตุการณ์ลิงก์ที่เกี่ยวข้องกับ พอร์ตอุปกรณ์หน่วยเก็บข้อมูลรีโมต

การสนับสนุนการติดตามแบบไดนามิกต้องใช้การกำหนดคอนฟิกต่อไปนี้:

- สภาวะแวดล้อมที่ทำสวิตช์แล้ว ซึ่งไม่สนับสนุนในสภาวะแวดล้อมแบบลูป ซึ่งประกอบด้วยพับลิกลูป
- อะแด็ปเตอร์เฟิร์มแวร์ FC 6227 ระดับ 3.22A 1 หรือสูงกว่า
- อะแด็ปเตอร์เฟิร์มแวร์ FC 6228 ระดับ 3.82A 1 หรือสูงกว่า
- อะแด็ปเตอร์เฟิร์มแวร์ FC 6239 ทุกระดับ
- รีลีสอะแด็ปเตอร์ FC ถัดมาทั้งหมดสนับสนุน Fast I/O Failure
- อุปกรณ์ worldwide Name (Port Name) และ Node Names ต้องคง เป็นค่าคงที่ และอุปกรณ์ worldwide Name ต้องเป็นค่าเฉพาะ การเปลี่ยนแปลง worldwide Name หรือ Node Name ของอุปกรณ์ที่มี หรืออุปกรณ์ออนไลน์อาจส่งผล ให้เกิดเกิดความล้มเหลว I/O นอกจากนี้ อินสแตนซ์ของอุปกรณ์หน่วยเก็บ FC แต่ละตัวต้องมีแอ็ตทริบิวต์ **world_wide_name** และ **node_name** ชุดไฟล์ที่อัพเดตซึ่งมีแอ็ตทริบิวต์ **sn_location** (โปรดดูสัญลักษณ์แสดงหัวข้อย่อยต่อไปนี้) ต้องถูกอัพเดตด้วยเช่นกัน เพื่อเก็บแอ็ตทริบิวต์ทั้งสองค่าเหล่านั้น
- อุปกรณ์หน่วยเก็บต้องจัดเตรียมเมธอดที่เชื่อถือได้ เพื่อแตกหมายเลขลำดับเฉพาะสำหรับแต่ละ LUN ไดรเวอร์อุปกรณ์ AIX FC ไม่ตรวจสอบตำแหน่งหมายเลขลำดับโดยอัตโนมัติ เมธอดสำหรับการแยกหมายเลข ลำดับต้องถูกระบุโดยผู้จำหน่ายหน่วยเก็บข้อมูลเพื่อสนับสนุนการติดตาม แบบไดนามิกสำหรับอุปกรณ์ที่ระบุ ข้อมูลนี้จะถ่ายทอดไปยังไดรเวอร์โดยใช้แอ็ตทริบิวต์ **sn_location** ODM สำหรับอุปกรณ์หน่วยเก็บข้อมูลแต่ละตัว ถ้าดิสก์หรือไดร์เวอร์เทปตรวจพบว่าแอ็ตทริบิวต์ **sn_location** ODM หายไป บันทึกข้อผิดพลาดที่เป็นชนิด `INFO` จะถูกสร้างขึ้นและการติดตามแบบไดนามิกจะไม่ถูกเปิดใช้งาน

หมายเหตุ: เมื่อรันคำสั่ง **lsattr** บน hdisk แอ็ตทริบิวต์ **sn_location** อาจไม่แสดงขึ้น นั่นคือ ชื่อแอ็ตทริบิวต์ไม่แสดงแม้ว่าจะมีอยู่ใน ODM

- ไดร์เวอร์อุปกรณ์ FC สามารถติดตามอุปกรณ์บน *SAN fabric* ถ้า N_Port IDs บน fabric คงที่ภายใน 15 วินาที *SAN fabric* คือ fabric ดังเห็นจากโฮสต์บัสอะแด็ปเตอร์เดียว ถ้า สายเคเบิลไม่ถูก reseat หรือ N_Port IDs ยังคงเปลี่ยนแปลงหลัง เริ่มต้นไปได้ 15 วินาที I/O จะล้มเหลว
- อุปกรณ์จะไม่ถูกติดตามระหว่างอะแด็ปเตอร์บัสโฮสต์ อุปกรณ์ถูก ติดตามถ้ายังคงเห็นได้จาก HBA เดียวกันที่ถูกเชื่อมต่อด้วย ตั้งแต่เริ่มต้น

  ตัวอย่างเช่น ถ้าอุปกรณ์ A ถูกย้ายจากที่หนึ่ง ไปอีกที่หนึ่งบน fabric A ที่เชื่อมต่อกับโฮสต์บัสอะแด็ปเตอร์ A (หรือ อีกนัยหนึ่ง N_Port บน fabric A เปลี่ยนแปลง) อุปกรณ์ถูกติดตาม โดยไม่มีการแทรกแซงผู้ใช้ และ I/O ไปยังอุปกรณ์นี้สามารถทำต่อได้

  อย่างไรก็ตาม หากอุปกรณ์ A สามารถมองเห็นได้จาก HBA A แต่มองไม่เห็นจาก HBA B และอุปกรณ์ A ถูกย้ายจาก fabric ที่พ่วงต่อกับ HBA A ไปยัง fabric ที่พ่วงต่อกับ HBA B อุปกรณ์ A ไม่สามารถเข้าถึงได้บน fabric A หรือไม่สามารถเข้าถึงได้บน fabric B การแทรกแซงผู้ใช้จำเป็นต้องทำเพื่อให้สามารถมีอยู่บน fabric B โดยการรันคำสั่ง **cfgmgr** อินสแตนซ์อุปกรณ์ AIX บน fabric A ไม่สามารถใช้ได้ และอินสแตนซ์อุปกรณ์บน fabric B ต้องถูกสร้างขึ้น อุปกรณ์นี้ต้องถูกเพิ่มด้วยตนเองไปยังกลุ่มวอลุ่ม อินสแตนซ์อุปกรณ์ มัลติพาธ และอื่นๆ ตามลำดับ โพรซีเดอร์นี้คล้ายกับการลบอุปกรณ์ ออกจาก fabric A และเพิ่มอุปกรณ์ไปยัง fabric B
- ไม่สามารถดำเนินการติดตามแบบไดนามิกสำหรับอุปกรณ์ดัมพ์ข้อมูล FC ขณะที่ การดัมพ์หน่วยความจำระบบ AIX กำลังดำเนินอยู่ นอกจากนี้ ไม่สนับสนุนการติดตามแบบไดนามิก ระหว่างรีสตาร์ทระบบ หรือโดยรันคำสั่ง **cfgmgr** ไม่สามารถดำเนินการเปลี่ยนแปลง SAN ได้ขณะที่มีการดำเนินการใดๆ เหล่านี้กำลังดำเนินอยู่
- หลังจากอุปกรณ์ถูกติดตาม ODM อาจมีข้อมูลเก่า ที่เป็นเช่นนี้เนื่องจาก Small Computer System Interface (SCSI) IDs ใน ODM ไม่แสดง SCSI IDs ที่แท้จริงบน SAN อีกต่อไป ODM ยังคงอยู่ในสภาวะ นี้จนกระทั่งคำสั่ง **cfgmgr** ถูกรันด้วยตนเอง หรือ ระบบรีบูต (ถ้าไดรเวอร์ทั้งหมด รวมถึงไดรเวอร์ปลายทาง FC SCSI ของบุคคลที่สาม สนับสนุนการติดตามแบบไดนามิก) ถ้าคำสั่ง **cfgmgr** ถูกรันด้วยตนเอง ต้องรันบนอุปกรณ์ **fscsi** ที่ได้รับผลทั้งหมด การดำเนินการนี้สามารถบรรลุเป้าหมายได้โดยการรัน **cfgmgr** โดยไม่ระบุอ็อพชันใดๆ หรือโดยการรัน **cfgmgr** บนอุปกรณ์ **fscsi** แต่ละตัว

  หมายเหตุ: การรัน **cfgmgr** ที่รันไทม์เพื่อทำการวัด SCSI ID ใหม่อาจไม่อัพเดต SCSI ID ใน ODM สำหรับอุปกรณ์ หน่วยเก็บข้อมูล และเป็นเช่นนี้เมื่ออุปกรณ์หน่วยเก็บข้อมูลถูกเปิด นั่นคือ เมื่อกลุ่มวอลุ่ม varied on คำสั่ง **cfgmgr** ต้องถูกรันบนอุปกรณ์ ที่ไม่ได้เปิดใช้งาน หรือระบบต้องถูกรีบูตเพื่อทำการวัด SCSI ID ใหม่ Stale SCSI IDs ใน ODM ไม่มีผล ในทางลบต่อไดร์เวอร์ FC และการปรับเทียบ SCSI IDs ใหม่ใน ODM ไม่จำเป็นสำหรับไดร์เวอร์ FC เพื่อให้ทำงานอย่างถูกต้อง อย่างไรก็ตาม แอ็พพลิเคชันใดๆ ที่สื่อสารกับไดรเวอร์อะแด็ปเตอร์โดยตรงโดยใช้การเรียก **ioctl** และที่ใช้ค่า SCSI ID จาก ODM ต้องมีการอัพเดต (โปรดดูหัวข้อย่อยถัดไป) เพื่อหลีกเลี่ยงการใช้ SCSI IDs เก่า
- แอ็พพลิเคชันทั้งหมดและส่วนขยายเคอร์เนลที่สื่อสารกับไดร์เวอร์อะแด็ปเตอร์ FC ผ่านการเรียก **ioctl** หรือสื่อสารโดยตรงกับ entry point ของไดร์เวอร์ FC โดยตรง อย่างใดอย่างหนึ่ง ต้องสนับสนุนเวอร์ชัน 1 **ioctl** และ **scsi_buf** API ของไดร์เวอร์อะแด็ปเตอร์ FC เพื่อทำงานได้อย่างถูกต้องด้วยการติดตาม FC แบบไดนามิก แอ็พพลิเคชัน และส่วนขยายเคอร์เนลที่ไม่ได้รับความยินยอมจะอาจทำงานได้อย่างไม่ถูกต้อง หรืออาจเกิดความล้มเหลวหลังจากเหตุการณ์การติดตามแบบไดนามิก ถ้า ไดร์เวอร์อะแด็ปเตอร์ FC ตรวจพบแอ็พพลิเคชันหรือส่วนขยายเคอร์เนลที่ ไม่ได้เป็นไปตามเวอร์ชันใหม่ 1 **ioctl** และ **scsi_buf** API ล็อกข้อผิดพลาดชนิด `INFO` จะถูกสร้างและการติดตามแบบ ไดนามิกไม่ถูกเปิดใช้งานสำหรับอุปรณ์ที่แอ็พพลิเคชันหรือส่วนขยาย เคอร์เนลนี้พยายามสื่อสารด้วย

  สำหรับส่วนขยายเคอร์เนล ISVs ที่กำลังพัฒนา หรือแอ็พพลิเคชันที่สื่อสารกับสแต็ก AIX Fibre Channel Driver โปรดอ้างอิง คำสั่ง FCP, iSCSI และ Virtual SCSI Client Adapter Device Driver ioctl ที่จำเป็น และการทำความเข้าใจกับโครงสร้าง scsi_buf สำหรับการเปลี่ยนแปลงที่จำเป็นเพื่อสนับสนุนการติดตามแบบไดนามิก

- แม้เมื่อเปิดใช้การติดตามแบบไดนามิก ผู้ใช้ต้องทำการเปลี่ยนแปลง SAN เช่นการย้ายหรือการสลับสายเคเบิล และการสร้างลิงก์ ISL ระหว่าง การดูแลรักษาหน้าต่าง ไม่สนับสนุนให้ทำการเปลี่ยนแปลง SAN ระหว่างรันการผลิตแบบเต็ม เนื่องจากช่วงเวลาในการรันการเปลี่ยนแปลง SAN สั้นเกินไป ตัวอย่างเช่น สายเคเบิลที่จัดวางไม่ถูกต้อง จะทำให้ I/O ล้มเหลว การรันการดำเนินการเหล่านี้ในระหว่างระยะเวลาของทราฟฟิกที่มีปริมาณน้อย หรือไม่มีทราฟฟิกจะลดผลกระทบของ I/O ที่เกิดความล้มเหลว
- การติดตามแบบไดนามิกบนพาร์ติชัน AIX เปิดให้ซอฟต์แวร์ที่มี วัตถุประสงค์ที่แสดงในการเรียกคืนเหตุการณ์การกำหนดคอนฟิกจาก SAN เหตุการณ์การกำหนดคอนฟิกถูกคาดไว้ระหว่างการดำเนินการโมบิลิตี LPAR นโยบาย AIX ถูกใช้เพื่อหลีกเลี่ยงการขาดเซอร์วิสขณะจัดเตรียม เหตุการณ์โมบิลิตี หรือระหว่างเหตุการณ์โมบิลิตี ดังนั้น การติดตามแบบไดนามิกถูกเปิดใช้งานเสมอสำหรับไคลเอ็นต์อะแด็ปเตอร์ FC เสมือน และไม่สามารถปิดใช้งาน

AIX FC SCSI Disk และ FC SCSI Tape และไดร์เวอร์อุปกรณ์ FastT พื้นฐานจะสนับสนุนการติดตามแบบไดนามิก อุปกรณ์หน่วยเก็บ IBM ESS, EMC Symmetrix และ HDS สนับสนุนการติดตามแบบไดนามิก ถ้าผู้จำหน่ายมี ODM filesets ที่มีแอ็ตทริบิวต์ **sn_location** and **node_name** จำเป็น โปรดติดต่อผู้จำหน่ายหน่วยเก็บข้อมูลเพื่อขอทราบว่าระดับ ODM fileset ปัจจุบันสนับสนุนการติดตามแบบไดนามิกหรือไม่

ถ้ารายการ ODM ของผู้จำหน่ายเฉพาะไม่ได้ถูกใช้งานสำหรับอุปกรณ์หน่วยเก็บ แต่ระบบย่อยหน่วยเก็บข้อมูล ESS, Symmetrix หรือ HDS ถูกกำหนดคอนฟิก ด้วยข้อความ `MPIO Other FC SCSI Disk`, การติดตาม แบบไดนามิกได้รับการสนับสนุนสำหรับอุปกรณ์ที่มีการกำหนดคอนฟิกนี้ เมธอด นี้ใช้แทนความจำเป็นสำหรับแอ็ตทริบิวต์ **sn_location** AIX Path Control Modules (PCM) ปัจจุบันทั้งหมดจะถูกจัดส่งมาพร้อมกับ AIX ที่สนับสนุนการติดตามแบบไดนามิก

อุปกรณ์เทป STK ที่ใช้ไดร์เวอร์อุปกรณ์ AIX แบบมาตรฐานยังสนับสนุนการติดตามแบบไดนามิกที่แสดงชุดไฟล์ STK ที่มีแอ็ตทริบิวต์ **sn_location** และ **node_name** ที่จำเป็น

หมายเหตุ: การเปลี่ยนแปลง SAN ที่เกี่ยวข้องกับอุปกรณ์เทปต้องกระทำโดยไม่มี I/O แอ็คทีฟ เนื่องจาก ธรรมชาติของอุปกรณ์เทปจะเป็นแบบเรียงลำดับ การล้มเหลว I/O เพียงครั้งเดียวก็สามารถทำให้ แอ็พพลิเคชันล้มเหลว รวมถึงการสำรองข้อมูลเทป

อุปกรณ์ที่มีการกำหนดคอนฟิกด้วยข้อความ `Other FC SCSI Disk` หรือ `Other FC SCSI Tape` ไม่สนับสนุนการติดตามแบบไดนามิก

อุปกรณ์ FC บางรายการสนับสนุนการเปิดใช้งานและการปิดใช้งาน การติดตามแบบไดนามิกขณะอุปกรณ์อยู่ในสภาวะ **พร้อมใช้งาน** เมื่อต้องการตรวจสอบว่าอุปกรณ์สนับสนุนการติดตามแบบไดนามิกหรือไม่ ให้ใช้คำสั่ง **lsattr** การติดตามแบบไดนามิกสามารถเปลี่ยนได้สำหรับอุปกรณ์ที่สนับสนุน โดยไม่ต้อง ยกเลิกการกำหนดคอนฟิกและกำหนดคอนฟิกอุปกรณ์อีกครั้ง หรือ cycling ลิงก์ การเปลี่ยนแปลงต้องมีการร้องขอเมื่อ storage area network (SAN) fabric มั่นคง คำร้องขอจะล้มเหลว ถ้าการกู้คืนข้อผิดพลาดแอ็คทีฟอยู่ใน SAN ระหว่างเวลาของคำร้องขอ คำร้องขอการเปลี่ยนแปลงอาจล้มเหลวได้ ถ้าอุปกรณ์ที่เชื่อมโยง เช่น ดิสก์ และอุปกรณ์เทป ไม่สามารถ รองรับการเปลี่ยนแปลง

ข้อมูลที่เกี่ยวข้อง:

อะแด็ปเตอร์ไคลเอ็นต์ FCP, iSCSI และ Virtual SCSI ที่จำเป็น การทำความเข้าใจกับโครงสร้าง scsi_buf

คำสั่ง lsattr

Fast I/O Failure และการโต้ตอบการติดตามไดนามิก

แม้ว่า Fast I/O Failure และการติดตามไดนามิกของอุปกรณ์ Fibre Channel (FC) ทางเทคนิคแล้วเป็นคุณลักษณะแยกกัน การเปิดใช้งานแบบหนึ่ง สามารถเปลี่ยนแปลงการแปลความหมายของอีกแบบหนึ่งในบางสถานการณ์ ตารางต่อไปนี้แสดงลักษณะการทำงานที่ถูกจัดแสดงโดยไดร์เวอร์ FC ด้วยการเปลี่ยนลำดับของค่าติดตั้งเหล่านี้:

dyntrk	fc_err_recov	ลักษณะการทำงานของไดร์เวอร์ FC
ไม่ใช่	delayed_fail	ค่ากำหนดดีฟอลต์ นี่คือลักษณะที่สืบทอดซึ่งมีอยู่ในเวอร์ชันก่อนหน้านี้ของ AIX ไดร์เวอร์ FC ไม่ได้กู้คืนถ้า SCSI ID ของการเปลี่ยนอุปกรณ์ และ I/O ที่ใช้เวลานานจะล้มเหลว เมื่อการหายไปของลิงก์เกิดขึ้นระหว่างพอร์ตหน่วยเก็บแบบรีโมต และสวิตช์ ซึ่งอาจต้องการสิ่งที่ดีกว่าในสถานการณ์แบบพาธเดี่ยว หากส่วนสนับสนุนการติดตามแบบไดนามิกไม่ใช่ข้อกำหนด
ไม่ใช่	fast_fail	ถ้าไดร์เวอร์ได้รับ RSCN จากสวิตช์ เหตุการณ์นี้สามารถบ่งชี้ถึงการหายไปของลิงก์ระหว่างพอร์ตของหน่วยเก็บแบบรีโมตและสวิตช์ หลังจากที่เวลาหน่วงเริ่มต้นที่ 15 วินาที ไดร์เวอร์ FC จะเคียวรีเพื่อดูอุปกรณ์ที่อยู่บน โครงสร้างหรือไม่ ถ้าไม่ I/Os จะล้างข้อมูลกลับโดยอะแด็ปเตอร์ ความพยายามในอนาคต หรือความล้มเหลวของ I/O ที่เกิดขึ้นใหม่โดยทันทีหากอุปกรณ์ยังคงไม่ได้อยู่บนโครงสร้าง ถ้าไดร์เวอร์ FC ตรวจพบว่า อุปกรณ์อยู่บนโครงสร้าง แต่ SCSI ID เปลี่ยนแปลงไป ไดร์เวอร์อุปกรณ์ FC จะไม่ถูกกู้คืน และ I/O จะเกิดความล้มเหลวด้วยข้อผิดพลาด PERM
ใช่	delayed_fail	ถ้าไดร์เวอร์ได้รับ RSCN จากสวิตช์ เหตุการณ์นี้สามารถบ่งชี้ถึงการหายไปของลิงก์ระหว่างพอร์ตของหน่วยเก็บแบบรีโมตและสวิตช์ หลังจากที่เวลาหน่วงเริ่มต้นที่ 15 วินาที ไดร์เวอร์ FC จะเคียวรีเพื่อดูอุปกรณ์ที่อยู่บน โครงสร้างหรือไม่ ถ้าไม่ I/Os จะล้างข้อมูลกลับโดยอะแด็ปเตอร์ ความพยายามในอนาคต หรือความล้มเหลวของ I/O ที่เกิดขึ้นใหม่โดยทันทีหากอุปกรณ์ยังคงไม่ได้อยู่บนโครงสร้าง แม้ว่าไดร์เวอร์หน่วยเก็บ (ดิสก์ เทป FastT) ไดร์เวอร์อาจได้รับเวลาหน่วงเล็กน้อย (2–5 วินาที) ระหว่างที่ลอง I/O ถ้าไดร์เวอร์ FC ตรวจพบว่า อุปกรณ์อยู่บนโครงสร้าง แต่ SCSI ID เปลี่ยนแปลงไป ไดร์เวอร์อุปกรณ์ FC จะเราต์ทราฟิกใหม่กับ SCSI ID ใหม่
ใช่	fast_fail	ถ้าไดร์เวอร์ได้รับ Registered State Change Notification (RSCN) จากสวิตช์ เหตุการณ์นี้สามารถบ่งชี้ถึงการหายไปของลิงก์ระหว่างพอร์ตของหน่วยเก็บแบบรีโมต และสวิตช์ หลังจากที่เวลาหน่วงเริ่มต้นที่ 15 วินาที ไดร์เวอร์ FC จะเคียวรีเพื่อดูอุปกรณ์ที่อยู่บนโครงสร้างหรือไม่ ถ้าไม่ I/Os จะล้างข้อมูลกลับโดยอะแด็ปเตอร์ ความพยายามในอนาคตหรือความล้มเหลวของ I/O ที่เกิดขึ้นใหม่โดยทันทีหากอุปกรณ์ยังคงไม่ได้อยู่บน โครงสร้าง ไดร์เวอร์หน่วยเก็บ (ดิสก์ เทป FastT) จะไม่หน่วงเวลาระหว่างที่พยายาม ถ้าไดร์เวอร์ FC ตรวจสอบอุปกรณ์ที่อยู่บนโครงสร้าง แต่ SCSI ID เปลี่ยนแปลงไป ไดร์เวอร์อุปกรณ์ FC จะเราต์ทราฟิกใหม่กับ SCSI ID ใหม่

เมื่อปิดใช้งานการติดตามแบบไดนามิก จะมีความแตกต่างที่ทำเครื่องหมายไว้ ระหว่างค่าติดตั้ง delayed_fail และ fast_fail ของแอ็ตทริบิวต์ **fc_err_recov** อย่างไรก็ตาม ด้วยการเปิดใช้งานการติดตามแบบไดนามิก ค่าติดตั้งของแอ็ตทริบิวต์ **fc_err_recov** จะมีความสำคัญน้อยลง เนื่องจาก มีการซ้อนทับบางอย่างในการติดตามแบบไดนามิก และนโยบายการกู้คืนข้อผิดพลาด ที่ล้มเหลวแบบเร็ว ดังนั้น การเปิดใช้งานการติดตามแบบไดนามิกจะเปิดใช้งาน ตรรกะความล้มเหลวแบบเร็วที่สืบทอดกันมา

กระบวนการกู้คืนข้อผิดพลาดทั่วไปเมื่ออุปกรณ์ไม่สามารถเข้าถึงได้อีกต่อไปบนโครงสร้าง ที่เหมือนกันสำหรับค่าติดตั้ง fc_err_recov ทั้งสองแบบ ด้วยการเปิดใช้งานการติดตามแบบไดนามิก ความแตกต่างเพียงเล็กน้อยคือ ไดร์เวอร์หน่วยเก็บสามารถเลือกเพื่อรับเวลาหน่วงระหว่าง I/O ที่ลองใหม่หาก **fc_err_recov** ถูกตั้งค่าเป็น delayed_fail เหตุการณ์นี้จะเพิ่มเวลาของความล้มเหลวสำหรับ I/O โดยเพิ่มจำนวนเพิ่มเติม ขึ้นอยู่กับค่าหน่วงเวลาและจำนวนครั้งที่พยายามใช้ ก่อนที่จะเกิดความล้มเหลวของ I/O อย่างถาวร ด้วยทราฟิก I/O ที่สูงนี้ ความแตกต่างระหว่าง delayed_fail และ fast_fail อาจมองเห็นได้

ผู้ดูแลระบบ SAN อาจต้องการการทดสอบกับค่าติดตั้งเหล่านี้ เพื่อค้นหาชุดของค่าติดตังสำหรับสภาวะแวดล้อมที่ถูกต้อง

มอดูลาร์ I/O

ไลบรารีมอดูลาร์ I/O (MIO) อนุญาตให้คุณวิเคราะห์ และปรับ I/O ของแอ็พพลิเคชันในระดับของแอ็พพลิเคชันสำหรับผลการทำงานที่เหมาะสม

บ่อยครั้งที่แอ็พพลิเคชันมีตรรกะจำนวนน้อยซึ่งถูกสร้างไว้ในโปรแกรม เพื่อจัดเตรียมโอกาสให้ผู้ใช้ออปติไมซ์ผลการทำงาน I/O ของแอ็พพลิเคชัน การไม่มีการปรับ I/O ในระดับของแอ็พพลิเคชันจะทิ้งให้ผู้ใช้ขั้นปลายอยู่ที่ระบบปฏิบัติการ เพื่อจัดเตรียมกลไกการปรับสำหรับผลการทำงาน I/O ตามปกติแล้ว แอ็พพลิเคชันจำนวนมากจะรันอยู่บนระบบที่กำหนดไว้ซึ่งจะมีความต้องการที่ขัดแย้งกันสำหรับผลลัพธ์ของการมีผลการทำงาน I/O ที่สูง และที่ดีที่สุด ในชุดของพารามิเตอร์การปรับที่จัดเตรียมผลการทำงานในระดับกลางสำหรับแอ็พพลิเคชันแบบผสมผสาน ไลบรารี MIO จะแสดงความต้องการสำหรับเมธอดในระดับของแอ็พพลิเคชันสำหรับการออปติไมซ์ I/O

ข้อควรระวังและผลประโยชน์

มีข้อได้เปรียบต่างๆ ในการใช้ MIO แต่ไลบรารีนี้ ควรนำมาใช้ด้วยความระมัดระวัง

ผลประโยชน์

- MIO เนื่องจากมีความง่ายต่อการนำไปปฏิบัติ และง่ายต่อการนำไปวิเคราะห์ I/O ของแอ็พพลิเคชัน
- MIO อนุญาตให้แคช I/O ที่ระดับของแอ็พพลิเคชัน: คุณสามารถออปติไมซ์การเรียกของระบบ I/O ดังนั้น ระบบจะอินเตอร์รัปต์
- แคช pf สามารถปรับแต่งได้สำหรับไฟล์แต่ละไฟล์ หรือสำหรับกลุ่มของไฟล์ ซึ่งสามารถปรับแต่งได้ง่ายกว่าแคช OS
- MIO สามารถนำมาใช้บนแอ็พพลิเคชัน I/O ที่รันอย่างพร้อมเพียงกัน การลิงก์ของแอ็พพลิเคชันด้วย MIO และการปรับแต่งแอ็พพลิเคชันให้ใช้แคช pf และ DIRECT I/O เพื่อส่งผ่านแคช JFS และ JFS2 ปกติ แอ็พพลิเคชันที่ลิงก์ MIO เหล่านี้จะปล่อยพื้นที่แคช OS เพิ่มเติมสำหรับแอ็พพลิเคชัน I/O ที่ไม่ได้ลิงก์กับ MIO
- แคช MIO จะมีประโยชน์สำหรับไฟล์การแอ็กเซสตามลำดับของขนาด

ข้อควรระวัง

- การใช้ในทางที่ผิดของคอนฟิกูเรชันแคชไลบรารี MIO สามารถเป็นสาเหตุทำให้ลดระดับ ผลการทำงานลง หากต้องการหลีกเลี่ยงวิธีนี้ อันดับแรกให้วิเคราะห์นโยบาย I/O ของแอ็พพลิเคชันของคุณ จากนั้น ค้นหาพารามิเตอร์อ็อพชันโมดูลที่ใช้กับสถานการณ์ของคุณ และตั้งค่าพารามิเตอร์เหล่านั้น เพื่อช่วยปรับปรุงผลการทำงาน ของแอ็พพลิเคชันของคุณ ตัวอย่างของการใช้ MIO ในทางที่ผิด:
  - สำหรับแอ็พพลิเคชันที่เข้าถึงไฟล์ที่เล็กกว่าขนาดหน่วยความจำ OS หากคุณปรับแต่งอ็อพชัน **direct** ของโมดูล **pf** คุณสามารถลดระดับผลการทำงานของคุณได้
  - สำหรับแร็นดอมแอ็กเซส แคชอาจลดระดับผลการทำงานของคุณลง
- แคช MIO จะถูกจัดสรรด้วยระบบย่อย malloc ในพื้นที่แอดเดรสของแอ็พพลิเคชัน ดังนั้น จึงควรระมัดระวังเนื่องจาก หากขนาดของแคช MIO ทั้งหมดใหญ่กว่าหน่วยความจำ OS ที่มีอยู่ ระบบจะใช้พื้นที่การเพจ ซึ่งสามารถเป็นสาเหตุทำให้เกิดการลดระดับผลการทำงาน หรือระบบปฏิบัติการล้มเหลว

สถาปัตยกรรมแบบ MIO

ไลบรารีมอดูลาร์ I/O ประกอบด้วยโมดูล I/O ทั้งหมด 5 โมดูล ซึ่งอาจเรียกใช้งานที่รันไทม์ตามไฟล์พื้นฐาน

โมดูลที่พร้อมใช้งานในปัจจุบันคือ:

- โมดูล *mio* ซึ่งเป็นอินเตอร์เฟสกับโปรแกรมผู้ใช้
- โมดูล *pf* ซึ่งเป็นโมดูลการดึงข้อมูลล่วงหน้า
- โมดูล *trace* ซึ่งเป็นโมดูลสำหรับการเก็บข้อมูลสถิติ
- โมดูล *recov* ซึ่งเป็นโมดูลที่ใช้ในการวิเคราะห์การประเมินผล I/O ที่ล้มเหลวและลองพยายามอีกครั้งในกรณีที่เกิดความล้มเหลว
- โมดูล *aix* ซึ่งเป็นอินเตอร์เฟส MIO กับระบบปฏิบัติการ

ดีฟอลต์โมดูลคือ *mio* และ *aix* ส่วนโมดูลอื่นๆ เป็นโมดูลเผื่อเลือก

การใช้ประโยชน์จาก I/O และโมดูล pf

โมดูล *pf* คือแคชพื้นที่ของผู้ใช้ที่ใช้กลไก LRU (Last Recently Used) แบบง่ายสำหรับการครอบครองเพจก่อน โมดูล *pf* ยังมอนิเตอร์การใช้เพจของแคชเพื่อคาดการณ์ถึงความต้องการข้อมูลไฟล์ในอนาคต การออกคำสั่ง **aio_read** เพื่อโหลดแคชด้วยข้อมูลล่วงหน้า

รูปแบบ I/O ทั่วไปคือลำดับของการอ่านของไฟล์ที่มีขนาดใหญ่มาก (สิบกิกะไบต์) แอ็พพลิเคชันที่แสดงรูปแบบ I/O นี้มีแนวโน้มที่ได้รับประโยชน์ อย่างน้อยที่สุดจากแคชบัฟเฟอร์ของระบบปฏิบัติการ พูลบัฟเฟอร์ของระบบปฏิบัติการที่มีขนาดใหญ่จะไม่มีประสิทธิภาพ เนื่องจากมีการนำข้อมูลกลับมาใช้น้อยมาก ไลบรารี MIO สามารถนำมาใช้เพื่อแสดงปัญหานี้ได้โดยเรียกใช้โมดูล *pf* ซึ่งจะตรวจพบรูปแบบการแอ็กเซสเข้าถึงตามลำดับ และการโหลดล่วงหน้าแบบอะซิงโครนัสที่เล็กกว่าพื้นที่แคชที่มีข้อมูลที่ต้องการ แคช *pf* ต้องการพื้นที่ที่มีขนาดใหญ่เพียงพอ เพื่อรักษาการอ่านก่อนหน้าไว้อย่างเพียงพอ (การดึงข้อมูลล่วงหน้า) โมดูล *pf* สามารถนำมาใช้กับ I/O ได้โดยตรงซึ่งจะหลีกเลี่ยงการทำสำเนาหน่วยความจำพิเศษ ลงในพูลบัฟเฟอร์ของระบบ และยังล้างข้อมูลบัฟเฟอร์ระบบจากการเข้าถึงแบบหนึ่งครั้งของทราฟิก I/O ซึ่งอนุญาตให้บัฟเฟอร์ของระบบถูกนำมาใช้งาน ได้อย่างมีประสิทธิภาพมากขึ้น จากประสบการณ์ก่อนหน้าที่มีต่อระบบไฟล์ JFS และ JFS2 ของ AIX จะมีการสาธิตการใช้ I/O โดยตรงจากโมดูล *pf* ซึ่งจะมีประโยชน์อย่างมากต่อทรูพุตของระบบสำหรับไฟล์ที่ประเมินผลไว้ตามลำดับของขนาด

การนำไป MIO ไปปฏิบัติ

มีสามเมธอดที่พร้อมใช้งานเพื่อนำ MIO ไปปฏิบัติ คือ: การลิงก์เพื่อเปลี่ยนทิศทาง **libtkio** การเปลี่ยนทิศทางที่ประกอบด้วย **libmio.h** และการเรียกรูทีน MIO ที่ชัดเจน

การนำไปปฏิบัติเป็นไปได้ง่ายโดยใช้หนึ่งในสามเมธอดเหล่านี้ อย่างไรก็ตาม การลิงก์เพื่อเปลี่ยนทิศทาง **libtkio** จะเป็นเมธอดที่แนะนำให้ใช้

การลิงก์การกลับทิศทางกับไลบรารี tkio

ไลบรารี Trap Kernel I/O (**tkio**) คือไลบรารีเพิ่มเติม ที่จัดส่งมาพร้อมกับแพ็กเกจ **libmio** ซึ่งช่วยให้การใช้ประโยชน์ MIO สามารถดำเนินการได้ง่ายในแอ็พพลิเคชัน

เพื่อดำเนินการ MIO คุณสามารถใช้ตัวแปรสภาพแวดล้อม *TKIO_ALTLIB* เพื่อตั้งค่าคอนฟิกเคอร์นัล I/O พื้นฐานเพื่อให้การเรียกสามารถเปลี่ยนรูทีนได้:`setenv TKIO_ALTLIB "libmio.a(get_mio_ptrs.so)"`

การตั้งค่าตัวแปรสภาพแวดล้อม *TKIO_ALTLIB* ในลักษณะนี้จะแทนที่อ็อบเจ็กต์แบบแบ่งใช้ดีฟอลต์ด้วยอ็อบเจ็กต์แบบแบ่งใช้ MIO (*get_mio_ptrs.so*) ดังนั้นจึงส่งคืนโครงสร้างที่เต็มไปด้วยตัวชี้ไปยัง MIO I/O รูทีนทั้งหมด โหลดมีการดำเนินการหนึ่งครั้งสำหรับการเรียกระบบครั้งแรก จากนั้นการเรียกระบบ I/O ทั้งหมดของแอ็พพลิเคชัน ซึ่งเป็น entry points สำหรับ libtkio จะถูกกลับทิศทางไปยัง MIO

นี่เป็นวิธีการที่แนะนำสำหรับการดำเนินงาน MIO เนื่องจากถ้าโหลดล้มเหลวหรือการเรียกฟังก์ชันล้มเหลว **libtkio** จะแปลงเป็นโครงสร้างดีฟอลต์ของตัวชี้ที่เป็นการเรียกระบบปกติ นอกจากนี้ ถ้ามีปัญหาเกี่ยวกับแอ็พพลิเคชันที่ใช้ MIO คุณสามารถลบ MIO ออกจากการรันได้ง่ายโดยไม่ต้องตั้งค่า *TKIO_ALTLIB*

การกลับทิศทางด้วย libmio.h

วิธีการดำเนินการ MIO นี้เกี่ยวข้องกับการเพิ่มบรรทัดสองบรรทัดลงใน ซอร์สโค้ดของแอ็พพลิเคชันของคุณ

คุณสามารถดำเนินการ MIO โดยการเรียกใช้แมโคร C *USE_MIO_DEFINES* ซึ่งกำหนดชุดของแมโครในไฟล์ส่วนหัว **libmio.h** ซึ่งจะกลับทิศทางการเรียก I/O ไปยังไลบรารี MIO โดยสากล ไฟล์ส่วนหัว **libmio.h** มีการจัดส่ง พร้อมกับแพ็กเกจ **libmio** และมีข้อความ **#define** ต่อไปนี้:

```
#define open64(a,b,c) MIO_open64(a,b,c,0)
#define close MIO_close
#define lseek64 MIO_lseek64
#define ftruncate64 MIO_ftruncate64
#define fstat64 MIO_fstat64
#define fcntl MIO_fcntl
#define ffinfo MIO_ffinfo
#define fsync MIO_fsync
#define read MIO_read
#define write MIO_write
#define aio_read64 MIO_aio_read64
#define aio_write64 MIO_aio_write64
#define aio_suspend64 MIO_aio_suspend64
#define lio_listio MIO_lio_listio
```

1. เมื่อต้องการดำเนินการ MIO โดยใช้วิธีการนี้ ให้เพิ่มบรรทัดสองบรรทัด ลงในซอร์สโค้ดของคุณ:

   ```
   #define USE_MIO_DEFINES
   #include "libmio.h"
   ```

2. รีคอมไพล์แอ็พพลิเคชัน

การเรียกรูทีน MIO แบบชัดเจน

MIO สามารถนำมาปฏิบัติได้โดยทำการเรียกรูทีน MIO แบบชัดเจน

แทนที่จะใช้ไฟล์ส่วนหัว `libmio.h` และคำสั่ง **#define** เพื่อเปลี่ยนทิศทาง I/O ที่เรียกไลบรารี MIO คุณสามารถเพิ่มคำสั่ง **#define** ให้กับซอร์สโค้ดของแอ็พพลิเคชันของคุณได้โดยตรง จากนั้นคอมไพล์ซอร์สโค้ดอีกครั้ง

ตัวแปรสภาวะแวดล้อม MIO

ตัวแปรสภาวะแวดล้อมสี่ตัวที่พร้อมใช้งานสำหรับการปรับแต่ง MIO

MIO_STATS

ใช้ *MIO_STATS* เพื่อชี้ไปยังไฟล์บันทึกการทำงาน สำหรับการวินิจฉัยข้อความและสำหรับเอาต์พุตที่ร้องขอจากโมดูล MIO

ซึ่งตีความได้เป็นชื่อไฟล์ที่มีกรณีพิเศษอยู่ 2 กรณี ถ้าไฟล์นั้นคือ stderr หรือ stdout อย่างใดอย่างหนึ่ง เอาต์พุตจะถูกออกคำสั่งไปยังสตรีมไฟล์ ที่เหมาะสม ถ้าชื่อไฟล์ขึ้นต้นด้วยเครื่องหมายบวก (+) เช่น *+filename.txt* แล้ว ไฟล์จะถูกเปิดไว้สำหรับการผนวก ถ้าไม่มีเครื่องหมายบวกนำหน้าชื่อไฟล์แล้ว ไฟล์จะถูกเขียนทับ

MIO_FILES

MIO_FILES จัดเตรียมคีย์เพื่อ กำหนดโมดูลที่จะเรียกสำหรับไฟล์ที่กำหนด เมื่อเรียก **MIO_open64**

รูปแบบสำหรับ MIO_FILES คือ:

```
first_file_name_list [ module list ] second_file_name_list [ module list] ...
```

เมื่อ **MIO_open64** ถูกเรียกการตรวจสอบ MIO สำหรับการมีอยู่ของตัวแปรสภาวะแวดล้อม *MIO_FILES* ถ้าตัวแปรสภาวะแวดล้อมคือการแสดง MIO วิเคราะห์ข้อมูลเพื่อพิจารณาถึงโมดูลที่เรียกใช้งานสำหรับไฟล์

MIO_FILES ถูกวิเคราะห์คำจากซ้ายมาขวา ส่วนนำหน้าอักขระทั้งหมด ด้วยเครื่องหมายวงเล็บซ้าย ([) จะถูกใช้เป็น *file_name_list file_name_list* คือรายการของรูปแบบ *file_name_template* ที่คั่นด้วยเครื่องหมายโคลอน (:) รูปแบบ *File_name_template* ใช้เพื่อจับคู่ชื่อของไฟล์ที่กำลังถูกเปิดโดย MIO และอาจ ใช้อักขระ wildcard ต่อไปนี้:

- เครื่องหมายดอกจัน (*) จับคู่กับศูนย์หรืออักขระอื่นๆ ของชื่อไดเร็กทอรี หรือชื่อไฟล์
- เครื่องหมายคำถาม (?) จับคู่กับอักขระหนึ่งตัวของชื่อไดเร็กทอรีหรือชื่อไฟล์
- เครื่องหมายดอกจันสองตัว (**) จับคู่กับอักขระที่เหลืออยู่ทั้งหมดของชื่อพาธเต็ม

ถ้ารูปแบบ *file_name_template* ไม่มีเครื่องหมายฟอร์เวิร์ดสแลช (/) ข้อมูลไดเร็กทอรีพาธทั้งหมดในชื่อไฟล์ที่ส่งไปยังรูทีนย่อย **MIO_open64** จะถูกละเว้น และการจับคู่ถูกใช้กับชื่อลีฟของไฟล์ ที่ถูกเปิดเท่านั้น

ถ้าชื่อของไฟล์ที่แสดงอยู่ใน *file_name_list* ตรงกับหนึ่งในรูปแบบ *file_name_template* ใน *file_name_list* รายการโมดูลที่บ่งชี้ในเครื่องหมายวงเล็บต่อจาก *file_name_list* จะถูกเรียก หากชื่อของไฟล์ที่แสดงอยู่ใน *file_name_list* ไม่ตรงกับแพตเทิร์น *file_name_template* ใดๆ ใน *file_name_list* แรก ตัววิเคราะห์จะย้ายไปยัง *file_name_list* และพยายาม จับคู่ชื่อของไฟล์ที่นั่น ถ้าชื่อไฟล์ตรงกับรูปแบบ *file_name_template* สองรายการขึ้นไป ระบบจะพิจารณา รูปแบบแรก ถ้าชื่อของไฟล์ที่เปิดไม่ตรงกับแพตเทิร์น *file_name_template* ใดๆ ใน *file_name_lists* ไฟล์จะถูกเปิดด้วยการเรียกทำงานตามค่าดีฟอลต์ ของโมดูล **aix** ถ้ามีการจับคู่กัน โมดูลจากโมดูลที่เชื่อมโยงจะถูกแสดงในตัวแปรสภาวะแวดล้อม *MIO_FILES* โมดูลถูกเรียกในลำดับจากซ้ายไปขวา ด้วย โมดูลที่อยู่ทางซ้ายสุดของโปรแกรมผู้ใช้ และโมดูลที่อยู่ทางขวาสุด คือโมดูลที่ใกล้กับระบบปฏิบัติการมากที่สุด ถ้ารายการโมดูล ไม่ได้เริ่มต้นด้วยโมดูล **mio** ค่าติดตั้งดีฟอลต์ ของโมดูล **mio** จะเพิ่มคำเติมหน้าลงในตัวแปร สภาวะแวดล้อม ถ้าไม่ได้ระบุโมดูล **aix** ค่าติดตั้ง ดีฟอลต์ของโมดูล **aix** จะมีการผนวกต่อท้ายตัวแปร สภาวะแวดล้อม

ต่อไปนี้คือตัวอย่างแบบง่ายๆ ของวิธีการที่ *MIO_FILES* ถูกจัดการ:

```
MIO_FILES= *.dat:*.scr [ trace ] *.f01:*.f02:*.f03 [ trace | pf | trace ]
```

รูทีนย่อย **MIO_open64** จะเปิดไฟล์ `test.dat` และจับคู่ชื่อกับรูปแบบ ***.dat** *file_name_template* ซึ่งส่งผลทำให้เกิดการเรียกใช้โมดูล **mio, trace** และ **aix**

รูทีนย่อย **MIO_open64** จะเปิดไฟล์ `test.f02` และจับคู่ชื่อกับ ***.f02** รูปแบบ *file_name_template* ที่สองใน *file_name_list* ที่สอง ส่งผลทำให้เกิดเรียกใช้โมดูล **mio, trace, pf, trace** และ **aix**

แต่ละโมดูลมีอ็อพชันดีฟอลต์ฮาร์ดโค้ดของตนเอง สำหรับการเรียกตัวแปรสภาวะแวดล้อมโดยดีฟอลต์ คุณสามารถลบล้างอ็อพชันดีฟอลต์โดยระบุค่ารายการโมดูล *MIO_FILES* ที่ เชื่อมโยง ตัวอย่างโค้ดต่อไปนี้เปิดสถิติสำหรับโมดูล **trace** และเปลี่ยนทิศทางเอาต์พุตไปยังไฟล์ `my.stats`:

```
MIO_FILES= *.dat : *.scr [ trace/stats=my.stats ]
```

อ็อพชัน สำหรับโมดูลจะคั่นด้วยเครื่องหมายสแลชนำหน้า (/) อ็อพชันบางรายการ ต้องการค่าเลขจำนวนเต็มที่เชื่อมโยง หรือค่าสตริง สำหรับ อ็อพชันที่ต้องการค่าสตริง ถ้าสตริงมีเครื่องหมายฟอร์เวิร์ดสแลช (/) ให้ใส่สตริงไว้ในวงเล็บปีกกา {} สำหรับอ็อพชันที่ต้องการ ค่าเลขจำนวนเต็ม คุณอาจผนวกต่อท้ายค่าเลขจำนวนเต็มด้วย k, m, g หรือ t เพื่อแสดงถึงกิโลไบต์ เมกะไบต์ กิกะไบต์ หรือเทระไบต์ ค่าเลขจำนวนเต็มยังสามารถป้อนในฐาน 10, 8 หรือ 16 ถ้าค่าเลขจำนวนเต็ม ใช้คำเติมหน้า 0x เลขจำนวนเต็มจะถูกตีความเป็นฐาน 16 ถ้าค่าเลขจำนวนเต็มใช้คำเติมหน้า 0 เลขจำนวนเต็มจะถูกตีความ เป็นฐาน 8 ถ้าสองการทดสอบเหล่านี้ล้มเหลว เลขจำนวนเต็มจะถูกตีความเป็น ฐาน 10

MIO_DEFAULTS

วัตถุประสงค์ของตัวแปรสภาวะแวดล้อม *MIO_DEFAULTS* คือ ความสามารถในการอ่านข้อมูลที่เก็บอยู่ในตัวแปรสภาวะแวดล้อม *MIO_FILES*

ถ้าผู้ใช้ระบุโมดูลทั้งหมดสำหรับ *file_name_list* จำนวนมาก และโมดูลที่แสดงเป็นคู่แล้ว ตัวแปรสภาวะแวดล้อม *MIO_FILES* อาจค่อนข้างยาว ถ้าผู้ใช้แทนที่ดีฟอลต์โค้ดที่เขียนด้วยมือในวิธีที่เหมือนกันแบบซ้ำๆ วิธีการนี้เป็นวิธีการแบบง่ายเพื่อระบุค่าดีฟอลต์ใหม่สำหรับโมดูล โดยใช้ตัวแปรสภาวะแวดล้อม *MIO_DEFAULTS* ตัวแปรสภาวะแวดล้อมนี้จะคั่นด้วยเครื่องหมายจุลภาคสำหรับรายการของโมดูล ด้วยค่าดีฟอลต์ใหม่ตามที่อยู่ในตัวอย่างต่อไปนี้:

```
MIO_DEFAULTS = trace/events=prob.events , aix/debug
```

ในเวลานี้ การเรียกใช้ค่าดีฟอลต์ใดๆ ของโมดูล **trace** จะมีการติดตามเหตุการณ์แบบไบนารีที่เปิดใช้งาน และออกคำสั่งไปยังไฟล์ `prob.events` และการเรียกใช้ค่าดีฟอลต์ใดๆ ของโมดูล **aix** จะมีอ็อพชัน **debug** ที่เปิดใช้งาน

MIO_DEBUG

วัตถุประสงค์ของตัวแปรสภาวะแวดล้อม *MIO_DEBUG* คือ การดีบัก MIO

MIO จะค้นหา *MIO_DEFAULTS* สำหรับคีย์เวิร์ด และจัดเตรียมเอาต์พุตการดีบักสำหรับอ็อพชัน คีย์เวิร์ดที่พร้อมใช้งานคือ:

ALL เปิดใช้งานคีย์เวิร์ดตัวแปรสภาวะแวดล้อม *MIO_DEBUG* ทั้งหมด

ENV เอาต์พุตตัวแปรสภาวะแวดล้อมที่ตรงกับคำร้องขอ

OPEN เอาต์พุตคำร้องขอที่สร้างขึ้นให้กับรูทีนย่อย **MIO_open64**

MODULES
เอาต์พุตโมดูลที่เรียกใช้งานสำหรับการเรียกรูทีนย่อย **MIO_open64** แต่ละครั้ง

TIMESTAMP
วางลงในไฟล์ stats สำหรับการประทับเวลาที่มาก่อนรายการแต่ละรายการ

DEF เอาต์พุตตารางนิยามของโมดูลแต่ละโมดูล ดัมพ์นี้จะถูกเรียกใช้งานสำหรับโมดูลไลบรารี MIO ทั้งหมดเมื่อเปิดไฟล์

นิยามอ็อพชันของโมดูล

โมดูล MIO แต่ละโมดูลมีอ็อพชันที่หลากหลายซึ่งพร้อมใช้งานเพื่อช่วยวิเคราะห์ และออปติไมซ์ผลการทำงานในระดับของแอ็พพลิเคชัน

นิยามอ็อพชันโมดูล MIO

โมดูล **mio** คืออินเตอร์เฟสกับโปรแกรมผู้ใช้ MIO และจะเรียกใช้งานที่รันไทม์ตามค่าดีฟอลต์

mode แทนที่โหมดการเข้าถึงไฟล์เพื่อเปิด

โหมดนี้จะถูกกำหนดเป็นพารามิเตอร์ในการเรียกของระบบเปิด AIX : โหมดเริ่มต้นจะถูกกำหนดในซอร์สโค้ดกับการเรียกของระบบเปิด AIX ซึ่งจะถูกแทนที่ด้วยโหมดนี้

nomode
ไม่ได้แทนที่โหมด นี่คืออ็อพชันดีฟอลต์

direct ตั้งค่าบิต **O_DIRECT** ในแฟล็กเปิด

nodirect
ล้างข้อมูลบิต **O_DIRECT** ในแฟล็กเปิด

osync ตั้งค่าบิต **O_SYNC** ในแฟล็กเปิด

noosync
ล้างข้อมูลบิต **O_SYNC** ในแฟล็กเปิด

คำนิยามอ็อพชันโมดูล TRACE

โมดูล **trace** คือโมดูลการรวบรวมข้อมูลสถิติสำหรับโปรแกรม ผู้ใช้ MIO และเป็นอ็อพชัน

stats{=output_file}
สถิติเอาต์พุตเมื่อปิด: ชื่อไฟล์สำหรับการวิเคราะห์เอาต์พุตการติดตาม

ถ้าไม่ได้ระบุ *output_file* หรือถ้าเป็น **mioout** ซึ่งเป็นค่าดีฟอลต์ โมดูล **trace** จะค้นหาไฟล์สถิติเอาต์พุต ที่กำหนดไว้ในตัวแปรสภาพแวดล้อม *MIO_STATS*

nostats ไม่ส่งเอาต์พุตสถิติ

events{=event_file}
สร้างไฟล์เหตุการณ์ไบนารี ค่าดีฟอลต์: `trace.events`

noevents
ไม่สร้างไฟล์เหตุการณ์ไบนารี นี่เป็นดีฟอลต์อ็อพชัน

ไบต์ สถิติเอาต์พุตในหน่วยไบต์ นี่เป็นขนาดหน่วยดีฟอลต์

kbytes สถิติเอาต์พุตในหน่วยกิโลไบต์

gbytes สถิติเอาต์พุตในหน่วยกิกะไบต์

tbytes สถิติเอาต์พุตในหน่วยเทระไบต์

inter ส่งเอาต์พุตสถิติระหว่างกาล

nointer ไม่ส่งเอาต์พุตสถิติระหว่างกาล นี่เป็นดีฟอลต์อ็อพชัน

คำนิยามอ็อพชันโมดูล PF

โมดูล **pf** คือโมดูล prefetching ข้อมูลสำหรับโปรแกรม ผู้ใช้ MIO และเป็นอ็อพชัน

pffw Prefetch หน้าแม้เมื่ออยู่ในโหมดบันทึก

nopffw ไม่ prefetch หน้าเมื่ออยู่ในโหมดบันทึก นี่เป็นดีฟอลต์อ็อพชัน

พฤติกรรมดีฟอลต์ของแคช **pf** คือการไม่อ่านหน้าล่วงหน้า เข้าในแคช ถ้าการเรียกที่ทริกเกอร์การอ่านล่วงหน้ามาจากผู้ใช้ ที่บันทึกเข้าในแคช ที่เป็นเช่นนี้เนื่องจากไม่มีประโยชน์ที่จะอ่านหน้าซึ่งจะ ถูกบันทึกทับ แต่ถ้าการบันทึกเข้าในหน้าในลำดับต่อมาของผู้ใช้ มีรูปแบบไม่ถูกต้อง (ไม่ได้เริ่มต้นและสิ้นสุดในขอบเขตส่วน) ตรรกะที่ทำเครื่อง หมายส่วนที่เสียจะไม่ถูกต้องอีกต่อไป ดังนั้นหน้าที่เสียจะต้องถูกบันทึกลงในดิสก์ แบบซิงโครนัส จากนั้นจะมีการอ่านหน้าทั้งหน้าแบบซิงโครนัสเข้าใน แคช ในกรณีนี้ เป็นการดีกว่าที่จะอ่านหน้าที่จะถูกบันทึกทับแบบอะซิงโครนัสเท่านั้น แทนที่จะดำเนินการ synchronous hit บน page miss

รีลีส ปล่อยหน้าแคชสากลให้เป็นอิสระเมื่อจำนวนการใช้ไฟล์แคชสากล เป็นศูนย์ นี่เป็นดีฟอลต์อ็อพชัน

norelease
ไม่ปล่อยหน้าแคชสากลให้เป็นอิสระเมื่อจำนวนการใช้ไฟล์แคชสากล เป็นศูนย์

อ็อพชัน **รีลีส** และ **norelease** ควบคุมสิ่งที่จะเกิดขึ้น กับแคชสากลเมื่อจำนวนการใช้ไฟล์เป็นศูนย์ พฤติกรรม ดีฟอลต์คือการปิดและรีลีสแคชสากล ถ้าแคชสากลถูกเปิด และปิดหลายครั้ง อาจมีปัญหาการแบ่งเฟรกเมนต์หน่วยความจำได้ ในบางเวลา การใช้อ็อพชัน **norelease** ช่วยเปิดแคชสากลค้างไว้ และมีอยู่แม้ว่าจำนวนการใช้ไฟล์เป็นศูนย์

ส่วนตัว ใช้แคชส่วนตัว นี่หมายความว่าเฉพาะไฟล์ที่เปิดแคชเท่านั้น สามารถใช้แคชนี้ได้

สากล ใช้แคชสากล นี่หมายความว่าหลายไฟล์สามารถใช้พื้นที่ว่างแคชเดียวกัน ได้ นี่เป็นดีฟอลต์อ็อพชัน

เราพบว่าบ่อยครั้งที่แคชสากลเป็นอ็อพชันที่ดีที่สุดด้วยเหตุผล หลายอย่าง เหตุผลหนึ่งคือสามารถกำหนดจำนวนหน่วยความจำได้ เนื่องจากคุณจะทราบ จำนวนแคชที่เปิดค้างไว้และขนาดของแคชเหล่านั้น เมื่อใช้แคชส่วนตัว ผู้ใช้อาจไม่ทราบจำนวนแคชส่วนตัวที่ใช้งานอยู่พร้อมกัน จำนวนสูงสุดของ แคชสากลที่สามารถเปิดได้คือ 256 ค่าดีฟอลต์คือใช้การตั้งค่าอ็อพชันสากลเป็นศูนย์ ซึ่งหมายความว่ามีแคชสากลที่เปิดค้างไว้หนึ่งแคช แต่ละแคชสากลที่เปิดค้างไว้ สามารถถูกกำหนดโดยผู้ใช้ให้กับกลุ่มเฉพาะของไฟล์

อะซิงโครนัส
ใช้การเรียกอะซิงโครนัสไปยังโมดูลชายน์ นี่เป็นดีฟอลต์อ็อพชัน

ซิงโครนัส
ใช้การเรียกซิงโครนัสไปยังโมดูลชายน์

noasynchronous
สมนามของซิงโครนัส

อ็อพชัน **อะซิงโครนัส, ซิงโครนัส,** และ **noasynchronous** ควบคุมว่าแคชควรจะใช้การเรียก I/O แบบอะซิงโครนัสไปยังข้อมูลโหลดแคช จากระบบไฟล์หรือไม่ บางครั้งใช้สำหรับการดีบักหรือเมื่อไม่ได้เปิดใช้งาน aio บนระบบ

โดยตรง
ใช้ DIRECT I/O

nodirect
ไม่ใช้ DIRECT I/O นี่เป็นดีฟอลต์อ็อพชัน

อ็อพชัน **โดยตรง** และ **nodirect** ควบคุมว่า O_DIRECT บิตคือ OR'd เข้าใน oflags ในการเปิดไฟล์หรือไม่ แคช **pf** สามารถทำ I/O โดยตรงได้ แคชจะจัดวางหน้าแคชทั้งหมดบนขอบเขต 4K และ พยายามออกใช้เฉพาะการร้องขอที่มีรูปแบบถูกต้อง ซึ่งต้องใช้เพื่อให้มั่นใจว่า การร้องขอ I/O เดินทางโดยตรง

ไบต์ สถิติเอาต์พุตแคชในหน่วยไบต์ นี่เป็นดีฟอลต์อ็อพชัน

kbytes สถิติเอาต์พุตแคชในหน่วยกิโลไบต์

mbytes สถิติเอาต์พุตแคชในหน่วยเมกะไบต์

gbytes สถิติเอาต์พุตแคชในหน่วยกิกะไบต์

tbytes สถิติเอาต์พุตแคชในหน่วยเทระไบต์

cache_size

ขนาดทั้งหมดในหน่วยไบต์ของแคช ขนาดในหน่วยกิโลไบต์ เมกะไบต์ กิกะไบต์ และเทระไบต์ยังมีการรับรู้ด้วย ค่าดีฟอลต์คือ 64k

page_size

ขนาดทั้งหมดในหน่วยไบต์ของแต่ละหน้าแคช ขนาดในหน่วยกิโลไบต์ เมกะไบต์ กิกะไบต์ และเทระไบต์ยังมีการรับรู้ด้วย ค่าดีฟอลต์คือ 4k

prefetch

จำนวนหน้าที่จะ prefetch ค่าดีฟอลต์คือ 1

stride ตั้งค่าปัจจัย stride ในหน่วยหน้า ค่าดีฟอลต์คือ 1

stats{=output_file}

สถิติการใช้เอาต์พุต prefetch: การวิเคราะห์ชื่อไฟล์สำหรับเอาต์พุต **pf**

ถ้าไม่ได้ระบุ *output_file* หรือถ้าเป็น **mioout** ซึ่งเป็นค่าดีฟอลต์ โมดูล **pf** จะค้นหาไฟล์สถิติเอาต์พุต ที่กำหนดไว้ในตัวแปรสภาพแวดล้อม *MIO_STATS*

nostats ไม่ส่งเอาต์พุตสถิติการใช้ prefetch

inter ส่งเอาต์พุตสถิติการใช้ prefetch ระหว่างกาลบนคำสั่ง **kill -SIGUSR1**

nointer ไม่ส่งเอาต์พุตสถิติการใช้ prefetch ระหว่างกาล นี่เป็นดีฟอลต์ อ็อพชัน

อ็อพชัน **inter** สร้างแคช **pf** เพื่อส่งเอาต์พุต สถิติการใช้เมื่อแอ็พพลิเคชันได้รับคำสั่ง **kill -30**

เก็บรักษา

เก็บรักษาข้อมูลไฟล์ไว้หลังจากปิดเพื่อเปิดใหม่

notain ไม่เก็บรักษาข้อมูลไฟล์หลังจากปิดเพื่อเปิดใหม่ นี่เป็นดีฟอลต์ อ็อพชัน

อ็อพชันเก็บรักษาจะสร้างแคช **pf** สากลเพื่อบันทึกหน้าของไฟล์ ที่จะใช้ใหม่ไว้ในแคช ถ้าเปิดไฟล์นั้นใหม่ในแคชสากล เดียวกัน ไฟล์ที่ดำเนินการต้องไม่มีการแก้ไขระหว่างช่วงเวลา ที่ปิดและเปิด หน้าในแคชยังคงต้องใช้ LRU แม้ว่าไฟล์ยังคงเปิดอยู่ในแคชก็ตาม

listio ใช้กลไก listio

nolistio ไม่ใช้กลไก listio นี่เป็นดีฟอลต์อ็อพชัน

โดยปกติ แคชไม่ใช้ listio ซึ่งใช้สำหรับการดีบัก เป็นหลัก

tag={tag string}

สตริงที่จะเสริมหน้าโฟลว์สถิติ

notag ไม่ใช้การเสริมหน้าโฟลว์สถิติ นี่เป็นดีฟอลต์อ็อพชัน

แท็กสตริงคือคำเสริมหน้าสำหรับเอาต์พุตที่พิมพ์ในไฟล์สถิติ เมื่อไฟล์สถิติ มีขนาดใหญ่ขึ้น แท็กสตริงจะทำให้ค้นหาส่วนที่ต้องการได้ ง่ายขึ้น

scratch ไฟล์เป็น scratch และจะถูกลบออกเมื่อปิด

noscratch

ไฟล์จะถูกฟลัชและบันทึกเมื่อปิด นี่เป็นดีฟอลต์อ็อพชัน

สร้างแคชเพื่อจัดการไฟล์เป็น scratch ซึ่งหมายความว่าไฟล์จะไม่ถูก ฟลัชเมื่อปิดและไม่เชื่อมโยงหลังจากการปิด

passthru

ช่วงไบต์ที่จะส่งผ่านแคช

ผู้ใช้สามารถระบุช่วงของไฟล์ที่จะไม่แคชได้ ข้อมูลมีการอ่าน และบันทึกผ่านทางแคช ลักษณะนี้เป็นสิ่งจำเป็นในอดีต สำหรับแอ็พพลิเคชันแบบขนานที่บันทึกลงในไฟล์เดียวกันและแบ่งใช้ข้อมูลส่วนหัว ที่ตอนต้นของไฟล์

คำนิยามอ็อพชันโมดูล RECOV

โมดูล **recov** วิเคราะห์การเข้าถึง I/O ที่ล้มเหลว และลองใหม่ในกรณีของความล้มเหลว นี่เป็นโมดูล MIO ที่เป็นอ็อพชัน

fullwrite

คาดหวังว่าการบันทึกทั้งหมดจะเป็นแบบเต็ม ถ้ามีความล้มเหลวในการบันทึกเนื่องจาก พื้นที่ว่างไม่เพียงพอ โมดูลจะลองใหม่ นี่เป็นดีฟอลต์อ็อพชัน

partialwrite

ไม่ได้คาดหวังว่าการบันทึกทั้งหมดจะเป็นแบบเต็ม ถ้ามีความล้มเหลวในการบันทึกเนื่องจาก พื้นที่ว่างไม่เพียงพอ จะไม่มีการลองใหม่

stats{=output_file}

เอาต์พุตสำหรับข้อความ recov

ถ้าไม่ได้ระบุ *output_file* หรือถ้าเป็น **mioout** ซึ่งเป็นค่าดีฟอลต์ โมดูล **recov** จะค้นหาไฟล์สถิติเอาต์พุต ที่กำหนดไว้ในตัวแปรสภาพแวดล้อม *MIO_STATS*

nostats ไม่มีสตรีมไฟล์เอาต์พุตสำหรับข้อความ recov

command

คำสั่งที่จะออกใช้บนข้อผิดพลาดการบันทึก ค่าดีฟอลต์คือ `command={ls -l}`

open_command

คำสั่งที่จะออกใช้บนข้อผิดพลาดการเปิดซึ่งเป็นผลมาจากการเชื่อมต่อที่ถูกปฏิเสธ ค่าดีฟอลต์คือ `open_command={echo connection refused}`

retry จำนวนครั้งที่จะลองใหม่ ค่าดีฟอลต์คือ 1

นิยามอ็อพชันโมดูล AIX

โมดูล **aix** คืออินเตอร์เฟส MIO กับระบบปฏิบัติการ และจะเรียกใช้งาน ณ รันไทม์ตามค่าดีฟอลต์

debug พิมพ์คำสั่งดีบักสำหรับเปิดและปิด

nodebug
ไม่พิมพ์คำสั่งดีบักสำหรับเปิดและปิด นี่คือ ค่าดีฟอลต์

sector_size
ระบุขนาดของเซ็กเตอร์ ถ้าไม่ระบุ ให้ตั้งค่าขนาดเซ็กเตอร์ให้เท่ากับขนาดของ ระบบไฟล์

notrunc
ไม่ต้องออกคำสั่งสำหรับการเรียกของระบบ trunc นี่เป็นสิ่งจำเป็นเพื่อหลีกเลี่ยงปัญหาเกี่ยวกับข้อผิดพลาด JFS O_DIRECT

trunc ออกคำสั่งการเรียกของระบบ trunc นี่คืออ็อพชันดีฟอลต์

ตัวอย่างการใช้ MIO

มีหลายสถานการณ์จำลองที่เกี่ยวข้องกับไลบรารี MIO

ตัวอย่างของการดำเนินการ MIO โดยการลิงก์ไปยัง libtkio

MIO สามารถดำเนินการได้โดยการลิงก์ไปยัง libtkio เพื่อกลับทิศทางการ เรียก I/O ไปยังไลบรารี MIO

สคริปต์นี้ตั้งค่าตัวแปรสภาพแวดล้อม MIO ลิงก์แอ็พพลิเคชันกับ ไลบรารี Trap Kernel I/O (**tkio**) และเรียก MIO

```
#!/bin/csh
#
setenv TKIO_ALTLIB "libmio.a(get_mio_ptrs.so)"

setenv MIO_STATS example.stats
setenv MIO_FILES " *.dat [ trace/stats ] "
setenv MIO_DEFAULTS " trace/kbytes "
setenv MIO_DEBUG OPEN
#
cc -o example example.c  -ltkio

#
./example file.dat
```

ตัวอย่างของการดำเนินการ MIO โดยการรวมไฟล์ส่วนหัว libmio.h

MIO สามารถดำเนินการได้โดยการเพิ่มไฟล์ส่วนหัว `libmio.h` ลงในซอร์สไฟล์ของแอ็พพลิเคชันเพื่อกลับทิศทางการเรียก I/O ไปยังไลบรารี MIO

สองบรรทัดต่อไปนี้จะมีการเพิ่มลงในไฟล์ `example.c`:

```
#define USE_MIO_DEFINES
#include "libmio.h"
```

This script sets the MIO environment variables, compiles and links the application with the MIO library, and to calls it.

```
#!/bin/csh
#
setenv MIO_STATS example.stats
setenv MIO_FILES " *.dat [ trace/stats ] "
setenv MIO_DEFAULTS " trace/kbytes "
```

```
setenv MIO_DEBUG OPEN
#
cc -o example example.c -lmio
#
./example file.dat
```

ไฟล์เอาต์พุตการวินิจฉัย MIO

ข้อมูลการวินิจฉัยไลบรารี MIO จะถูกเขียนลงในไฟล์สถิติ เมื่อเรียกรูทีนย่อย **MIO_close**

ชื่อของไฟล์สถิติจะถูกกำหนดอยู่ในตัวแปรสภาวะแวดล้อม *MIO_STATS* ถ้าอ็อพชัน **stats** ตั้งค่าเป็นค่าดีฟอลต์ของ `mioout` ไฟล์สถิติสามารถมีข้อมูลต่อไปนี้:

- ข้อมูลการดีบัก
- การวินิจฉัยจากโมดูล **trace** ถ้าอ็อพชัน **stats** ของโมดูล **trace** ถูกตั้งค่าไว้

  หมายเหตุ: หากต้องการยกเลิกข้อมูลการวินิจฉัยนี้ ให้ใช้อ็อพชัน **nostats** ของโมดูล **trace** หรือแยกข้อมูลการวินิจฉัยโมดูล **trace** ออกจากโมดูลอื่น ใช้อ็อพชัน **stats{=output_file}** ของโมดูล **trace**
- การวินิจฉัยจากโมดูล **pf** หากอ็อพชัน **stats** ของโมดูล **pf** ถูกตั้งค่าไว้

  หมายเหตุ: หากต้องการยกเลิกการวินิจฉัยเหล่านี้ ให้ใช้อ็อพชัน **nostats** ของโมดูล **pf** หรือแยกข้อมูลการวินิจฉัยโมดูล **pf** ออกจากโมดูลอื่นๆ ใช้อ็อพชัน **stats{=output_file}** ของโมดูล **pf**
- การกู้คืนข้อมูลการติดตามหากอ็อพชัน **stats** ของโมดูล **recov** ถูกตั้งค่าไว้

  หมายเหตุ: หากต้องการแยกข้อมูลการวินิจฉัยโมดูล **recov** จากโมดูลอื่น ให้ใช้อ็อพชัน **stats{=output_file}** ของโมดูล **recov**

ตัวอย่างไฟล์การวิเคราะห์โมดูล trace:

ไฟล์ stat ของโมดูล **trace** ประกอบด้วยข้อมูลดีบัก และการวิเคราะห์

องค์ประกอบส่วนหัว

- วันที่
- ชื่อโฮสต์
- aio เปิดใช้งานหรือไม่
- ชื่อโปรแกรม
- MIO ไลบรารีเวอร์ชัน
- ตัวแปรสภาพแวดล้อม

องค์ประกอบดีบัก

- รายการของอ็อพชันดีบักค่าติดตั้งทั้งหมด
- ตารางคำนิยามโมดูลทั้งหมดถ้ามีการตั้งค่า DEF ดีบักอ็อพชัน
- การร้องขอการเปิดที่ทำใน MIO_open64 ถ้ามีการตั้งค่า OPEN ดีบัก
- โมดูลที่เรียกใช้ถ้ามีการตั้งค่า MODULES ดีบักอ็อพชัน

องค์ประกอบเฉพาะของโมดูลการติดตามที่มีโครงร่าง

- เวลา ถ้ามีการตั้งค่า **TIMESTAMP** ดีบักอ็อพชัน
- การติดตามการปิดหรือการขัดจังหวะระหว่างกาล
- การติดตามตำแหน่งโมดูลใน module_list
- ชื่อไฟล์ที่ประมวลผล
- อัตรา: จำนวนข้อมูลหารด้วยเวลาทั้งหมด; จำนวนเวลาสะสมที่ใช้ในโมดูลการติดตาม
- อัตราความต้องการ: จำนวนข้อมูลหารด้วยระยะเวลาที่เปิดไฟล์ รวมถึงเวลาที่เปิดและปิดไฟล์
- ขนาดไฟล์ปัจจุบัน (ขณะติดตาม) และขนาดไฟล์สูงสุดในระหว่าง การประมวลผลไฟล์นี้
- ข้อมูลระบบไฟล์: ชนิดไฟล์ ขนาดส่วน
- โหมดการเปิดไฟล์และแฟล็ก
- สำหรับแต่ละฟังก์ชัน: จำนวนครั้งที่เรียกฟังก์ชันนี้ และเวลาการประมวลผล สำหรับฟังก์ชันนี้
- สำหรับฟังก์ชันการอ่านหรือการบันทึก: ข้อมูลเพิ่มเติม เช่น ที่ร้องขอ (ขนาดของ การอ่านหรือการบันทึกที่ร้องขอ), ทั้งหมด (ขนาดของการอ่านหรือการบันทึกจริง: ส่งคืนโดยการเรียกระบบ aix), min, และ max
- สำหรับการค้นหา: delta การค้นหาเฉลี่ย (delta การค้นหาทั้งหมดและจำนวนการค้นหา)
- สำหรับการอ่านหรือการบันทึก: ข้อมูลที่พักไว้ เช่น จำนวน เวลาและอัตราของ เวลาโอนย้าย รวมถึงเวลาที่พักไว้ของการอ่านและการบันทึก
- จำนวนของ fcntl page_info requests : หน้า

```
date
Trace on close or intermediate  :  previous module or calling program <-> next module : file name : (total transferred bytes/total time)=rate
    demand rate=rate/s=total transferred bytes/(close time-open time)
    current size=actual size of the file  max_size=max size of the file
mode=file open mode FileSystemType=file system type given by fststat(stat_b.f_vfstype) sector size=Minimum direct i/o transfer size
oflags=file open flags
open    open count          open time
fcntl      fcntl count         fcntl time
read      read count         read time   requested size      total size      minimum      maximum
aread    aread count       aread time requested size      total size      minimum      maximum
   suspend  count            time           rate
write      write count      write time  requested size       total size      minimum      maximum
seek      seek count        seek time      average seek delta
size
page    fcntl page_info count
```

ตัวอย่าง

```
MIO statistics file : Tue May 10 14:14:08 2005
hostname=host1  : with Legacy aio available
Program=/mio/example
MIO library libmio.a 3.0.0.60  AIX 32 bit addressing built Apr 19 2005 15:08:17
MIO_INSTALL_PATH=
MIO_STATS       =example.stats
MIO_DEBUG       =OPEN
MIO_FILES       = *.dat [ trace/stats ]
MIO_DEFAULTS    = trace/kbytes

MIO_DEBUG OPEN =T

Opening file file.dat
   modules[11]=trace/stats
================================================================================
```

```
Trace close : program <-> aix : file.dat : (4800/0.04)=111538.02 kbytes/s
     demand rate=42280.91 kbytes/s=4800/(0.12-0.01))
     current size=0   max_size=1600
  mode =0640  FileSystemType=JFS sector size=4096
  oflags =0x302=RDWR  CREAT  TRUNC
  open              1     0.00
  write           100     0.02       1600       1600      16384      16384
  read            200     0.02       3200       3200      16384      16384
  seek            101     0.01 average seek delta=-48503
  fcntl             1     0.00
  trunc             1     0.01
  close             1     0.00
  size            100
================================================================================
```

ตัวอย่างไฟล์การวิเคราะห์โมดูล pf:

ไฟล์ stat ของโมดูล **pf** ประกอบด้วยข้อมูลดีบัก และการวิเคราะห์

องค์ประกอบและโครงร่าง

```
pf  close for <name of the file in the cache>
pf close for global or private cache <global cache number>
<number of page compute by cache_size/page-size> page of <page-size>   <sector_size> bytes per sector
<real number page not prefetch because of pffw option( suppress number of page prefetch because sector not valid)> /
                                                    <page not prefetch because of pffw option>  pages not preread for write
<number of unused prefetch>  unused prefetches out of  <number of started prefetch>  prefetch=<number of page to prefetch>
<number> of write behind
<number> of page syncs forced by ill formed writes
<number> of pages retained over close
<unit> transferred / Number of requests
program --> <bytes written into the cache by parent>/<number of write from parent> --> pf -->

                                                <bytes written out of the cache from the child>/<number of partial page written>
program <-- <bytes read out of the cache by parent>/<number of read from parent> <-- pf <--
                                                <bytes read in from child of the cache>/<number of page read from child>
```

ตัวอย่าง

```
pf close for /home/user1/pthread/258/SM20182_0.SCR300
50 pages of 2097152 bytes 131072 bytes per sector
133/133 pages not preread for write
23 unused prefetches out of 242 : prefetch=2
95 write behinds
mbytes transferred / Number of requests
program --> 257/257 --> pf --> 257/131 --> aix
program <-- 269/269 <-- pf <-- 265/133 <-- aix
```

ตัวอย่างไฟล์การวิเคราะห์โมดูล recov:

ไฟล์ stat ของโมดูล **recov** ประกอบด้วยข้อมูลดีบัก และการวิเคราะห์

ถ้ารูทีนการเปิดหรือการบันทึกล้มเหลว โมดูล **recov** จะเพิ่ม ข้อคิดเห็นในไฟล์เอาต์พุตที่มีองค์ประกอบดังต่อไปนี้:

- ค่า **open_command** หรือ **command** ของโมดูล ที่จะดำเนินการในกรณีของความล้มเหลว โปรดดู “คำนิยามอ็อพชันโมดูล RECOV” ในหน้า 248 สำหรับ ข้อมูลเพิ่มเติมเกี่ยวกับอ็อพชันเหล่านี้
- หมายเลขข้อผิดพลาด
- จำนวนของการพยายามลองใหม่

```
15:30:00
 recov : command=ls -l file=file.dat errno=28 try=0
 recov : failure : new_ret=-1
```

ตัวอย่างคอนฟิกูเรชัน MIO

คุณสามารถปรับแต่ง MIO ในระดับของแอ็พพลิเคชันได้

คอนฟิกูเรชัน OS

แอ็พพลิเคชัน **alot_buf** จะบรรลุเป้าหมายดังต่อไปนี้:

- เขียนไฟล์ขนาด 14 GB
- 140 000 ลำดับการเขียนด้วยบัฟเฟอร์ขนาด 100 KB
- อ่านไฟล์ตามลำดับด้วยบัฟเฟอร์ขนาด 100 KB
- อ่านไฟล์ย้อนกลับตามลำดับด้วยบัฟเฟอร์ขนาด 100 KB

```
# vmstat
 System Configuration: lcpu=2 mem=512MB
 kthr     memory             page              faults        cpu
 ----- ----------- ------------------------ ------------ -----------
  r  b   avm   fre  re  pi  po  fr   sr  cy  in   sy  cs us sy id wa
  1  1 35520 67055   0   0   0   0    0   0 241   64  80  0  0 99  0

# ulimit -a
 time(seconds)        unlimited
 file(blocks)          unlimited
 data(kbytes)         131072
 stack(kbytes)        32768
 memory(kbytes)       32768
 coredump(blocks)     2097151
 nofiles(descriptors) 2000


# df -k /mio
Filesystem    1024-blocks      Free %Used    Iused %Iused Mounted on
/dev/fslv02      15728640  15715508    1%      231     1% /mio

# lslv fslv02
LOGICAL VOLUME:     fslv02                 VOLUME GROUP:   mio_vg
LV IDENTIFIER:      000b998d00004c00000000f17e5f50dd.2 PERMISSION:     read/write
VG STATE:           active/complete        LV STATE:       opened/syncd
TYPE:               jfs2                   WRITE VERIFY:   off
MAX LPs:            512                    PP SIZE:        32 megabyte(s)
COPIES:             1                      SCHED POLICY:   parallel
LPs:                480                    PPs:            480
STALE PPs:          0                      BB POLICY:      relocatable
INTER-POLICY:       minimum                RELOCATABLE:    yes
INTRA-POLICY:       middle                 UPPER BOUND:    32
MOUNT POINT:        /mio                   LABEL:          /mio
MIRROR WRITE CONSISTENCY: on/ACTIVE
EACH LP COPY ON A SEPARATE PV ?: yes
Serialize IO ?:     NO
```

คอนฟิกูเรชัน MIO เพื่อวิเคราะห์แอ็พพลิเคชัน /mio/alot_buf

```
setenv MIO_DEBUG " OPEN MODULES TIMESTAMP"
setenv MIO_FILES "* [ trace/stats/kbytes ]"
setenv MIO_STATS mio_analyze.stats

time /mio/alot_buf
```

หมายเหตุ: ไฟล์วินิจฉัยเอาต์พุตคือ mio_analyze.stats สำหรับข้อมูล **ดีบัก** และข้อมูลโมดูล **การติดตาม** ค่าทั้งหมดจะอยู่ในหน่วยกิโลไบต์

หมายเหตุ: คำสั่ง **time** จะสั่งให้ MIO ติดประกาศเวลาของการประมวลผลคำสั่ง

ผลลัพธ์ของการวิเคราะห์

- เวลาของการประมวลผลคือ 28:06
- ไฟล์เอาต์พุตการวินิจฉัยการวิเคราะห์ MIO คือ mio_analyse.stats

```
MIO statistics file : Thu May 26 17:32:22 2005
hostname=miohost  : with Legacy aio available
Program=/mio/alot_buf
MIO library libmio.a 3.0.0.60  AIX 64 bit addressing built Apr 19 2005 15:07:35
MIO_INSTALL_PATH=
MIO_STATS       =mio_analyze.stats
MIO_DEBUG       = MATCH OPEN MODULES TIMESTAMP
MIO_FILES       =* [ trace/stats/kbytes ]
MIO_DEFAULTS    =

MIO_DEBUG OPEN =T
MIO_DEBUG MODULES =T
MIO_DEBUG TIMESTAMP =T

17:32:22
Opening file test.dat
   modules[18]=trace/stats/kbytes
   trace/stats={mioout}/noevents/kbytes/nointer
   aix/nodebug/trunc/sector_size=0/einprogress=60
================================================================================

18:00:28
Trace close : program <-> aix : test.dat : (42000000/1513.95)=27741.92 kbytes/s
     demand rate=24912.42 kbytes/s=42000000/(1685.92-0.01))
     current size=14000000   max_size=14000000
  mode =0640  FileSystemType=JFS2 sector size=4096
  oflags =0x302=RDWR  CREAT  TRUNC
  open              1     0.01
  write        140000   238.16   14000000   14000000     102400     102400
  read         280000  1275.79   28000000   28000000     102400     102400
  seek         140003    11.45 average seek delta=-307192
  fcntl             2     0.00
  close             1     0.00
  size         140000
================================================================================
```

หมายเหตุ:

- 140 000 การเขียนของขนาด 102 400 ไบต์
- 280 000 การอ่านของขนาด 102 400 ไบต์
- อัตราของ 27 741.92 KB/วินาที

คอนฟิกูเรชัน MIO เพื่อเพิ่มผลการทำงาน I/O

```
setenv MIO_FILES "* [ trace/stats/kbytes | pf/cache=100m/page=2m/pref=4/stats/direct | trace/stats/kbytes ]"
setenv MIO_DEBUG "OPEN MODULES TIMESTAMP"
setenv MIO_STATS mio_pf.stats

time /mio/alot_buf
```

- วิธีที่ดีในการวิเคราะห์ I/O ของแอ็พพลิเคชันของคุณคือ การใช้รายการโมดูล trace | pf | trace วิธีนี้ คุณสามารถขอรับผลการทำงานที่แอ็พพลิเคชันมองเห็นจากแคช **pf** และผลการทำงานที่แคช **pf** มองเห็นจากระบบปฏิบัติการ
- แคชโกลบอล **pf** จะมีขนาด 100 MB แต่ละเพจจะมีขนาด 2 MB จำนวนของเพจที่ต้องการดึงข้อมูลออกก่อนคือสี่เพจ แคช **pf** จะทำการเรียกของระบบ I/O โดยตรงแบบอะซิงโครนัส
- ไฟล์วินิจฉัยเอาต์พุตคือ mio_pf.stats สำหรับข้อมูล **ดีบัก** ข้อมูลโมดูล **การติดตาม** และข้อมูลโมดูล **pf** ค่าทั้งหมดจะอยู่ในหน่วยกิโลไบต์

ผลลัพธ์ของการทดสอบผลการทำงาน

- เวลาของการประมวลผลคือ 15:41
- ไฟล์เอาต์พุตการวินิจฉัยการวิเคราะห์ MIO คือ mio_pf.stats

```
MIO statistics file : Thu May 26 17:10:12 2005
hostname=uriage  : with Legacy aio available
Program=/mio/alot_buf
MIO library libmio.a 3.0.0.60  AIX 64 bit addressing built Apr 19 2005 15:07:35
MIO_INSTALL_PATH=
MIO_STATS       =mio_fs.stats
MIO_DEBUG       = MATCH OPEN MODULES TIMESTAMP
MIO_FILES       =* [ trace/stats/kbytes | pf/cache=100m/page=2m/pref=4/stats/direct | trace/stats/kbytes ]
MIO_DEFAULTS    =

MIO_DEBUG OPEN =T
MIO_DEBUG MODULES =T
MIO_DEBUG TIMESTAMP =T

17:10:12
Opening file test.dat
   modules[79]=trace/stats/kbytes|pf/cache=100m/page=2m/pref=4/stats/direct|trace/stats/kbytes
   trace/stats={mioout}/noevents/kbytes/nointer
 pf/nopffw/release/global=0/asynchronous/direct/bytes/cache_size=100m/page_size=2m/prefetch=4/st
ride=1/stats={mioout}/nointer/noretain/nolistio/notag/noscratch/passthru={0:0}
   trace/stats={mioout}/noevents/kbytes/nointer
   aix/nodebug/trunc/sector_size=0/einprogress=60
================================================================================

17:25:53
Trace close : pf <-> aix : test.dat : (41897728/619.76)=67603.08 kbytes/s
     demand rate=44527.71 kbytes/s=41897728/(940.95-0.01))
     current size=14000000   max_size=14000000
  mode =0640  FileSystemType=JFS2 sector size=4096
  oflags =0x8000302=RDWR  CREAT  TRUNC  DIRECT
  open             1     0.01
```

```
  ill form          0   mem misaligned 0
  write             1     0.21       1920       1920    1966080    1966080
  awrite         6835     0.20   13998080   13998080    2097152    2097152
   suspend       6835   219.01    63855.82 kbytes/s
  read              3     1.72       6144       6144    2097152    2097152
  aread         13619     1.02   27891584   27891584    1966080    2097152
   suspend      13619   397.59    69972.07 kbytes/s
  seek          20458     0.00 average seek delta=-2097036
  fcntl             5     0.00
  fstat             2     0.00
  close             1     0.00
  size           6836

17:25:53
pf close for test.dat
   50 pages of 2097152 bytes   4096 bytes per sector
   6840/6840 pages not preread for write
   7 unused prefetches out of 20459 : prefetch=4
         6835 write behinds
   bytes transferred / Number of requests
   program -->   14336000000/140000 --> pf --> 14336000000/6836   --> aix
   program <--   28672000000/280000 <-- pf <-- 28567273472/13622  <-- aix

17:25:53
pf close for global cache 0
   50 pages of 2097152 bytes   4096 bytes per sector
   6840/6840 pages not preread for write
   7 unused prefetches out of 20459 : prefetch=0
         6835 write behinds
   bytes transferred / Number of requests
   program -->   14336000000/140000 --> pf --> 14336000000/6836   --> aix
   program <--   28672000000/280000 <-- pf <-- 28567273472/13622  <-- aix

17:25:53
Trace close : program <-> pf : test.dat : (42000000/772.63)=54359.71 kbytes/s
     demand rate=44636.36 kbytes/s=42000000/(940.95-0.01))
     current size=14000000   max_size=14000000
  mode =0640  FileSystemType=JFS2 sector size=4096
  oflags =0x302=RDWR  CREAT  TRUNC
  open              1     0.01
  write        140000   288.88   14000000   14000000     102400     102400
  read         280000   483.75   28000000   28000000     102400     102400
  seek         140003    13.17 average seek delta=-307192
  fcntl             2     0.00
  close             1     0.00
  size         140000
================================================================================
```

หมายเหตุ: โปรแกรมจะเรียกใช้งาน 140 000 การเขียนของขนาด 102 400 ไบต์ และ 280 000 การอ่านของขนาด 102 400 ไบต์ แต่โมดูล pf จะเรียกใช้งาน 6 836 การเขียน (ของ 6 835 การเขียนแบบอะซิงโครนัส) ของขนาด 2 097 152 ไบต์และเรียกทำงาน 13 622 การอ่าน (ของ 13 619 ซึ่งเป็นการอ่านแบบอะซิงโครนัส) ของขนาด 2 097 152 อัตราคือ 54 359.71 KB/วินาที

ผลการทำงานของระบบไฟล์

คอนฟิกูเรชันของชนิดของระบบไฟล์ทั้งหลายที่สนับสนุนโดย AIX จะมีผลกระทบกับผลการทำงานของระบบโดยรวม และจะใช้เวลานานเพื่อทำการเปลี่ยนแปลงหลังจากการติดตั้ง

หากต้องการดูข้อมูลพื้นฐานเกี่ยวกับระบบไฟล์ โปรดดู *Operating system and device management*

ชนิดของระบบไฟล์

AIX สนับสนุนชนิดของระบบไฟล์อยู่สองชนิดคือ: ระบบโลคัลไฟล์และระบบรีโมตไฟล์

ระบบไฟล์ต่อไปนี้จะถูกจัดกลุ่มเป็นระบบโลคัลไฟล์:

- ระบบไฟล์ที่เจอร์นัลแล้ว
- ระบบไฟล์ที่เจอร์นัลแล้วซึ่งได้รับการปรับปรุง
- ระบบไฟล์ CD ROM
- ระบบไฟล์บนดิสก์ RAM

ระบบไฟล์ต่อไปนี้จะถูกจัดกลุ่มเป็นระบบรีโมตไฟล์:

- ระบบเน็ตเวิร์กไฟล์
- ระบบไฟล์แบบขนาดทั่วไป

ระบบไฟล์ที่เจอร์นัลแล้วหรือ JFS

การทำเจอร์นัลระบบไฟล์จะอนุญาตให้ใช้สำหรับการกู้คืนระบบไฟล์แบบด่วน หลังจากที่ระบบพังโดยทำบันทึกการทำงานเมตาดาต้าของไฟล์

ด้วยการเปิดใช้งานการทำบันทึกการทำงานระบบไฟล์ ระบบจะบันทึกการเปลี่ยนแปลงทุกครั้งในเมตาดาต้าของไฟล์ ลงในพื้นที่ที่สงวนไว้ของระบบไฟล์ การดำเนินการเขียนที่แท้จริง จะดำเนินการหลังจากการทำบันทึกการทำงานของการเปลี่ยนแปลงในเมตาดาต้า เสร็จสิ้นลงแล้ว

เนื่องจาก JFS ถูกสร้างขึ้นเพื่อใช้กับเคอร์เนลแบบ 32 บิตในรีลีสก่อนหน้านี้ของ AIX ซึ่งไม่ได้ปรับแต่งไว้สำหรับสภาวะแวดล้อมเคอร์เนลแบบ 64 บิต อย่างไรก็ตาม JFS อาจยังถูกใช้กับเวอร์ชัน AIX 6.1 และใหม่กว่า ของสภาวะแวดล้อมเคอร์เนล 64 บิต

JFS ที่ปรับปรุงแล้ว

JFS ที่ปรับปรุงแล้ว หรือ JFS2 คือระบบไฟล์การทำเจอร์นัล AIX เนทีฟอื่น

JFS ที่ปรับปรุงแล้วคือดีฟอลต์ระบบไฟล์สำหรับสภาวะแวดล้อมเคอร์เนลแบบ 64 บิต เนื่องจากข้อจำกัดเกี่ยวกับพื้นที่แอดเดรสของเคอร์เนลแบบ 32 บิต JFS ที่ปรับปรุงแล้วจะไม่แนะนำให้ใช้ในสภาวะแวดล้อมเคอร์เนลแบบ 32 บิต

การสนับสนุนชุดข้อมูลมีการรวมเข้าใน JFS2 เป็น ส่วนหนึ่งของระบบปฏิบัติการ AIX ชุดข้อมูลคือยูนิตของการควบคุมดูแลข้อมูล ซึ่งประกอบด้วยแผนผังไดเร็กทอรี ที่มีไดเร็กทอรีรากอย่างน้อยหนึ่งไดเร็กทอรี การควบคุมดูแล อาจรวมถึงการสร้างชุดข้อมูลใหม่ การสร้างและการเก็บรักษาสำเนา ทั้งหมด (เรพพลิกา) ของชุดข้อมูลบนคอลเล็กชันของเซิร์ฟเวอร์ หรือ การย้ายชุดข้อมูลไปยังเซิร์ฟเวอร์อื่น ชุดข้อมูลอาจมีอยู่เป็นส่วนหนึ่ง ของระบบไฟล์ที่ติดตั้งไว้ นั่นคือ อินสแตนซ์ของระบบไฟล์ที่ติดตั้งไว้

อาจมีชุดข้อมูลจำนวนมาก การสนับสนุนชุดข้อมูลเปิดใช้งานใน JFS2 โดยใช้คำสั่ง **mkfs -o dm=on** โดยดีฟอลต์ การสนับสนุนชุดข้อมูลไม่เปิดใช้งาน อินสแตนซ์ของ JFS2 ที่เปิดใช้งานชุดข้อมูล สามารถถูกจัดการผ่าน Dataset Services Manager (DSM) ได้

ความแตกต่างระหว่าง JFS และ JFS ที่ปรับปรุงแล้ว

มีความแตกต่างระหว่าง JFS และ JFS ที่ปรับปรุงแล้ว

ตารางที่ 4. ความแตกต่างในเชิงการทำงานระหว่าง JFS และ JFS ที่ปรับปรุงแล้ว

ฟังก์ชัน	JFS	JFS ที่ปรับปรุงแล้ว
การ Optimization	เคอร์เนลแบบ 32 บิต	เคอร์เนลแบบ 64 บิต
ขนาดสูงสุดของระบบไฟล์	32 เทราไบต์	4 เพทาไบต์ **หมายเหตุ:** นี่คือข้อจำกัดในเชิงของสถาปัตยกรรม AIX จะสนับสนุนมากสุด 16 เทราไบต์ในปัจจุบัน
ขนาดสูงสุดของไฟล์	64 กิกะไบต์	4 เพทาไบต์ **หมายเหตุ:** นี่คือข้อจำกัดในเชิงของสถาปัตยกรรม AIX จะสนับสนุนมากสุด 16 เทราไบต์ในปัจจุบัน
จำนวนของ I-nodes	ไม่เปลี่ยนแปลงขณะสร้างระบบไฟล์	ไดนามิก จำกัดพื้นที่ดิสก์
สนับสนุนไฟล์ขนาดใหญ่	ตามอ็อพชันการประกอบเข้า	ดีฟอลต์
การรวมแฟรกเมนต์แบบออนไลน์	ใช่	ใช่
namefs	ใช่	ใช่
DMAPI	ไม่ใช่	ใช่
การบีบอัด	ใช่	ไม่ใช่
โควต้า	ใช่	ใช่
อัพเดตที่เลื่อนออกไป	ใช่	ไม่ใช่
ส่วนสนับสนุน I/O โดยตรง	ใช่	ใช่

หมายเหตุ:

- การโคลนด้วยการสำรองระบบด้วย **mksysb** จากระบบ JFS2 ที่เปิดใช้งานขนาด 64 บิตลงไปยังระบบแบบ 32 บิตจะไม่เป็นผลสำเร็จ
- ไม่เหมือนกับระบบไฟล์ JFS ระบบไฟล์ JFS2 จะไม่อนุญาตให้ **link( )** API ถูกนำมาใช้บนไดเร็กทอรีชนิดไบรารี ข้อจำกัดนี้อาจทำให้แอ็พพลิเคชันบางตัว ที่ทำงานอย่างถูกต้องบนระบบไฟล์ JFS เกิดความล้มเหลวบนระบบไฟล์ JFS2

การทำเจอร์นัล:

ก่อนที่จะเขียนโค้ดจริง การทำเจอร์นัลระบบไฟล์จะบันทึกการทำงานเมตาดาต้า ดังนั้นจึงเกิดผลเสียสำหรับการใช้ที่ทำให้เขียนทรูพุตได้ช้าลง

วิธีหนึ่งของการปรับปรุงผลการทำงานภายใต้ JFS คือ การปิดใช้งานการทำบันทึกการทำงานโดยใช้อ็อพชันการประกอบเข้า **แบบไม่สมบูรณ์** หมายเหตุ ผลการทำงานที่ปรับปรุงแล้วจะบรรลุผลสำเร็จในการใช้เมตาดาต้าที่สมบูรณ์ ดังนั้น ให้ใช้อ็อพชันนี้ด้วยความระมัดระวังสูงสุด เนื่องจากระบบที่ขัดข้องสามารถทำให้ระบบไฟล์ ประกอบเข้ากับอ็อพชันนี้ที่ไม่สามารถกู้คืนได้

ในทางตรงกันข้ามกับ JFS JFS ไม่อนุญาตให้คุณปิดใช้งานการทำบันทึกการทำงาน เมตาดาต้า อย่างไรก็ตาม การนำไปปฏิบัติของการทำเจอร์นัลบน JFS ที่ปรับปรุงแล้ว จะทำให้เหมาะสมกับการจัดการแอ็พพลิเคชันเมตาดาต้าที่ละเอียดถี่ถ้วน ดังนั้น ผลเสียของผลการทำงานจะไม่สูงมากนักภายใต้ JFS ที่ปรับปรุงแล้วภายใต้ JFS

การจัดการไดเร็กทอรีอย่างเป็นระบบ:

โหนดดัชนี หรือ i-node คือโครงสร้างข้อมูลที่เก็บคุณสมบัติของไฟล์ และไดเร็กทอรีทั้งหมดไว้ เมื่อโปรแกรมมองหาไฟล์ โปรแกรมจะค้นหา i-node ที่เหมาะสมโดยการมองหาชื่อไฟล์ในไดเร็กทอรี

เนื่องจากการดำเนินการเหล่านี้ได้ถูกดำเนินการบ่อย กลไกที่ใช้สำหรับการค้นหาจึงเป็นส่วนสำคัญ

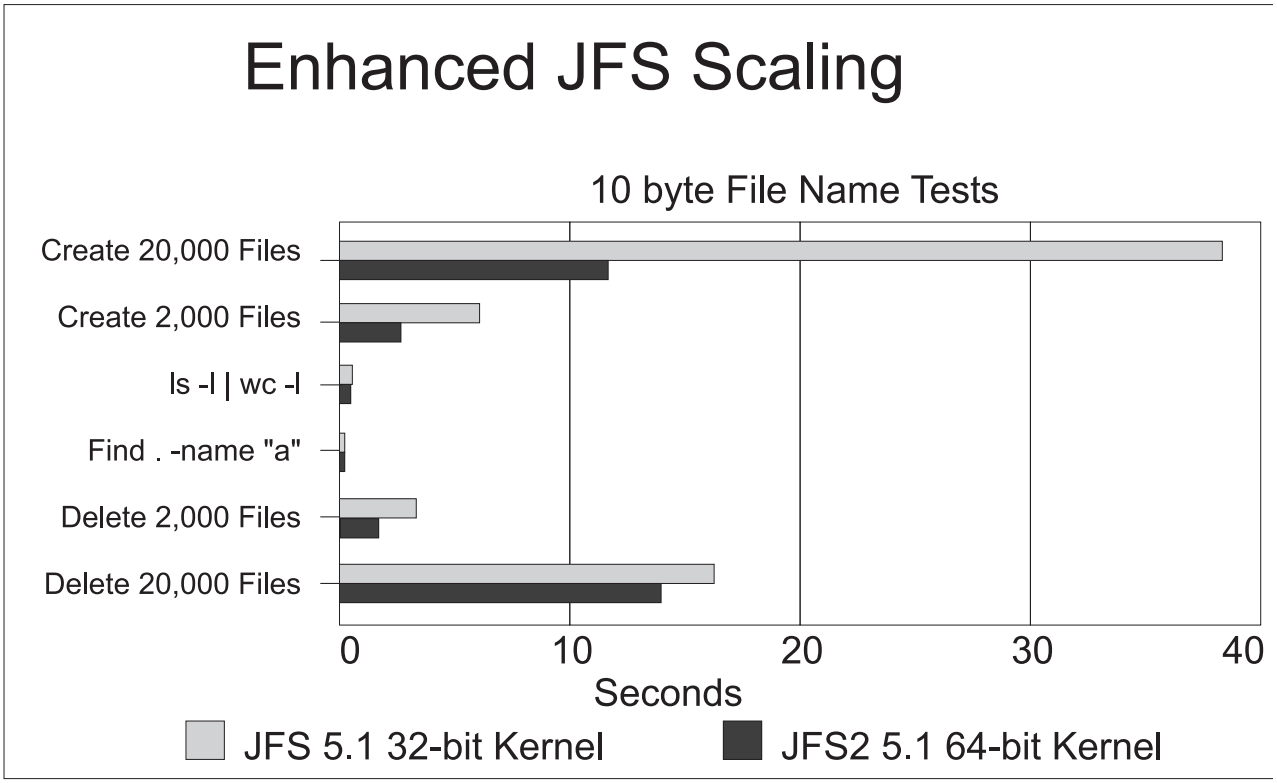
JFS จะใช้การจัดการเชิงเส้นอย่างเป็นระบบสำหรับไดเร็กทอรี ดังนั้น จึงทำการค้นหาแบบเชิงเส้นด้วยเช่นกัน ในทางตรงกันข้าม JFS ที่ปรับปรุงแล้วจะใช้การแทนค่า binary tree สำหรับไดเร็กทอรี ดังนั้น จึงเข้าถึงไฟล์ได้อย่างรวดเร็ว

การปรับสเกล:

ประโยชน์หลักของการใช้ Enhanced JFS แทน JFS คือการปรับสเกล

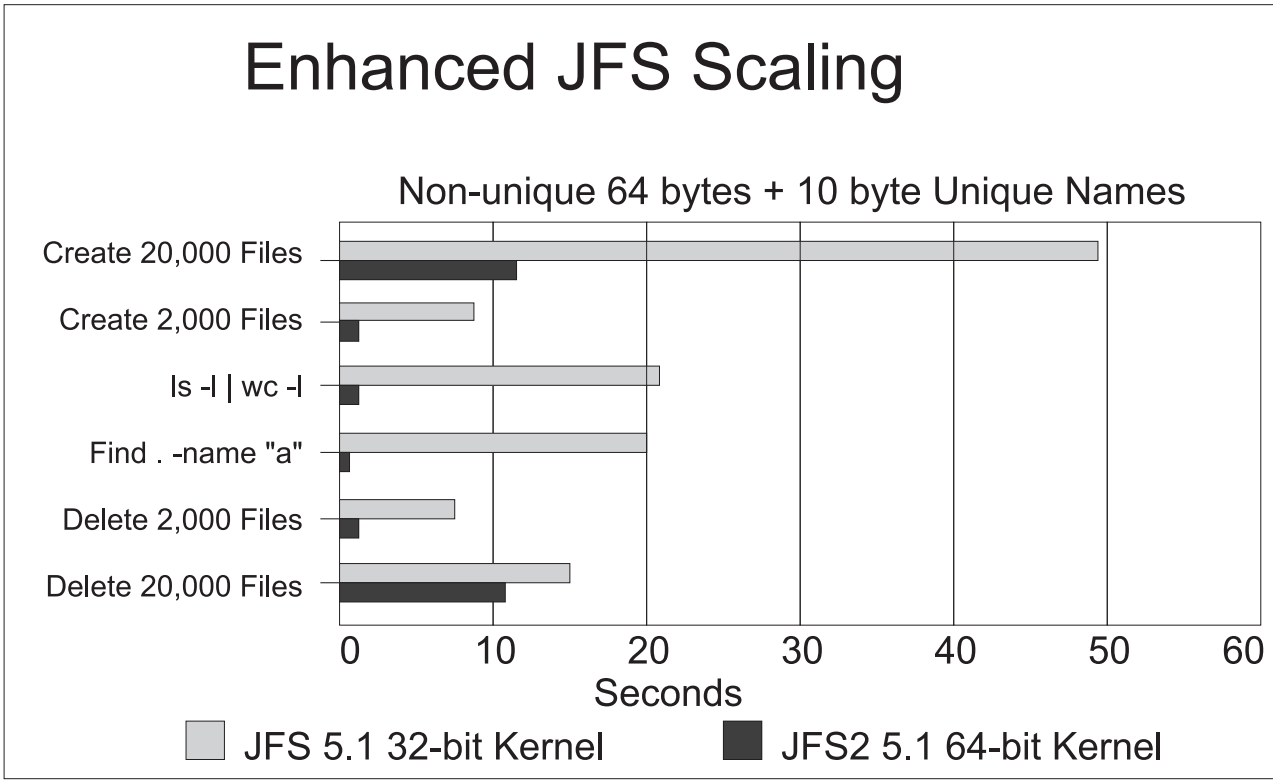
Enhanced JFS ช่วยให้สามารถจัดเก็บไฟล์ขนาดใหญ่กว่า JFS เดิมเป็นอย่างมาก ขนาดสูงสุดของไฟล์ภายใต้ JFS คือ 64 กิกะไบต์ ภายใต้ Enhanced JFS ในปัจจุบัน AIX สนับสนุน ไฟล์ที่มีขนาดถึง 16 เทระไบต์ แม้ว่าอาจมีการตั้งค่าสถาปัตยกรรมระบบไฟล์ให้สามารถจัดการขนาดไฟล์ได้สูงสุดถึง 4 เพทะไบต์ในท้ายที่สุด

ประเด็นการปรับสเกลอีกเรื่องหนึ่งคือการเข้าถึงไฟล์จำนวนมาก ภาพสาธิตต่อไปนี้แสดงให้เห็นว่า Enhanced JFS สามารถปรับปรุงประสิทธิภาพ การเข้าถึงชนิดนี้ได้อย่างไร



ตัวอย่างข้างบนประกอบด้วยการสร้าง การลบ และการค้นหาไดเร็กทอรี ที่มีชื่อไฟล์เฉพาะขนาด 10 ไบต์ ผลลัพธ์แสดงให้เห็นว่าการสร้างและการลบ ไฟล์ภายใต้ Enhanced JFS เร็วกว่าภายใต้ JFS เป็นอย่างมาก ประสิทธิภาพการค้นหาเกือบจะเท่ากันในระบบไฟล์ ทั้งสองประเภท

ตัวอย่างข้างล่างแสดงให้เห็นว่าผลการดำเนินงานสร้าง ลบ และค้นหา โดยทั่วไปแล้ว บน Enhanced JFS เร็วกว่า JFS ได้อย่างไร เมื่อใช้ชื่อไฟล์ ไม่เฉพาะ ในตัวอย่างนี้ มีการเลือกชื่อไฟล์ที่มี ข้อมูล 64 ไบต์แรกเหมือนกันตามด้วยชื่อเฉพาะอีก 10 ไบต์ ภาพสาธิตต่อไปนี้แสดงผลของการทดสอบนี้:



ในหมายเหตุที่เกี่ยวข้อง การแคชชื่อไฟล์ที่ยาว (มากกว่า 32 อักขระ) ได้รับการสนับสนุนในทั้งแคชชื่อของ JFS และ Enhanced JFS การสนับสนุนนี้ช่วยพัฒนาประสิทธิภาพการดำเนินงานของไดเร็กทอรี เช่น คำสั่ง **ls** และ **find** บนไดเร็กทอรีที่มีรายการชื่อไฟล์ที่ยาวจำนวนมาก

ระบบไฟล์ CD ROM

ระบบไฟล์ CD ROM จะเป็นระบบไฟล์แบบอ่านอย่างเดียวที่จะถูกเก็บอยู่บนสื่อบันทึก CD ROM

AIX สนับสนุนชนิดของระบบไฟล์ CD-ROM ที่หลากหลายตามที่กล่าวถึงในส่วนของ ชนิดของระบบไฟล์ ใน *Operating system and device management*

ระบบไฟล์ RAM

RAM ดิสก์เป็นดิสก์ไดร์ฟจำลองที่อยู่ในหน่วยความจำ

RAM ดิสก์ได้รับการออกแบบมาเพื่อให้ประสิทธิภาพ I/O ที่สูงขึ้นกว่า ฟิสิคัลไดร์ฟเป็นอย่างมาก และโดยปกติแล้ว ใช้เพื่อแก้ปัญหาคอขวด I/O ที่มี ไฟล์ชั่วคราว ขนาดสูงสุดของระบบไฟล์ RAM ถูกจำกัดโดยจำนวนของ หน่วยความจำระบบที่มีอยู่ คุณ

สามารถสร้างระบบไฟล์บนอุปกรณ์ RAM ดิสก์เพื่อทำให้ระบบไฟล์มีอยู่สำหรับการใช้ระบบไฟล์ปกติ อย่าใช้ RAM ดิสก์สำหรับข้อมูลถาวร เนื่องจากข้อมูลทั้งหมดจะหายไปถ้าระบบเสียหายหรือ รีบูต

ระบบไฟล์เครือข่าย

ระบบไฟล์เครือข่าย (Network File System หรือ NFS) คือระบบไฟล์แบบแจกจ่ายที่ อนุญาตให้คุณเข้าถึงไฟล์และไดเร็กทอรีที่อยู่บนรีโมตคอมพิวเตอร์ และ จัดการกับไฟล์และไดเร็กทอรีเหล่านั้นได้ราวกับว่าเป็นแบบโลคัล ตัวอย่างเช่น คุณ สามารถใช้คำสั่งระบบปฏิบัติการเพื่อสร้าง ลบ อ่าน บันทึก และตั้งค่า แอ็ตทริบิวต์ไฟล์สำหรับไฟล์และไดเร็กทอรีแบบรีโมตได้

การปรับประสิทธิภาพและประเด็นอื่นๆ เกี่ยวกับ NFS มีอยู่ในหัวข้อ ประสิทธิภาพ NFS

ระบบ Name File (NameFS)

NameFS นำเสนอฟังก์ชันของจุดติดตั้งไฟล์บนไฟล์และไดเร็กทอรีบนไดเร็กทอรี (เรียกอีกอย่างว่าซอฟต์เมาต์) ซึ่งอนุญาตให้คุณติดตั้งแผนผังย่อยของระบบไฟล์ไว้ใน ตำแหน่งอื่นในพื้นที่ว่างชื่อไฟล์ ซึ่งช่วยให้สามารถเข้าถึงไฟล์ โดยใช้ชื่อพาธที่แตกต่างกันสองชื่อได้

ฟังก์ชันนี้ยังมีประโยชน์ในการแก้ไขแอ็ตทริบิวต์การติดตั้งสำหรับบาง ไดเร็กทอรีด้วย ตัวอย่างเช่น ถ้าไฟล์อยู่ในไดเร็กทอรีเฉพาะหนึ่งที่ต้องการการสนับสนุน I/O โดยตรง แต่ระบบไฟล์ทั้งระบบไม่เหมาะสมสำหรับ I/O โดยตรง ไดเร็กทอรีหรือไฟล์นั้นสามารถมีการติดตั้งใหม่ด้วย **namefs** โดยใช้ แฟล็ก **–o dio** (สมมุติว่าชนิดระบบไฟล์มีการสนับสนุนอ็อบเจ็กต์ **dio**)

ระบบไฟล์ NameFS คือโลจิคัลเอนทิตีอย่างแท้จริง เอนทิตีนี้มีอยู่ในระหว่างเวลา ที่ติดตั้งเท่านั้น และทำหน้าที่เป็นเพียงสื่อในการจัดกลุ่มไฟล์ด้วยเหตุผลทางตรรกะ เท่านั้น การดำเนินงานทั้งหมดบนอ็อบเจ็กต์ที่เข้าถึงผ่านทาง NameFS มีการดำเนินการโดยระบบไฟล์ฟิสิคัลที่มีอ็อบเจ็กต์นั้น และการใช้ฟังก์ชันและความหมายของระบบ ไฟล์นั้นดำเนินการในลักษณะราวกับว่าไม่มีการใช้ NameFS

ระบบไฟล์ NameFS มีการสร้างขึ้นโดยการติดตั้งชื่อพาธ `path1` บน ชื่อพาธอื่น `path2` อ็อบเจ็กต์ที่ระบุโดย `path1` และ `path2` ต้อง เป็นไฟล์หรือไดเร็กทอรีปกติ และชนิดของอ็อบเจ็กต์ต้อง ตรงกัน หลังจากการติดตั้งไฟล์บนไฟล์แล้ว อ็อบเจ็กต์ที่เข้าถึงผ่านทาง `path2` คือ อ็อบเจ็กต์ที่ระบุโดย `path1` หลังจากการติดตั้งไดเร็กทอรีบนไดเร็กทอรีแล้ว อ็อบเจ็กต์ที่เข้าถึงผ่านทาง `path2/`*<ชื่อพาธ>* คือ อ็อบเจ็กต์ที่ระบุโดย `path1/`*<ชื่อพาธ>* อินเตอร์เฟสการเขียนโปรแกรม NameFS ครอบคลุมอยู่ในการเรียกระบบ อินเตอร์เฟสไฟล์มาตรฐาน NameFS มีการเข้าถึงโดยการระบุ **gfstype** ในโครงสร้าง **vmount** ที่ส่งผ่านไปยังการเรียกระบบ `vmount()` อินเตอร์เฟสผู้ใช้ใน NameFS คือคำสั่ง **mount** คำสั่ง **mount** รับรู้ **vfs** ชนิด *namefs* ว่าเป็นอ็อพชันที่ถูกต้องสำหรับแฟล็ก **–v**

หมายเหตุ: ระบบไฟล์ NameFS ไม่สามารถส่งออกโดย NFS ได้

ระบบไฟล์แบบขนานทั่วไป

ระบบไฟล์แบบขนานทั่วไป หรือ GPFS™ มีผลการทำงานสูง ซึ่งเป็นระบบไฟล์ที่แบ่งใช้ดิสก์ซึ่งสามารถจัดเตรียมการเข้าถึงข้อมูลแบบเร็วกับโหนดทั้งหมดที่อยู่ในคลัสเตอร์เซิร์ฟเวอร์ แอ็พพลิเคชันแบบขนานและอนุกรมที่เข้าถึงไฟล์โดยใช้อินเตอร์เฟสของระบบไฟล์ UNIX มาตรฐาน เช่น แอ็พพลิเคชันที่อยู่ใน AIX

GPFS จัดเตรียมผลการทำงานที่สูงโดยการ strip I/O ระหว่างดิสก์จำนวนมาก มีสภาพพร้อมใช้งานสูงผ่านการทำบันทึกการทำงาน และการรองรับความล้มเหลวทั้งเซิร์ฟเวอร์และดิสก์

ข้อมูลที่เกี่ยวข้อง:

GPFS บนคลัสเตอร์ AIX: High Performance File System Administration Simplified

Inhibitors ประสิทธิภาพที่อาจเกิดขึ้นได้สำหรับ JFS และ Enhanced JFS

มีสถานการณ์หลายอย่างที่สามารถยับยั้งประสิทธิภาพ JFS และ Enhanced JFS ที่อาจเกิดขึ้นได้

การทำบันทึกการทำงานระบบไฟล์ที่มีผลกับทรูพุตของระบบไฟล์

เนื่องจากการดำเนินการเขียนจะถูกดำเนินการหลังจากที่ทำบันทึกการทำงานของเมตาดาต้าเสร็จสิ้นแล้ว การเขียนทรูพุตอาจได้รับผลกระทบ

สำหรับคำอธิบายของวิธีการหลีกเลี่ยงผลการทำงานในทางลบที่เชื่อมโยงกับการทำบันทึกการทำงาน โปรดดู “ผลการทำงานของโลจิคัลวอลุ่มและดิสก์ I/O” ในหน้า 195

การบีบอัดและการแตกแฟรกเมนต์

ระบบไฟล์ที่เจอร์นัลแล้วจะสนับสนุนระบบไฟล์ที่แตกแฟรกเมนต์และระบบไฟล์ที่บีบอัด เพื่อประหยัดพื้นที่ดิสก์

โดยทั่วไป การบีบอัดข้อมูลจะประหยัดพื้นที่ดิสก์ซึ่งเกี่ยวข้องกับปัจจัยทั้งสองแบบ อย่างไรก็ตาม การแตกแฟรกเมนต์และการบีบอัดอาจทำให้เสียผลการทำงาน ที่เชื่อมโยงกับกิจกรรมการจัดสรรที่เพิ่มมากขึ้น สำหรับคำอธิบายถึงวิธีการที่การบีบอัดและการแตกแฟรกเมนต์อาจกระทบกับผลการทำงาน โปรดดู “ผลการทำงานของโลจิคัลวอลุ่มและดิสก์ I/O” ในหน้า 195

หากต้องการปรับปรุงผลการทำงาน ทั้ง JFS และ JFS ที่ปรับปรุงจะอนุญาตให้ใช้สำหรับการรวมแฟรกเมนต์แบบออนไลน์ ระบบไฟล์สามารถประกอบเข้าและสามารถเข้าถึงได้ขณะที่กระบวนการรวมแฟรกเมนต์ กำลังดำเนินการอยู่

การปรับปรุงผลการทำงานของระบบไฟล์

มีนโยบายและกลไกที่หลากหลาย ซึ่งคุณสามารถใช้เพื่อปรับปรุงผลการทำงานของระบบไฟล์ ภายใต้ AIX ได้

Read ahead หน้าตามลำดับ

VMM คาดการณ์ความต้องการในอนาคตสำหรับหน้าของไฟล์โดยการสังเกต รูปแบบซึ่งโปรแกรมเข้าถึงไฟล์

เมื่อโปรแกรมเข้าถึงหน้าที่ต่อเนื่องกันสองหน้าของไฟล์ VMM สมมุติว่า โปรแกรมจะเข้าถึงไฟล์ตามลำดับ และ VMM จัดตารางเวลา การอ่านตามลำดับเพิ่มเติมของไฟล์ การอ่านเหล่านี้ซ้อนเหลื่อมกับ การประมวลผลโปรแกรม และจะทำให้ข้อมูลมีอยู่สำหรับโปรแกรมเร็วขึ้นกว่าถ้า VMM ต้องรอให้โปรแกรมเข้าถึงหน้าถัดไป ก่อนการเริ่มต้น I/O

สำหรับ JFS จำนวนหน้าที่จะ read ahead มีการกำหนดโดย VMM thresholds ต่อไปนี้:

minpgahead
จำนวนของหน้า read ahead เมื่อ VMM ตรวจพบรูปแบบการเข้าถึงตามลำดับ เป็นครั้งแรก

ถ้าโปรแกรมยังคงเข้าถึงไฟล์ตามลำดับต่อไป read ahead ถัดไปจะเกิดขึ้นหลังจากโปรแกรมเข้าถึง 2 * *minpgahead* หน้า ถัดไปหลังจาก 4 * *minpgahead* หน้า และเป็นเช่นนี้ต่อไปจนกว่า จำนวนหน้าครบถึง *maxpgahead*

maxpgahead
จำนวนหน้าสูงสุดที่ VMM จะ read ahead ในหนึ่งไฟล์

สำหรับ Enhanced JFS จำนวนหน้าที่จะ read ahead มีการกำหนดโดย VMM thresholds ต่อไปนี้:

j2_minPageReadAhead
จำนวนของหน้า read ahead เมื่อ VMM ตรวจพบรูปแบบการเข้าถึงตามลำดับ เป็นครั้งแรก

ถ้าโปรแกรมยังคงเข้าถึงไฟล์ตามลำดับต่อไป read ahead ถัดไปจะเกิดขึ้นหลังจากโปรแกรมเข้าถึง 2 * *j2_minPageReadAhead* หน้า ถัดไปหลังจาก 4 * *j2_minPageReadAhead* และเป็นเช่นนี้ต่อไป จนกว่าจำนวนหน้าครบถึง *j2_maxPageReadAhead*

j2_maxPageReadAhead
จำนวนหน้าสูงสุดที่ VMM จะ read ahead ในไฟล์ตามลำดับ

Write behind ตามลำดับและแบบสุ่ม

Write behind มีอยู่สองชนิดคือตามลำดับและแบบสุ่ม

โค้ดระบบไฟล์ AIX แบ่งแต่ละไฟล์ทางตรรกะเป็นคลัสเตอร์ 16 KB สำหรับ JFS และคลัสเตอร์ 128 KB สำหรับ Enhanced JFS ด้วยเหตุผลดังต่อไปนี้:

- เพิ่มประสิทธิภาพการบันทึก
- จำกัดจำนวนของหน้าไฟล์ที่สกปรกในหน่วยความจำ
- ลดโอเวอร์เฮดระบบ
- ลดการแบ่งเฟรกเมนต์ดิสก์ให้เหลือน้อยที่สุด

หน้าของพาร์ติชันที่กำหนดจะไม่ถูกบันทึกลงในดิสก์จนกว่าโปรแกรมบันทึก ไบต์แรกของพาร์ติชัน 16 KB ถัดไป ณ จุดนี้ โค้ดระบบไฟล์ บังคับหน้าที่สกปรกสี่หน้าของพาร์ติชันแรกซึ่งจะบันทึก ลงในดิสก์ หน้าของข้อมูลยังคงอยู่ในหน่วยความจำจนกว่าเฟรมของหน้านั้นมีการนำมาใช้ใหม่ในเวลาที่ไม่ต้องการ I/O เพิ่มเติม ถ้าโปรแกรมเข้าถึงหน้าใดๆ อีกครั้ง ก่อนเฟรมของหน้านั้นมีการนำมาใช้ใหม่ จะไม่ต้องการ I/O

ถ้าหน้าของไฟล์ที่สกปรกจำนวนมากยังคงอยู่ในหน่วยความจำและไม่มีการนำไปใช้ใหม่ sync daemon จะบันทึกหน้าดังกล่าวไว้ในดิสก์ ซึ่งอาจส่งผลให้การใช้ประโยชน์ดิสก์ผิดปกติ เพื่อแจกจ่ายกิจกรรม I/O ให้เท่าเทียมกันมากขึ้น คุณสามารถเปิด write behind เพื่อบอกให้ระบบทราบถึงจำนวนหน้าที่จะเก็บไว้ในหน่วยความจำก่อนการบันทึกหน้าเหล่านั้นลงในดิสก์ Write behind threshold ใช้สำหรับแต่ละไฟล์ ซึ่งทำให้หน้าถูกบันทึกลงในดิสก์ ก่อน sync daemon รัน

ขนาดของ write behind partitions และ write behind threshold สามารถเปลี่ยนได้ด้วยคำสั่ง **ioo** โปรดดู “การปรับประสิทธิภาพ write behind ตามลำดับและแบบสุ่ม” ในหน้า 270 สำหรับข้อมูลเพิ่มเติม

หน่วยความจำที่แม็พไฟล์และการซ่อนการเขียน

ไฟล์ปกติจะถูกแม็พกับเซ็กเมนต์เพื่อจัดเตรียม ไฟล์ที่แม็พ ซึ่งหมายความว่า ไฟล์ปกติจะเข้าถึงผ่านบัฟเฟอร์เคอร์เนลแบบดั้งเดิม และรูทีนบล็อก I/O ซึ่งอนุญาตให้ไฟล์ใช้หน่วยความจำเพิ่มเติม เมื่อหน่วยความจำพิเศษ พร้อมใช้งาน การทำแคชไฟล์ไม่ได้ถูกจำกัดไว้กับพื้นที่บัฟเฟอร์เคอร์เนลที่ประกาศแล้ว

ไฟล์สามารถแม็พได้อย่างชัดเจนด้วยรูทีนย่อย **shmat( )** หรือ **mmap( )** แต่วิธีนี้ไม่ได้จัดเตรียมพื้นที่หน่วยความจำเพิ่มเติมสำหรับการแคชของไฟล์เหล่านั้น แอ็พพลิเคชันต่างๆ ที่ใช้รูทีนย่อย **shmat( )** หรือ **mmap( )** เพื่อแม็พไฟล์ที่ชัดเจน และเข้าถึงด้วยแอดเดรสแทนการเข้าถึงโดยรูทีนย่อย **read( )** และ **write( )** อาจหลีกเลี่ยงความยาวพาธของการใช้งานการเรียกของระบบ แต่แอ็พพลิเคชันเหล่านั้นจะสูญเสียประโยชน์จากคุณลักษณะการอ่านของระบบล่วงหน้าและการเขียนด้านหลัง

เมื่อแอ็พพลิเคชันไม่ได้ใช้รูทีนย่อย **write( )** เพจที่แก้ไขจะถูกสะสมไว้ในหน่วยความจำ และเขียนด้วยการสุ่มเมื่อถูกกำจัดโดยอัลกอริธึมการแทนที่เพจ หรือ **sync** daemon ผลลัพธ์นี้ในการเขียนขนาดเล็กจำนวนมากลงดิสก์ ซึ่งเป็นสาเหตุทำให้การใช้ประโยชน์จากดิสก์และ CPU ไม่มีประสิทธิภาพ เช่นเดียวกับการแตกแฟรกเมนต์ที่อาจอ่านไฟล์ในอนาคตได้ช้าลง

กลไก release-behind

Release-behind คือกลไกสำหรับ JFS และ Enhanced JFS ซึ่ง หน้าจะมีการปล่อยเป็นอิสระในทันทีที่ committed ในหน่วยเก็บถาวรโดยการ บันทึก หรือเมื่อหน้าถูกจัดส่งไปยังแอ็พพลิเคชันโดยการอ่าน โซลูชันนี้ช่วยแก้ไขปัญหา การปรับสเกลเมื่อทำ I/O ตามลำดับบนไฟล์ขนาดใหญ่มากที่มีหน้า ซึ่งจะไม่มีการเข้าถึงซ้ำในอนาคตอันใกล้นี้

เมื่อบันทึกไฟล์ขนาดใหญ่โดยไม่ใช้ release-behind การบันทึกจะเร็วมาก ในทุกเมื่อที่มีหน้าซึ่งพร้อมใช้งานอยู่ในรายการที่ว่าง เมื่อจำนวนหน้า ลดลงถึงค่าของพารามิเตอร์ *minfree* VMM จะใช้ขั้นตอนวิธี Least Recently Used (LRU) เพื่อค้นหาหน้าที่จะตัด ออกไป ในส่วนหนึ่งของกระบวนการนี้ VMM ต้องจัดหาล็อกที่ใช้สำหรับการบันทึก ด้วย การช่วงชิงล็อกนี้อาจส่งผลให้ประสิทธิภาพลดลง อย่างมาก

คุณสามารถเปิดใช้งาน release-behind โดยการระบุแฟล็ก release-behind sequential read (rbr), แฟล็ก release-behind sequential write (rbw), หรือแฟล็ก release-behind sequential read and write (rbrw) อย่างใดอย่างหนึ่ง เมื่อออกใช้คำสั่ง **mount**

ผลข้างเคียงของการใช้กลไก release-behind คือการใช้ประโยชน์ CPU เพิ่มขึ้นสำหรับอัตราผลผลิตการอ่านหรือการบันทึกเดียวกัน เมื่อเปรียบเทียบกับ การไม่ใช้ release-behind ที่เป็นเช่นนี้เนื่องจากงานของการทำให้หน้าว่าง ซึ่งโดยปกติแล้ว จะมีการจัดการในภายหลังโดย LRU daemon โปรดทราบด้วยว่า การเข้าถึงหน้าไฟล์ทั้งหมดเกิดขึ้นในดิสก์ I/O เนื่องจากข้อมูลไฟล์ไม่ได้แคชโดย VMM

คุณสามารถใช้คำสั่ง **mount -o rbr** เพื่อใช้ release-behind สำหรับ NFS

ส่วนสนับสนุน I/O โดยตรง

ทั้ง JFS และ JFS ที่ได้รับการปรับปรุงแล้วจะนำเสนอส่วนสนับสนุน I/O ที่เข้าถึงไฟล์โดยตรง

เมธอด I/O โดยตรงจะผ่านแคชของไฟล์ และถ่ายโอนข้อมูลจากดิสก์ไปยังพื้นที่บัฟเฟอร์ของผู้ใช้โดยตรง ซึ่งตรงข้ามกับการใช้นโยบายแคชปกติ ของการแทนที่เพจในหน่วยความจำเคอร์เนล สำหรับคำอธิบายเกี่ยวกับวิธีการปรับ I/O โดยตรง โปรดดู “การปรับ I/O โดยตรง” ในหน้า 277

การดำเนินการเขียนที่หน่วงเวลา

JFS อนุญาตให้คุณเลื่อนเวลาการอัพเดตข้อมูลลงในหน่วยเก็บถาวร การดำเนินการเขียนที่หน่วงเวลาจะบันทึกการดำเนินการดิสก์พิเศษ สำหรับไฟล์ที่ถูกเขียนบ่อยๆ

คุณสามารถเปแดใช้งานคุณลักษณะการเขียนที่หน่วงเวลา โดยเปิดไฟล์ด้วยแฟล็กที่เลื่อนการอัพเดต นั่นคือ **O_DEFER** คุณลักษณะนี้จะแคชข้อมูล ซึ่งอนุญาตให้ดำเนินการอ่านและเขียนได้เร็วกว่าจากการประมวลผลอื่นๆ

ขณะที่เขียนลงไฟล์ซึ่งถูกเปิดด้วยแฟล็กนี้ ข้อมูลจะไม่ถูก commit ในหน่วยเก็บถาวรจนกว่าการประมวลผลจะออกคำสั่ง **fsync** ซึ่งบังคับให้ข้อมูลที่อัพเดตทั้งหมดถูก commit ในดิสก์ และ ถ้าการประมวลผลออกคำสั่งให้ดำเนินการเขียนแบบซิงโครนัส นั่นคือ การประมวลผลที่เปิดไฟล์ด้วยแฟล็ก **O_SYNC** การดำเนินการนั้นจะไม่เลื่อนออก แม้ว่าไฟล์จะถูกสร้างด้วยแฟล็ก **O_DEFER** ก็ตาม

หมายเหตุ: คุณลักษณะนี้ ไม่พร้อมใช้งานสำหรับ JFS ที่ปรับปรุงแล้ว

ส่วนสนับสนุน I/O แบบพร้อมเพียงกัน

JFS ที่ปรับปรุงจะสนับสนุนไฟล์การเข้าถึงไฟล์แบบพร้อมเพียงกัน

คล้ายกับ I/O โดยตรง เมธอดการเข้าถึงนี้จะส่งแคชของไฟล์ และโอนย้ายข้อมูลจากดิสก์ไปยังบัฟเฟอร์พื้นที่ของผู้ใช้ และยังส่งไปยังล็อก inode ที่อนุญาตให้เธรดจำนวนมากอ่านผ่านไปยังไฟล์ระบบแบบพร้อมเพียงกัน

หมายเหตุ: คุณลักษณะนี้ไม่พร้อมใช้งานสำหรับ JFS

แอ็ตทริบิวต์ระบบไฟล์ที่กระทบกับผลการทำงาน

ระบบไฟล์ที่มีขนาดยาวกว่า ก็จะมีการแตกแฟรกเมนต์เพิ่มขึ้นด้วย ด้วยการจัดสรรรีซอร์สแบบไดนามิก บล็อกของไฟล์จะถูกกระจายเพิ่มมากขึ้น ไฟล์ต่อเนื่องจะถูกแตกแฟรกเมนต์ และโลจิคัลวอลุ่ม (LV) ที่ต่อเนื่อง จะถูกแตกแฟรกเมนต์

รายการต่อไปนี้ของสิ่งที่เกิดขึ้น เมื่อไฟล์ถูกเข้าถึงจากโลจิคัลวอลุ่ม ที่แตกแฟรกเมนต์:

- การแอ็กเซสตามลำดับไม่มีลำดับที่ยาวกว่า
- แร็นดิมแอ็กเซสจะช้ากว่า
- ช่วงเวลาการแอ็กเซสจะถูกครอบครองโดยเวลาในการค้นหาที่ยาวนานกว่า

อย่างไรก็ตาม หากไฟล์อยู่ในหน่วยความจำ ผลกระทบเหล่านี้จะลดน้อยลง ผลการทำงานของระบบไฟล์ ยังมีผลมาจากข้อควรพิจารณาเกี่ยวกับฟิสิคัล เช่น:

- ชนิดของดิสก์และจำนวนของอะแด็ปเตอร์
- จำนวนของหน่วยความจำสำหรับการบัฟเฟอร์ไฟล์
- จำนวนของการเข้าถึงไฟล์แบบโลคัลเปรียบเทียบกับแบบรีโมต
- รูปแบบและจำนวนของการเข้าถึงไฟล์โดยแอ็พพลิเคชัน

JFS อนุญาตให้คุณเปลี่ยนขนาดแฟรกเมนต์ของระบบไฟล์สำหรับการใช้ประโยชน์จากพื้นที่ที่ดีกว่า โดยแบ่งย่อยเป็นบล็อกขนาด 4 KB จำนวนของไบต์ต่อ i-node หรือ NBPI จะถูกใช้เพื่อควบคุมจำนวนของ i-nodes ที่สร้างขึ้นสำหรับระบบไฟล์ การบีบอัดสามารถนำมาใช้สำหรับระบบไฟล์ พร้อมกับขนาดของแฟรกเมนต์ที่น้อยกว่า 4 KB ขนาดของแฟรกเมนต์ และการบีบอัดจะกระทบกับผลการทำงาน และจะกล่าวถึงในส่วนต่อไปนี้ :

- “ขนาดแฟรกเมนต์ JFS”
- “การบีบอัด JFS” ในหน้า 266

ขนาดแฟรกเมนต์ JFS

คุณลักษณะแฟรกเมนต์ใน JFS อนุญาตให้ใช้พื้นที่ในระบบไฟล์ เพื่อจัดสรรในส่วนที่มีขนาดน้อยกว่า 4 KB

เมื่อคุณสร้างระบบไฟล์ คุณสามารถระบขนาดของแฟรกเมนต์ในระบบไฟล์ ขนาดที่อนุญาตคือ 512, 1024, 2048 และ 4096 ไบต์ ค่าดีฟอลต์คือ 4096 ไบต์ ไฟล์ที่เล็กกว่าแฟรกเมนต์จะถูกเก็บพร้อมกับแฟรกเมนต์แต่ละตัว เก็บรักษาให้มากเท่ากับพื้นที่ดิสก์ ซึ่งเป็นวัตถุประสงค์หลัก

ไฟล์ที่มีขนาดเล็กกว่า 4096 ไบต์จะถูกเก็บในจำนวนของแฟรกเมนต์ต่อเนื่องที่มีค่าต่ำที่สุด ไฟล์ที่มีขนาดระหว่าง 4096 ไบต์ และ 32 KB (ครอบคลุม) จะถูกเก็บในบล็อกเต็มตั้งแต่หนึ่งบล็อกขึ้นไป (4 KB) และในแฟรกเมนต์จำนวนมาก ที่ต้องการพักส่วนที่เหลือ ตัวอย่างเช่น ไฟล์ขนาด 5632 ไบต์ จะถูกจัดสรรบล็อกขนาด 4 KB หนึ่งบล็อก และชี้ไปยังตัวชี้แรกใน i-node ถ้าขนาดของแฟรกเมนต์คือ 512 จำนวนแปดแฟรกเมนต์จะถูกนำมาใช้สำหรับบล็อกขนาด 4 KB แรก ขนาด 1.5 KB สุดท้ายจะใช้สามแฟรกเมนต์ ซึ่งชี้ไปยังตัวชี้ที่สองใน i-node สำหรับไฟล์ที่ใหญ่กว่า 32 KB การจัดสรรจะถูกทำขึ้นในบล็อกขนาด 4 KB และตัวชี้ i-node จะชี้ไปยังบล็อกขนาด 4 KB เหล่านี้

ไม่ว่าแฟรกเมนต์จะมีขนาดเท่าใด บล็อกเต็มจะถูกนำมาพิจารณาให้มีขนาด 4096 ไบต์ ถ้าในระบบไฟล์ที่มีขนาดแฟรกเมนต์ที่น้อยกว่า 4096 ไบต์ ความต้องการสำหรับบล็อกเต็มสามารถตอบสนองได้โดยลำดับของแฟรกเมนต์ต่อเนื่อง ทั้งหมด 4096 ไบต์ ซึ่งไม่ต้องการเริ่มต้นจำนวนของขอบเขตขนาด 4096 ไบต์

ระบบไฟล์จะลองจัดสรรพื้นที่สำหรับไฟล์ที่อยู่ในแฟรกเมนต์ต่อเนื่อง โดยการกระจายไฟล์ระหว่างโลจิคัลวอลุ่มเพื่อลดจำนวนการจัดสรรระหว่างไฟล์ ที่รบกวนและการแตกแฟรกเมนต์

ผลการทำงานหลักที่เป็นอันตรายต่อระบบไฟล์ที่มีขนาดแฟรกเมนต์ที่เล็ก คือการแตกแฟรกเมนต์พื้นที่ การมีอยู่ของไฟล์ขนาดเล็กที่กระจายระหว่างโลจิคัลวอลุ่ม อาจเป็นไปไม่ได้ที่จะจัดสรรบล็อกพื้นที่ที่ต่อเนื่องหรือใกล้เคียงกัน สำหรับไฟล์ขนาดใหญ่ ผลการทำงานอาจไม่ดีเมื่อเข้าถึงไฟล์ขนาดใหญ่ สำหรับการใช้งานที่มีมาก การแตกแฟรกเมนต์พื้นที่อาจเป็นไปไม่ได้ที่จะจัดสรรพื้นที่สำหรับไฟล์ แม้ว่าจะมีแฟรกเมนต์ว่างจำนวนมาก

ผลกระทบที่ตรงข้ามกับกิจกรรมดิสก์ I/O คือจำนวนของการดำเนินการ I/O สำหรับไฟล์ที่มีขนาด 4 KB ซึ่งเก็บอยู่ในแฟรกเมนต์เดี่ยวที่มีขนาด 4 KB เฉพาะการดำเนินการดิสก์ I/O จำเป็นต้องมีการอ่านหรือการเขียนไฟล์อย่างใดอย่างหนึ่ง ถ้าตัวเลือกของขนาดแฟรกเมนต์ 512 ไบต์ แปดแฟรกเมนต์จะถูกจัดสรรไว้กับไฟล์นี้ และสำหรับการอ่านหรือเขียนเพื่อเสร็จสิ้นการดำเนินการกับดิสก์ I/O เพิ่มเติมทั้งหลาย (การค้นหาดิสก์ การถ่ายโอนข้อมูล และกิจกรรมการจัดสรร) จำเป็นต้องมี ดังนั้นสำหรับระบบไฟล์ที่ใช้แฟรกเมนต์ขนาด 4 KB จำนวนของการดำเนินการดิสก์ I/O อาจจะน้อยกว่าระบบไฟล์ ที่ใช้ขนาดแฟรกเมนต์ที่เล็ก

ส่วนของการตัดสินใจในการสร้างระบบไฟล์แฟรกเมนต์ขนาดเล็กควรเป็นนโยบาย สำหรับการรวมแฟรกเมนต์พื้นที่ในระบบไฟล์นั้นด้วยคำสั่ง **defragfs** นโยบายนี้ต้องพิจารณาต้นทุนผลการทำงานของการรันคำสั่ง **defragfs** โปรดดู “การรวมแฟรกเมนต์ระบบไฟล์” ในหน้า 268

การบีบอัด JFS

ถ้าระบบไฟล์ถูกบีบอัด ข้อมูลทั้งหมดจะถูกบีบอัดโดยอัตโนมัติ โดยใช้การบีบอัด Lempel-Zev (LZ) ก่อนที่จะเขียนลงดิสก์ และข้อมูลทั้งหมดจะถูกขยายโดยอัตโนมัติเมื่ออ่านจากดิสก์ อัลกอริธึม LZ จะแทนที่ลำดับเหตุการณ์ ของสตริงที่กำหนดด้วยตัวชี้ไปยังการเกิดขึ้นในครั้งแรก สำหรับค่าเฉลี่ย 50 เปอร์เซ็นต์ที่ช่วยประหยัดพื้นที่ดิสก์

ข้อมูลระบบไฟล์ที่ถูกบีบอัดที่ระดับของบล็อกโลจิคัล หากการบีบอัดข้อมูลมีหน่วยขนาดใหญ่ (ตัวอย่างเช่น บล็อกโลจิคัลทั้งหมดของไฟล์) จะส่งผลทำให้เกิดการสูญหายของพื้นที่ดิสก์ที่พร้อมใช้งาน ตามการบีบอัดบล็อกโลจิคัลของไฟล์ การสุ่มค้นหาและการอัพเดต จะถูกนำด้วยความเร็วสูง

เมื่อไฟล์ถูกเขียนลงในระบบไฟล์สำหรับการบีบอัดที่ระบุ อัลกอริธึมการบีบอัดจะบีบอัดข้อมูล 4096 ไบต์ (หน้า) ในแต่ละครั้ง และข้อมูลที่บีบอัดจะถูกเขียนลงในจำนวนของแฟรกเมนต์ที่ต่อเนื่อง ด้วยค่าที่ต่ำที่สุด จริงๆ แล้ว ถ้าขนาดแฟรกเมนต์ของระบบไฟล์คือ 4 KB จะไม่มีระยะเวลาในการคืนทุนสำหรับความพยายามบีบอัดข้อมูล ดังนั้น การบีบอัดต้องการแฟรกเมนต์เพื่อนำมาใช้ พร้อมกับขนาดแฟรกเมนต์ ที่เล็กกว่า 4096

แม้ว่า การบีบอัดควรส่งผลถึงพื้นที่ทั้งหมดที่ครอบคลุม มีเหตุผลที่ถูกต้องสำหรับการปล่อยพื้นที่ที่ไม่ได้ใช้งานในระบบไฟล์:

- เนื่องจากระดับที่บล็อกของข้อมูลขนาด 4096 ไบต์แต่ละบล็อกที่จะบีบอัด จะไม่เป็นที่รู้จักในอนาคต ระบบไฟล์จะจองบล็อกของพื้นที่ทั้งหมดไว้ แฟรกเมนต์ที่ไม่ต้องการจะถูกปล่อยหลังจากการบีบอัด แต่นโยบายการจัดสรรที่ครอบคลุมอาจนำไปสู่การบ่งชี้ "ไม่มีพื้นที่ว่าง" ก่อนกำหนด
- พื้นที่ว่างบางส่วนจำเป็นต้องมีเพื่ออนุญาตให้คำสั่ง **defragfs** ทำงาน

นอกจากกิจกรรมดิสก์ I/O ที่เพิ่มขึ้นและปัญหาของการแตกแฟรกเมนต์พื้นที่ว่าง ระบบไฟล์ที่ใช้การบีบอัดข้อมูลจะมีข้อควรพิจารณาเกี่ยวกับผลการทำงานดังต่อไปนี้ :

- การลดระดับในความสามารถในการใช้งานของระบบไฟล์ที่เพิ่มขึ้นตามผลลัพธ์ของการบีบอัดข้อมูล/ กิจกรรมที่แตกการบีบอัด ถ้าเวลาในการบีบอัดและการแตกการบีบอัดข้อมูล ใช้เวลานาน จึงอาจเป็นไปไม่ได้ที่จะใช้ระบบไฟล์ที่บีบอัดข้อมูลโดยเฉพาะในสภาวะแวดล้อมเชิงพาณิชย์ที่ข้อมูลจำเป็นต้อง ใช้งานโดยทันที
- โลจิคัลบล็อกทั้งหมดในระบบไฟล์ที่บีบอัด เมื่อแก้ไขไว้ในครั้งแรก จะถูกจัดสรรพื้นที่ดิสก์ขนาด 4096 ไบต์ และพื้นที่นี้จะถูกจัดสรรอีกครั้ง เมื่อโลจิคัลบล็อกถูกเขียนลงในดิสก์ ต้นทุนผลการทำงาน จะเชื่อมโยงกับการจัดสรรใหม่ ซึ่งไม่ได้เกิดขึ้นในระบบไฟล์ที่ไม่ได้บีบอัด
- หากต้องการดำเนินการบีบอัดข้อมูล วงรอบ 50 CPU ต่อไบต์ จำเป็นต้องมี และวงรอบ 10 CPU ต่อไบต์สำหรับการแตกการบีบอัด การบีบอัดข้อมูล จะวางโหลดบนตัวประมวลผลโดยเพิ่มจำนวนของวงรอบตัวประมวลผล
- การบีบอัด JFS kproc (jfsc) จะรันที่ระดับความสำคัญที่ไม่เปลี่ยนแปลงที่ 30 เพื่อว่า ขณะที่เกิดการบีบอัด CPU ที่ kproc นี้กำลังรันอยู่ อาจไม่พร้อมใช้งานกับการประมวลผลอื่น จนกว่าจะรันที่ระดับความสำคัญที่ดีกว่า

การจัดระเบียบระบบไฟล์ใหม่

คุณสามารถลดการแบ่งเฟรกเมนต์ระบบไฟล์ได้ดังนี้:

- การคัดลอกไฟล์ไปยังสื่อบันทึกแบ็คอัพ
- การสร้างระบบไฟล์ขึ้นใหม่โดยใช้คำสั่ง **mkfs** *fsname* หรือการลบเนื้อหาของระบบไฟล์
- การรีโหลดไฟล์เข้าในระบบไฟล์

โพรซีเดอร์นี้โหลดไฟล์ตามลำดับและลดการแบ่งเฟรกเมนต์ ส่วนต่อไปนี้ให้ข้อมูลเพิ่มเติม:

- “การจัดระเบียบระบบไฟล์ใหม่”
- “การรวมแฟรกเมนต์ระบบไฟล์” ในหน้า 268

การจัดระเบียบระบบไฟล์ใหม่

ส่วนนี้อธิบายขั้นตอนการจัดระเบียบระบบไฟล์ใหม่

ในตัวอย่างต่อไปนี้ ระบบมีโลจิคัลวอลุ่มและ ระบบไฟล์แยกกัน *hd11* (จุดติดตั้ง: /home/op) เนื่องจากเราตัดสินใจว่าระบบไฟล์ *hd11* จำเป็นต้องจัดระเบียบใหม่ เราจึงทำ ดังต่อไปนี้:

1. แบ็คอัพระบบไฟล์โดยใช้ชื่อไฟล์ ถ้าคุณแบ็คอัพระบบไฟล์โดยใช้ i-node แทนชื่อ คำสั่ง **restore** จะวางไฟล์ กลับไปในที่ตั้งดั้งเดิมซึ่งจะไม่แก้ไขปัญหา รันคำสั่งต่อไปนี้:

   ```
   # cd /home/op
   # find . -print | backup -ivf/tmp/op.backup
   ```

   คำสั่งนี้สร้างไฟล์แบ็คอัพ (ในระบบไฟล์อื่น) ซึ่งมีไฟล์ทั้งหมดในระบบไฟล์ ที่จะรับรู้ใหม่ ถ้าพื้นที่ว่างดิสก์บนระบบมีอยู่จำกัด คุณสามารถใช้เทปเพื่อแบ็คอัพระบบไฟล์ได้
2. รันคำสั่งต่อไปนี้:

   ```
   # cd /
   # unmount /home/op
   ```

   ถ้ากระบวนการใดๆ กำลังใช้ /home/op หรือ ไดเร็กทอรีย่อยของโลจิคัลวอลุ่มนั้นอยู่ คุณต้องยุติกระบวนการเหล่านั้นก่อนที่คำสั่ง **unmount** จะสามารถเสร็จสมบูรณ์ได้
3. สร้างระบบไฟล์ขึ้นใหม่บนโลจิคัลวอลุ่ม /home/op ดังนี้:

```
# mkfs /dev/hd11
```

คุณจะได้รับพร้อมต์สำหรับการยืนยัน ก่อนระบบไฟล์เก่าจะถูกทำลาย ชื่อของระบบไฟล์ไม่เปลี่ยนแปลง

4. เมื่อต้องการคืนสภาพสถานการณ์ดั้งเดิม (ยกเว้นว่า /home/op ว่าง) ให้รันดังต่อไปนี้:

```
# mount /dev/hd11 /home/op
# cd /home/op
```

5. เรียกคืนข้อมูลดังนี้:

```
# restore -xvf/tmp/op.backup >/dev/null
```

เอาต์พุตมาตรฐาน ถูกกลับทิศทางไปยัง /dev/null เพื่อหลีกเลี่ยงการแสดงชื่อ ของแต่ละไฟล์ที่มีการเรียกคืน ซึ่งใช้เวลานาน

6. ตรวจทานไฟล์ขนาดใหญ่ที่ตรวจสอบไปก่อนหน้านี้ (โปรดดู การประเมินการวางตำแหน่งไฟล์โดยใช้คำสั่ง fileplace) ดังนี้:

```
# fileplace -piv big1
```

เราเห็นว่าขณะนี้ไฟล์ (เกือบ) ติดกัน:

```
File: big1  Size: 3554273 bytes  Vol: /dev/hd11
Blk Size: 4096  Frag Size: 4096  Nfrags: 868   Compress: no
Inode: 8290  Mode: -rwxr-xr-x  Owner: hoetzel  Group: system

INDIRECT BLOCK: 60307

  Physical Addresses (mirror copy 1)                          Logical Fragment

  ----------------------------------                          ----------------

  0060299-0060306  hdisk1     8 frags    32768 Bytes,   0.9%    0008555-0008562
  0060308-0061167  hdisk1   860 frags  3522560 Bytes,  99.1%    0008564-0009423

  868 frags over space of 869 frags:  space efficiency = 99.9%
  2 fragments out of 868 possible:  sequentiality = 99.9%
```

อ็อพชัน **–i** ที่เราเพิ่ม ในคำสั่ง **fileplace** บ่งชี้ว่าช่องว่างหนึ่งบล็อก ระหว่างแปดบล็อกแรกของไฟล์กับส่วนที่เหลือมีบล็อกทางอ้อม ซึ่งต้องใช้เพื่อเสริมข้อมูล i–node เมื่อความยาวของไฟล์ เกินกว่าแปดบล็อก

บางระบบไฟล์หรือโลจิคัลวอลุ่มไม่ควรจะรับรู้ใหม่ เนื่องจากข้อมูลเป็นแบบชั่วคราว (ตัวอย่างเช่น /tmp) หรือไม่ได้อยู่ในรูปแบบระบบไฟล์ (log) โดยปกติ ระบบไฟล์รากไม่มีการเปลี่ยนแปลงบ่อยนักและแทบไม่จำเป็นต้องรับรู้ใหม่ สามารถทำได้ในโหมดติดตั้ง/ดูแลรักษาเท่านั้น เช่นเดียวกับ /usr เนื่องจาก ไฟล์เหล่านี้จำนวนมากเป็นสิ่งจำเป็นสำหรับการดำเนินงานระบบปกติ

การรวมแฟรกเมนต์ระบบไฟล์

ถ้าระบบไฟล์ได้ถูกสร้างขึ้นด้วยขนาดแฟรกเมนต์ขนาดเล็กกว่า 4 KB ระบบไฟล์นั้นจะกลายเป็นสิ่งจำเป็นในภายหลังเพื่อเคียวรี่ จำนวนของแฟรกเมนต์ที่ไม่สามารถนำมาใช้ได้ซึ่งกระจัดกระจายอยู่ ถ้ามีแฟรกเมนต์ขนาดเล็กที่กระจัดกระจายอยู่เป็นจำนวนมาก จึงเป็นการยากที่จะค้นหาพื้นที่ต่อเนื่องที่พร้อมใช้งาน

หากต้องการกู้คืนพื้นที่ที่กระจัดกระจายขนาดเล็กเหล่านี้ ให้ใช้คำสั่ง **smitty dejfs** หรือคำสั่ง **smitty dejfs2** หรือคำสั่ง **defragfs** อย่างใดอย่างหนึ่ง พื้นที่ว่างบางส่วน ต้องพร้อมใช้งานสำหรับโพรซีเดอร์การรวมแฟรกเมนต์ที่นำมาใช้ ระบบไฟล์ ต้องถูกประกอบเข้าเพื่ออ่าน-เขียน

การปรับผลการทำงานของระบบไฟล์

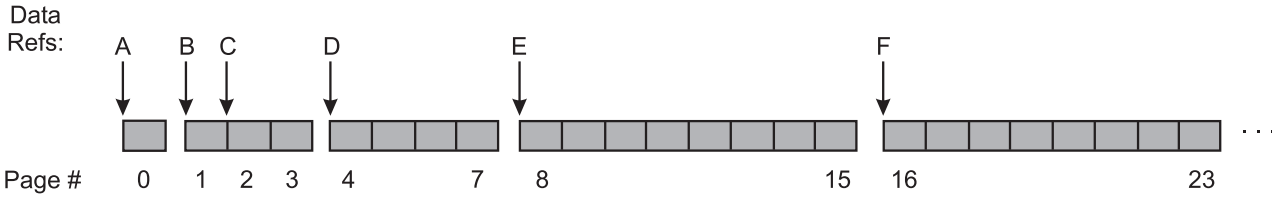
มีหลายแง่มุมในการปรับผลการทำงานของระบบไฟล์

การปรับประสิทธิภาพการอ่านตามลำดับ

คุณลักษณะ read-ahead ตามลำดับของ VMM สามารถปรับปรุง ประสิทธิภาพของโปรแกรมที่เข้าถึงไฟล์ขนาดใหญ่แบบตามลำดับได้

คุณลักษณะ read-ahead ตามลำดับของ VMM มีการอธิบายไว้ใน “Read ahead หน้าตามลำดับ” ในหน้า 262

ข้อมูลต่อไปนี้แสดงสถานการณ์ read-ahead ปกติ



รูปที่ 20. ตัวอย่าง Read-Ahead ตามลำดับ. ภาพสาธิตนี้ แสดงแถวของบล็อกที่จำลองแทร็กที่แบ่งเซกเมนต์ของหมายเลขหน้าไฟล์ บล็อกเซกเมนต์เหล่านี้มีหมายเลขเป็น 0, 1 ถึง 3, 4 ถึง 7, 8 ถึง 15 และ 16 ถึง 23 ขั้นตอนของ read-ahead ตามลำดับแสดงเป็น ข้อความต่อจากภาพสาธิต

ในตัวอย่างนี้ *minpgahead* คือ 2 และ *maxpgahead* คือ 8 (ค่าดีฟอลต์) โปรแกรมกำลังประมวลผลไฟล์ตามลำดับ มีการแสดงเฉพาะข้อมูลอ้างอิงที่มีความสำคัญต่อกลไก read-ahead เท่านั้น โดยกำหนดเป็น A ถึง F ลำดับของขั้นตอน คือ:

A การเข้าถึงไฟล์ครั้งแรกทำให้มีการอ่านหน้าแรก (หน้า 0) ของ ไฟล์ ณ จุดนี้ VMM ไม่ได้กำหนดสมมุติฐานเกี่ยวกับการเข้าถึง แบบสุ่มหรือตามลำดับ

B เมื่อโปรแกรมเข้าถึงไบต์แรกของหน้าถัดไป (หน้า 1) โดยไม่ได้เข้าถึงหน้าอื่นของไฟล์ VMM จะสรุปว่าโปรแกรมกำลังเข้าถึงตามลำดับ และจัดตารางเวลาหน้าเพิ่มเติม *minpgahead* (2) (หน้า 2 และ 3) ที่จะอ่าน ดังนั้นการเข้าถึง B ส่งผลให้มีหน้าที่จะอ่าน ทั้งหมด 3 หน้า

C เมื่อโปรแกรมเข้าถึงไบต์แรกของหน้าแรกที่มีการ read ahead (หน้า 2) VMM จะเพิ่มค่า page-ahead ขึ้นสองเท่า เป็น 4 และจัดตารางเวลาหน้า 4 ถึง 7 ที่จะอ่าน

D เมื่อโปรแกรมเข้าถึงไบต์แรกของหน้าแรกที่มีการ read ahead (หน้า 4) VMM จะเพิ่มค่า page-ahead ขึ้นสองเท่า เป็น 8 และจัดตารางเวลาหน้า 8 ถึง 15 ที่จะอ่าน

E เมื่อโปรแกรมเข้าถึงไบต์แรกของหน้าแรกที่มีการ read ahead (หน้า 8) VMM กำหนดว่าค่า page-ahead เท่ากับ *maxpgahead* และจัดตารางเวลา หน้า 16 ถึง 23 ที่จะอ่าน

F VMM อ่านหน้า *maxpgahead* ต่อไป เมื่อโปรแกรมเข้าถึงไบต์แรกของกลุ่มก่อนหน้านี้ของหน้า read-ahead จนกว่าสิ้นสุดไฟล์

ถ้าโปรแกรมเบี่ยงเบนไปจากรูปแบบการเข้าถึงตามลำดับ และเข้าถึงหน้าของไฟล์ที่ไม่เป็นไปตามลำดับ read-ahead ตามลำดับจะถูกยุติลง คุณลักษณะนั้นอาจจะเรียกคืนด้วยหน้า *minpgahead* ถ้า VMM ตรวจพบว่าโปรแกรมเรียกคืนการเข้าถึงตามลำดับ

ค่า *minpgahead* และ *maxpgahead* สามารถเปลี่ยนได้โดยใช้อ็อพชัน **-r** และ **-R** ในคำสั่ง **ioo** ถ้าคุณกำลังพิจารณาที่จะเปลี่ยนค่าเหล่านี้ โปรดจำว่า:

- ค่าควรจะเป็นค่าจากชุด: 0, 1, 2, 4, 8, 16, และต่อไป การใช้ค่าอื่นอาจทำให้ประสิทธิภาพหรือผลการทำงาน แย่ลง
  - ค่าควรจะยกกำลัง 2 เนื่องจากขั้นตอนวิธีเพิ่มค่าเป็นสองเท่า ของ VMM
  - ค่า *maxpgahead* ที่มากกว่า 16 (reads ahead มากกว่า 64 KB) เกินกว่าความสามารถของไดรเวอร์ อุปกรณ์ดิสก์บางประเภท ในกรณีนั้น ขนาดการอ่านยังคงอยู่ที่ 64 KB
  - สามารถใช้ค่า *maxpgahead* ที่สูงขึ้นในระบบ ซึ่งประสิทธิภาพตามลำดับของ striped โลจิคัลวอลุ่ม มีความสำคัญสูงสุด
- ค่า 0 สำหรับทั้ง *minpgahead* และ *maxpgahead* มีผลยกเลิกกลไก ซึ่งอาจส่งผลกระทบด้านลบต่อประสิทธิภาพ อย่างไรก็ตาม การตั้งค่าเช่นนั้นอาจมีประโยชน์ในบางกรณีที่ I/O เป็นแบบสุ่ม แต่ขนาดของ I/Os ทำให้ขั้นตอนวิธี read-ahead ของ VMM มีผลบังคับใช้
- ค่า *maxpgahead* เป็น 8 หรือ 16 อาจให้ประสิทธิภาพ I/O ตามลำดับสูงสุดสำหรับระบบไฟล์แบบ non-striped
- การเพิ่มค่า read-ahead จาก *minpgahead* เป็น *maxpgahead* เร็วมากจนการเพิ่มค่า *minpgahead* ไม่มีประโยชน์สำหรับขนาดไฟล์ส่วนใหญ่
- Read-Ahead ตามลำดับสามารถปรับแยกกันได้สำหรับ JFS และ Enhanced JFS JFS Page Read-Ahead สามารถปรับด้วย *minpgahead* และ *maxpgahead* ในขณะที่ *j2_minPageReadAhead* และ *j2_maxPageReadAhead* ใช้สำหรับ Enhanced JFS

การปรับประสิทธิภาพ write behind ตามลำดับและแบบสุ่ม

Write behind เกี่ยวข้องกับการบันทึกหน้าที่แก้ไขในหน่วยความจำลงใน ดิสก์แบบอะซิงโครนัสหลังมีจำนวนถึง threshold แทนที่จะรอให้ **syncd** daemon ฟลัชหน้าไปยังดิสก์

คุณลักษณะนี้ทำเพื่อจำกัดจำนวนของหน้าที่สกปรกในหน่วยความจำ ลดโอเวอร์เฮด ระบบ และลดการแบ่งเฟรกเมนต์ดิสก์ให้เหลือน้อยที่สุด Write-behind มีอยู่สองชนิดคือ ตามลำดับและแบบสุ่ม

Write behind ตามลำดับ:

ถ้าทั้ง 4 หน้าของคลัสเตอร์สกปรก ในทันทีที่มีการแก้ไขหน้าใน พาร์ติชันถัดไป 4 หน้าที่สกปรกของคลัสเตอร์จะถูกจัดตารางเวลา ไปยังดิสก์ ถ้าไม่มีคุณลักษณะนี้ หน้าจะยังคงอยู่ในหน่วยความจำจนกว่า **syncd** daemon รัน ซึ่งอาจทำให้เกิดปัญหาคอขวด I/O และการแบ่งเฟรกเมนต์ของไฟล์

โดยค่าดีฟอลต์ ไฟล์ JFS มีการแบ่งพาร์ติชันเป็นพาร์ติชัน 16 KB หรือ 4 หน้า แต่ละพาร์ติชันเหล่านี้เรียกว่า *คลัสเตอร์*

จำนวนคลัสเตอร์ที่ VMM ใช้เป็น threshold สามารถปรับได้ ค่าดีฟอลต์ คือหนึ่งคลัสเตอร์ คุณสามารถถ่วงเวลา write behind ได้โดยการเพิ่มพารามิเตอร์ *numclust* โดยใช้คำสั่ง **ioo -o** *numclust*

สำหรับ Enhanced JFS คำสั่ง **ioo -o** *j2_nPagesPerWriteBehindCluster* ใช้เพื่อระบุจำนวนหน้าที่จะจัดตารางเวลาในแต่ละครั้งแทนที่จะเป็น จำนวนของคลัสเตอร์ จำนวนหน้าดีฟอลต์สำหรับคลัสเตอร์ Enhanced JFS คือ 32 แสดงว่าขนาดดีฟอลต์สำหรับ Enhanced JFS คือ 128 KB

Write behind แบบสุ่ม:

คุณลักษณะ write behind นำเสนอกลไกในลักษณะเดียวกับที่ใช้เมื่อ จำนวนของหน้าที่สกปรกในหน่วยความจำของไฟล์ที่กำหนดเกินกว่า threshold ที่กำหนดไว้ หน้าที่บันทึกในลำดับต่อมาจะถูกจัดตารางเวลาให้บันทึกลงในดิสก์

อาจมีแอ็พพลิเคชันที่ทำ I/O แบบสุ่มจำนวนมาก นั่นคือรูปแบบ I/O ไม่ตรงกับข้อกำหนดของขั้นตอนวิธี write behind ดังนั้น หน้าทั้งหมดจะยังคงอยู่ในหน่วยความจำจนกว่า **syncd** daemon รัน ถ้าแอ็พพลิเคชันมีหน้าที่แก้ไขจำนวนมากในหน่วยความจำ อาจส่งผลให้หน้าจำนวน มากถูกบันทึกลงในดิสก์เมื่อ **syncd** daemon ออกใช้การเรียก **sync( )**

คุณสามารถปรับ threshold โดยใช้คำสั่ง **ioo** ที่มีพารามิเตอร์ JFS *maxrandwrt* ค่าดีฟอลต์คือ 0 ที่บ่งชี้ว่า write behind แบบสุ่ม ปิดใช้งาน การเพิ่มค่านี้ เป็น 128 บ่งชี้ว่า หลังจากหน้าที่อยู่ในหน่วยความจำจำนวน 128 หน้าของไฟล์ สกปรก หน้าที่สกปรกในเวลาต่อมาใดๆ จะถูกจัดตารางเวลาให้บันทึกลงในดิสก์ ชุดแรกของหน้าจะถูกฟลัชหลังจากการเรียก **sync( )**

สำหรับ Enhanced JFS อ็อพชันคำสั่ง **ioo j2_nRandomCluster** (แฟล็ก **-z**) และ **j2_maxRandomWrite** (แฟล็ก **-J**) ใช้เพื่อปรับ write behind แบบสุ่ม ทั้งสองอ็อพชันมีค่าดีฟอลต์เป็น 0 อ็อพชัน **j2_maxRandomWrite** มีฟังก์ชันเหมือนกันสำหรับ enhanced JFS เช่นเดียวกับที่ *maxrandwrt* ดำเนินการ สำหรับ JFS นั่นคือการระบุขีดจำกัดจำนวนของหน้าที่สกปรกต่อไฟล์ ซึ่งสามารถคงอยู่ในหน่วยความจำได้ อ็อพชัน **j2_nRandomCluster** ระบุ จำนวนคลัสเตอร์ระหว่างการบันทึกที่ต่อเนื่องกันสองครั้งซึ่งจะถือว่าเป็น แบบสุ่ม

การปรับผลการทำงานดิสก์ I/O แบบอะซิงโครนัส

ถ้าแอ็พพลิเคชันดำเนินการกับ I/O แบบซิงโครนัส แอ็พพลิเคชันจะรอให้ I/O ดำเนินการจนเสร็จสิ้น ในทางตรงกันข้าม การดำเนินการกับ I/O แบบอะซิงโครนัสจะรันอยู่ในพื้นหลัง และจะไม่บล็อกแอ็พพลิเคชันผู้ใช้ ซึ่งจะช่วยปรับปรุงผลการทำงาน เนื่องจากการดำเนินการ I/O และการประมวลผลแอ็พพลิเคชันสามารถรันอย่างพร้อมเพียงกันได้ แอ็พพลิเคชันจำนวนมาก เช่น ฐานข้อมูลและไฟล์เซิร์ฟเวอร์ จะใช้ประโยชน์ของความสามารถ ในการซ้อนทับการประมวลผลและ I/O

แอ็พพลิเคชันสามารถใช้รูทีนย่อย **aio_read( )**, **aio_write( )** หรือ **lio_listio( )** (หรือชุดสำเนาแบบ 64 บิต) เพื่อดำเนินการกับดิสก์ I/O แบบอะซิงโครนัส การควบคุมจะส่งคืนไปยังแอ็พพลิเคชัน จากรูทีนย่อยในทันทีที่คำร้องขออยู่ในคิว จากนั้น แอ็พพลิเคชันจะยังคงประมวลผลต่อ ขณะที่การดำเนินการของดิสก์จะยังคงดำเนินการอยู่

หากต้องการจัดการกับ I/O แบบอะซิงโครนั้ส คำร้องขอ I/O แบบอะซิงโครนัสแต่ละครั้ง จะมีการควบคุมบล็อกที่สอดคล้องกันในพื้นที่แอดเดรสของแอ็พพลิเคชัน การควบคุมบล็อกนี้จะมีข้อมูลการควบคุม และข้อมูลสถานะสำหรับคำร้องขอ ซึ่งสามารถนำมาใช้ได้อีกครั้ง เมื่อเสร็จสิ้นการดำเนินการ I/O

ผู้ใช้แอ็พพลิเคชันสามารถพิจารณาวิธีที่จะแจ้งเตือน เมื่อการดำเนินการ I/O เสร็จสิ้นด้วยวิธีต่อไปนี้:

- แอ็พพลิเคชันสามารถเลือกสถานะของการดำเนินการ I/O ได้
- ระบบสามารถแจ้งเตือนแอ็พพลิเคชันแบบอะซิงโครนัสได้ เมื่อการดำเนินการ I/O เสร็จสิ้น
- แอ็พพลิเคชันสามารถบล็อกได้จนกว่า การดำเนินการ I/O จะเสร็จสิ้น

แต่ละ I/O ที่ถูกจัดการโดยการประมวลผลเคอร์เนลเดี่ยว หรือ kproc และ kproc ไม่สามารถประมวลผลคำร้องขอเพิ่มเติมได้จากคิวจนกว่า I/O นั้นจะเสร็จสิ้น ค่าดีฟอลต์ของ `minservers` ที่สามารถปรับแต่งได้คือ 3 และ `maxservers` ที่สามารถปรับแต่งได้คือ 30 ค่า `maxservers` คือจำนวนของ I/O kprocs แบบอะซิงโครนัสต่อตัวประมวลผล หากต้องการขอรับจำนวนสูงสุดของ I/O kprocs แบบอะซิงโครนัสที่รันอยู่บนระบบ AIX จะคูณค่า `maxservers` ด้วยจำนวนของตัวประมวลผล ที่รันอยู่ในปัจจุบัน

AIO ที่สามารถปรับแต่งได้ทั้งหมดจะมีค่าปัจจุบัน ค่าดีฟอลต์ ค่าต่ำสุด และค่าสูงสุด ซึ่งสามารถดูได้ด้วยคำสั่ง **ioo** เฉพาะค่าปัจจุบันเท่านั้น ที่สามารถเปลี่ยนได้ด้วยคำสั่ง **ioo** ค่าสามค่าอื่นๆ จะไม่เปลี่ยนแปลง และจะแสดงอยู่เพื่อแจ้งให้ผู้ใช้ถึงข้อจำกัดของการปรับแต่ง ค่าปัจจุบันที่สามารถปรับแต่งได้จะเปลี่ยนแปลงได้ทุกเวลา และสามารถทำให้คงอยู่ระหว่างที่ระบบปฏิบัติการรีสตาร์ทได้ ในระบบที่รันแอ็พพลิเคชันด้วยตนเอง ซึ่งใช้ I/O แบบอะซิงโครนัส ค่าดีฟอลต์จะเป็นค่าที่เพียงพออยู่แล้ว

สิ่งสำคัญคือ ทั้ง `minservers` และ `maxservers` คือตัวประมวลผลที่สามารถปรับแต่งได้ ทั้งความสามารถในการปรับแต่งเหล่านี้จะเป็นแบบไดนามิก แต่การเปลี่ยนแปลงค่าไม่ได้ส่งผลทำให้การซิงโครนัสเปลี่ยนแปลงในจำนวนของเซิร์ฟเวอร์ที่พร้อมใช้งานในระบบ ถ้าค่าของ `minservers` เพิ่มขึ้น จำนวนของเซิร์ฟเวอร์จริงที่เกิดขึ้นโดยตรงตามสัดส่วนของจำนวนของคำร้องขอ I/O ที่พร้อมเพียงกัน หากค่า `minservers` ใหม่ สามารถเข้าถึงได้ ค่านั้นจะกลายเป็นพื้นใหม่ ในทางกลับกัน เมื่อ `minservers` ลดจำนวนลง จำนวนของเซิร์ฟเวอร์ที่พร้อมใช้งานจะตกลงที่ระดับนั้น ซึ่งเป็นระดับที่เซิร์ฟเวอร์ออกเนื่องจากไม่มีกิจกรรม ถ้าจำนวนของคำร้องขอ I/O แบบอะซิงโครนัสมีปริมาณที่สูง ให้เพิ่มค่า `maxservers` ให้เป็นจำนวนของ I/O ที่อาจเป็นได้อย่างพร้อมเพียงกัน ซึ่งตามปกติแล้ว การปล่อยให้พารามิเตอร์ `minservers` มีค่าดีฟอลต์จะเป็นสิ่งที่ดีกว่า เนื่องจากส่วนขยายเคอร์เนล AIO จะสร้างเซิร์ฟเวอร์เพิ่มเติม ถ้าต้องการ

หมายเหตุ: AIO I/O ที่ดำเนินการตามโลจิคัลวอลุ่มหรือไฟล์ที่เปิดอยู่ในโหมด CIO จะไม่ใช้การประมวลผลเซิร์ฟเวอร์ kproc ค่าติดตั้งของ `maxservers` และ `minservers` จะไม่ได้รับผลกระทบในกรณีนี้

ด้วยการมองหาการใช้ประโยชน์จากตัวประมวลผลของเซิร์ฟเวอร์ AIO ถ้าการใช้ประโยชน์ถูกแบ่งออกระหว่างเซิร์ฟเวอร์ทั้งหมด นั่นหมายความว่า เซิร์ฟเวอร์ทั้งบหมดกำลังถูกใช้งาน คุณอาจต้องพยายามเพิ่มเซิร์ฟเวอร์เองในกรณีนี้ หากต้องการดูเซิร์ฟเวอร์ AIO ตามชื่อ ให้รันคำสั่ง **pstat -a** รันคำสั่ง **ps -k** เพื่อดูเซิร์ฟเวอร์ AIO ตามชื่อ `kproc`

สำหรับสภาวะแวดล้อมที่ผลการทำงานของดิสก์ I/O แบบอะซิงโครนัสเกิดเหตุการณ์ที่รุนแรง และวอลุ่มของคำร้องขอมีปริมาณที่สูง แต่คุณไม่มีจำนวนของ I/O ที่เกิดขึ้นพร้อมกันอย่างเหมาะสม ขอแนะนำว่า ให้ตั้งค่า `maxservers` ให้มีค่าอย่างน้อย 10 * (จำนวนของดิสก์ที่เข้าถึงแบบอะซิงโครนัส)

หมายเหตุ: การรีสตาร์ทระบบไม่จำเป็นต้องทำเพื่อให้มีผลต่อการเปลี่ยนแปลง `minservers` หรือ `maxservers` ที่สามารถปรับแต่งได้ ค่า `minservers` ที่สามารถปรับแต่งได้ต้องถูกตั้งค่าที่ระดับ เพื่อทำให้เกิดผลการทำงานสูงสุดระหว่างเวิร์กโหลดโดยเฉลี่ย

ค่าของ `minservers` ที่สามารถปรับแต่งได้ไม่สามารถมีค่าเกินกว่าค่าของ `maxservers` ที่สามารถปรับแต่งได้

สำหรับข้อมูลเพิ่มเตมเกี่ยวกับค่าการปรับ I/O แบบอะซิงโครนัส โปรดดู การเปลี่ยนค่าที่สามารถปรับแต่งได้สำหรับ I/O แบบอะซิงโครนัส

การปรับผลการทำงานของการประสานเวลาไฟล์

มีหลายวิธีในการปรับปรุงการประสานเวลาของไฟล์

ไฟล์ JFS I/O ที่ไม่ได้เป็นลำดับจะเก็บสะสมในหน่วยความจำ จนกว่าตรงตามเงื่อนไขบางอย่าง:

- รายการอิสระหดตัวเป็น *minfree* และการแทนที่ เพจต้องเกิดขึ้น
- **syncd** daemon จะล้างข้อมูลเพจที่ช่วงของการกำหนดตารางเวลา ปกติ
- คำสั่ง **sync** จะถูกออกคำสั่ง
- การสุ่มเขียนด้านหลังจะล้างข้อมูล dirty pages หลังจากที่เข้าใกล้ threshold การสุ่มเขียนด้านหลัง

ถ้าเพจจำนวนมากเก็บสะสมไว้ก่อนที่หนึ่งในเงื่อนไขเหล่านี้เกิดขึ้น ผลคือเมื่อเพจได้รับการล้างข้อมูลโดย **syncd** daemon การล็อก i-node จะได้รับและพักไว้จนกว่าเพจที่สกปรกทั้งหมดถูกเขียน ลงในดิสก์แล้ว ในระหว่างเวลานี้ เธรดที่พยายามเข้าถึงไฟล์นั้น จะถูกบล็อกเนื่องจากการล็อก i-node ไม่พร้อมใช้งาน คำสั่ง **fuser** ยังถูกบล็อกสำหรับไฟล์ เนื่องจากคำสั่ง **fuser** ต้องการล็อก inode เพื่อให้มีข้อมูลที่ต้องการ โปรดจำว่า **syncd** daemon จะล้างข้อมูล dirty pages ของไฟล์ทั้งหมด แต่ล้างข้อมูลหนึ่งไฟล์ต่อครั้ง บนระบบที่มีจำนวนหน่วยความจำขนาดใหญ่ และจำนวนของเพจขนาดหใญ่ที่ได้รับการแก้ไข ช่วงเวลาสูงสุดของ I/O สามารถเกิดขึ้นได้ เมื่อ **syncd** daemon ล้างข้อมูลเพจ

AIX มีอ็อพชันที่สามารถปรับแต่งได้ ซึ่งเรียกว่า **sync_release_ilock** คำสั่ง **ioo** ที่มีอ็อพชัน **-o sync_release_ilock=1** จะอนุญาตให้การล็อก i-node ปลดออก ขณะที่ dirty page ของไฟล์นั้นจะถูกล้างข้อมูล เหตุการณ์นี้อาจส่งผลทำให้เวลาตอบสนองดีขึ้น เมื่อเข้าถึงไฟล์นี้ในระหว่างการเรียก **sync( )**

ผลกระทบจากการบล็อกนี้ยังลดจำนวนลงได้โดยเพิ่มความถี่ของการซิงโครไนซ์ใน **syncd** daemon เปลี่ยน `/sbin/rc.boot` ในตำแหน่ง ที่เริ่มต้น **syncd** daemon ดังนั้น ให้รีบูตระบบ เพื่อให้มีผลบังคับใช้ สำหรับระบบปัจจุบัน ให้หยุดทำงาน **syncd** daemon และรีสตาร์ทด้วยค่าใหม่

วิธีที่สามคือ การปรับลักษณะการทำงานนี้โดยเปิดการสุ่มเขียนด้านหลัง โดยใช้คำสั่ง **ioo** (โปรดดู “การปรับประสิทธิภาพ write behind ตามลำดับและแบบสุ่ม” ในหน้า 270)

Tunables การซิงโครไนซ์ FS2

การดำเนินงาน ซิงโครไนซ์ระบบไฟล์อาจไม่มีประสิทธิภาพในสถานการณ์ต่างๆ ที่มีกิจกรรม I/O แบบสุ่มในไฟล์ขนาดใหญ่ เมื่อเกิดการซิงค์ การอ่านและการเขียนทั้งหมดจากโปรแกรมผู้ใช้ไปยังไฟล์ จะถูกบล็อก เนื่องจากมี หน้าที่สกปรกจำนวนมากอยู่ในไฟล์ จึงต้องใช้เวลานานกว่าจะเขียนลงในดิสก์ได้เสร็จ พารามิเตอร์ JFS2 tunable ต่อไปนี้ สามารถใช้ในสถานการณ์จำลองดังกล่าวได้:

- `j2_syncPageCount`: จำกัดจำนวนของหน้าที่แก้ไข และถูกจัดตารางเวลาให้เขียนโดยซิงค์ในหนึ่งการส่งผ่านสำหรับไฟล์ เมื่อมีการตั้งค่า tunable นี้ ระบบไฟล์เขียนจำนวนหน้าที่ระบุ โดยไม่บล็อก I/O ไปยังส่วนที่เหลือของไฟล์ การเรียกซิงค์วนซ้ำ ในการดำเนินงานเขียนจนกว่ามีการเขียนหน้าที่แก้ไขครบทั้งหมด
- `j2_syncPageLimit`: ยกเลิกพารามิเตอร์ `j2_syncPageCount` เพื่อใช้งานถึง threshold ใช้พารามิเตอร์นี้เพื่อตรวจสอบให้แน่ใจการดำเนินการซิงค์เสร็จสิ้นสำหรับไฟล์ Tunables มีการดูแลรักษา ในวิธีมาตรฐาน

การปรับค่าถูกจัดการโดยใช้คำสั่ง **ioo**

`j2_syncPageCount` และ `j2_syncPageLimit` tunables มีการเพิ่มลงในรายการค่าซึ่งควบคุมโดยคำสั่ง **ioo**

ใช้ แฟล็ก `-o` เพื่อแสดงหรือเปลี่ยนค่า และเมื่อต้องการ ดูอ็อพชันวิธีใช้ ให้ใช้แฟล็ก `-h`

```
# ioo -h j2_syncPageCount
```

ตั้งค่า จำนวนสูงสุดของหน้าที่แก้ไขของไฟล์ซึ่งถูกเขียนลงใน ดิสก์โดยการเรียกระบบซิงค์ในการดำเนินงานเดียว

ค่า: ดีฟอลต์: 0 ช่วง: 0-65536

ชนิด: ไดนามิก

หน่วย: หน้า 4 KB

การปรับ: เมื่อกำลังรันแอ็พพลิเคชันที่ใช้การแคชระบบไฟล์และทำ การเขียนแบบสุ่มจำนวนมาก จำเป็นต้องปรับค่าติดตั้งนี้ เพื่อหลีกเลี่ยงเวลาหน่วงแอ็พพลิเคชันที่ยาวนานในระหว่างการดำเนินงานซิงค์ ค่า ต้องอยู่ในช่วง 256 ถึง 1024 ค่าดีฟอลต์คือ ศูนย์ ซึ่งส่งผลให้ เกิดลักษณะการทำงานซิงค์ปกติของการเขียนหน้าที่สกปรกทั้งหมดใน การเรียกครั้งเดียว หากมีการตั้งค่าต่ำ สำหรับ tunables จะส่งผลให้ เวลาซิงค์นานขึ้นและเวลาหน่วงสั้นลงในเวลาการตอบกลับของแอ็พพลิเคชัน หากมีการตั้งค่าสูง เวลาหน่วงในเวลาการตอบกลับจะนานขึ้นและ เวลาซิงค์สั้นลง

```
# ioo -h j2_syncPageLimit
```

ตั้งค่า จำนวนครั้งสูงสุดที่การเรียกระบบซิงค์ใช้ `j2_syncPageCount` เพื่อจำกัดหน้าซึ่งจะถูกเขียน เพื่อปรับปรุงประสิทธิภาพ ของการดำเนินงานซิงค์

ค่า: ดีฟอลต์: 256 ช่วง: 16-65536

ชนิด: ไดนามิก

หน่วย: ตัวเลข

การปรับ: มีการตั้งค่าเมื่อมีการตั้งค่า `j2_syncPageCount` และต้องเพิ่มขึ้น ถ้าผลจากการเปลี่ยน `j2_syncPageCount` ไม่ เพียงพอ ค่าที่ยอมรับได้อยู่ในช่วงตั้งแต่ 250 ถึง 8000 `j2_syncPageLimit` ไม่มีผลกระทบถ้า `j2_syncPageCount` เป็น 0

ต้องตั้งค่า tunable นี้เมื่อมีการตั้งค่า `j2_syncPageCount` และต้องเพิ่มขึ้นเพื่อให้ผลของการเปลี่ยนแปลง `j2_syncPageCount` ไม่ลดเวลาตอบกลับของแอ็พพลิเคชัน

ค่าต้องอยู่ในช่วง 1 ถึง 8000 ค่าที่ดีที่สุดสำหรับ tunables เหล่านี้ ขึ้นอยู่กับขนาดหน่วยความจำและแบนด์วิธ I/O จุดเริ่มต้น โดยทั่วไป คือการตั้งค่าทั้งสอง tunables นี้เป็น 256

ระยะเวลาการซิงโครไนซ์ JFS2 และการทำงานพร้อมกัน

การซิงโครไนซ์ระบบไฟล์ถูกจัดการโดย sync daemon (**syncd**) ใช้ พารามิเตอร์ที่ปรับค่าได้ JFS2 เพื่ออนุญาตให้ใช้ระบบไฟล์ เพื่อจัดการกับการซิงโครไนซ์โดยใช้ **syncd**

ตัวจัดการซิงค์ JFS2 กระจายกระบวนการ sync เช่น ข้อมูลที่แคชข้อมูลถูกเขียนไปยังดิสก์ในเวลาเดียวกัน การซิงค์ถูกดำเนิน การบนระบบไฟล์เดี่ยวในแต่ละครั้ง ระบบไฟล์แต่ละระบบถูกจัดตาราง เพื่อเริ่มต้นการดำเนินการซิงค์ถัดไปหลังจากการ ดำเนินการก่อนหน้าสิ้นสุดรัน คุณยังสามารถเพิ่ม จำนวนเธรดที่กำลังจัดการกับการดำเนินการซิงค์เมื่อระบบไฟล์จำนวนมาก ต้อง ถูกประมวลผล

ใช้พารามิเตอร์ที่สามารถปรับค่าได้ต่อไปนี้สำหรับการซิงโครไนซ์ระบบไฟล์ JFS2:

j2_syncByVFS:
ระบุใช้ของตัวจัดการซิงค์ JFS2 และตั้งค่าช่วงเวลาระหว่างการดำเนินการซิงค์สำหรับ แต่ละระบบไฟล์

j2_syncConcurrency:
ตั้งค่าจำนวนเธรดที่กำลังจัดการการซิงโครไนซ์ระบบไฟล์ ค่านี้บ่งชี้ จำนวนระบบที่การดำเนินการซิงค์ต้องถูกดำเนิน การอย่างพร้อมเพียงกัน เฉพาะหนึ่งเธรดการซิงค์เท่านั้น ที่ดำเนินการซิงค์บนระบบไฟล์แต่ละระบบ

คำสั่ง **ioo** จัดการกับพารามิเตอร์ที่สามารถปรับค่า I/O ได้ โปรดอ้างถึงเอกสารคู่มือเกี่ยวกับคำสั่ง **ioo** สำหรับข้อมูลเพิ่มเติม

```
# ioo -h j2_syncByVFS
```

วัตถุประสงค์: ระบุจำนวนวินาทีที่รอระหว่างที่ระบบเรียกเพื่อซิงค์ระบบไฟล์ JFS2 ค่านี้แทนที่ค่าที่ระบุโดยคำสั่ง **syncd**

ค่า: ดีฟอลต์: 0 ช่วง: 0–86400

ชนิด: ไดนามิก

หน่วย: วินาที

การปรับ: ค่านี้บ่งชี้จำนวนวินาทีระหว่างการวนซ้ำกระบวนการซิงค์ ตัวจัดการซิงค์ JFS2 จนถึงเวลาที่ระบุโดยพารามิเตอร์ที่สามารถปรับค่าได้ j2_syncByVFS ก่อนที่จะเริ่มต้นการเรียกไปยังรูทีนย่อย syncvfs สำหรับระบบไฟล์ JFS2 ค่า 0 บ่งชี้ว่าการประมวลผล syncd ปกติควรถูกนำมาใช้ ค่าที่ไม่ใช่ศูนย์จะแทนทับเวลา ที่ระบุอยู่บนคำสั่ง **syncd** และเป็นสาเหตุทำให้ตัวจัดการซิงค์ไฟล์ที่ระบุเฉพาะ JFS2 ต้องถูกใช้

```
# ioo -h j2_syncConcurrency
```

วัตถุประสงค์: ตั้งค่าจำนวนเธรดที่ต้องใช้สำหรับการดำเนินการซิงค์ JFS2

ค่า: ค่าดีฟอลต์: 1 ช่วง: 1–128

ชนิด: ไดนามิก

หน่วย: ตัวเลข

การปรับ: sync daemon เริ่มต้นกระบวนการ sync พร้อมกันสำหรับจำนวนของระบบไฟล์ ที่ตั้งค่าโดยพารามิเตอร์ที่สามารถปรับค่าได้ j2_syncConcurrency ค่านี้จะมีผลก็ต่อเมื่อ พารามิเตอร์ที่สามารถปรับค่าได้ j2_syncByVFS ไม่ใช่ศูนย์
ข้อมูลที่เกี่ยวข้อง:
คำสั่ง ioo

การปรับบัฟเฟอร์ของระบบไฟล์

พารามิเตอร์ **ioo** และ **vmstat –v** ต่อไปนี้สามารถมีประโยชน์ในการตรวจพบคอขวดของบัฟเฟอร์ I/O และปรับดิสก์ I/O:

ตัวนับของ I/O เนื่องจากมีบัฟเฟอร์ไม่เพียงพอ

คำสั่ง **vmstat –v** จะแสดงตัวนับ I/O เนื่องจากบัฟเฟอร์ไม่เพียงพอในคอมโพเนนต์เคอร์เนลต่างๆ นี่คือส่วนหนึ่งของตัวอย่างของเอาต์พุต **vmstat –v** :

```
...
              0 paging space I/Os blocked with no psbuf
              2740 filesystem I/Os blocked with no fsbuf
              0 external pager filesystem I/Os blocked with no fsbuf

...
```

ตัวนับ paging space I/Os blocked with no psbuf และ filesystem I/Os blocked with no fsbuf จะเพิ่มขึ้น ไม่ว่า bufstruct จะพร้อมใช้งานและ VMM จะวางเธรดไว้บนรายการที่รอ VMM ตัวนับ external pager filesystem I/Os

`blocked with no fsbuf` จะเพิ่มขึ้นไม่ว่า bufstruct บนระบบไฟล์ JFS ที่ปรับปรุงแล้ว ไม่พร้อมใช้งานก็ตาม

พารามิเตอร์ numfsbufs

ถ้ามีการทำงานอย่างพร้อมเพียงกัน หรือ I/O มีขนาดใหญ่ในระบบไฟล์ หรือถ้ามี I/O ตามลำดับที่มีขนาดใหญ่ ในระบบไฟล์ จึงเป็นไปได้ที่ I/O อาจเกิดคอขวดที่ระดับของระบบไฟล์ ขณะที่รอ bufstructs คุณสามารถเพิ่มจำนวนของ bufstructs ต่อระบบไฟล์ได้ ซึ่งรู้จักกันว่าเป็น *numfsbufs* พร้อมคำสั่ง **ioo** ค่าจะมีผลบังคับใช้เมื่อระบบได้ถูก mount ดังนั้น ถ้าคุณเปลี่ยนค่า คุณต้อง unmount และ mount ระบบไฟล์อีกครั้ง ค่าดีฟอลต์สำหรับ *numfsbufs* คือ 93 bufstructs ต่อระบบไฟล์ในปัจจุบัน

พารามิเตอร์ j2_nBufferPerPagerDevice

หมายเหตุ: เมื่อ vmstat -v แสดงระบบไฟล์ที่ไม่เพียงพอของระบบไฟล์ buffstructs สำหรับ JFS ที่ปรับปรุงแล้ว *j2_dynamicBufferPreallocation* ที่สามารถปรับแต่งได้ควรจะปรับแต่ก่อนที่จะทำการเปลี่ยนพารามิเตอร์ *j2_nBufferPerPagerDevice*

ใน JFS ที่ปรับปรุงแล้ว จำนวนของ bufstructs จะถูกระบุด้วยพารามิเตอร์ *j2_nBufferPerPagerDevice* จำนวนของ bufstructs ที่เป็นค่าดีฟอลต์สำหรับระบบไฟล์ JFS ที่ปรับปรุงแล้วคือ 512 จำนวนของ bufstructs ต่อระบบไฟล์ JFS ที่ปรับปรุงแล้ว (*j2_nBufferPerPagerDevice*) สามารถเพิ่มขึ้นได้โดยใช้คำสั่ง **ioo** ค่ามีผลบังคับใช้ เมื่อระบบไฟล์ถูก mount

พารามิเตอร์ lvm_bufcnt

ถ้าแอ็พพลิเคชันกำลังออกคำสั่ง I/O ที่มีขนาดใหญ่มากแทนการเขียนลงในระบบไฟล์ ชนิดของคอขวดเดียวกันกับระบบไฟล์ สามารถเกิดขึ้นได้ที่เลเยอร์ LVM I/O ขนาดใหญ่มากจะรวมกับอุปกรณ์ I/O ที่เร็วมากจะถูกต้องการเนื่องจากสาเหตุของคอขวด ที่อยู่ในเลเยอร์ LVM แต่ถ้าเหตุการณ์เกิดขึ้น พารามิเตอร์ที่เรียกว่า *lvm_bufcnt* จะสามารถเพิ่มขึ้นโดยคำสั่ง **ioo** เพื่อจัดเตรียมจำนวนของบัฟเฟอร์ "uphysio" ที่มีขนาดใหญ่ ค่าจะมีผลบังคับใช้ในทันที ค่าดีฟอลต์ปัจจุบันคือ 9 "uphysio" บัฟเฟอร์ เนื่องจาก LVM จะแยก I/O ลงใน 128 K สำหรับแต่ละตัว และเนื่องจากค่าดีฟอลต์ของ *lvm_bufcnt* คือ 9 9*128 K สามารถถูกเขียนในหนึ่งครั้ง ถ้า I/O มีขนาด 9*128 K การเพิ่มขึ้นของ *lvm_bufcnt* อาจมีประโยชน์

พารามิเตอร์ pd_npages

พารามิเตอร์ *pd_npages* ระบุจำนวนเพจที่ควรถูกลบออกหนึ่งส่วนจาก RAM เมื่อไฟล์ถูกลบทิ้ง การเปลี่ยนค่านี้อาจมีประโยชน์ต่อแอ็พพลิเคชันแบบเรียลไทม์ ที่ลบไฟล์ ด้วยการลบค่าของพารามิเตอร์ *pd_npages* แอ็พพลิเคชันแบบเรียลไทม์สามารถได้รับเวลาตอบสนองที่ดีขึ้น เนื่องจากจำนวนของเพจเล็กน้อยจะถูกลบออกก่อนที่จะจัดส่งการประมวลผล/เธรด ค่าดีฟอลต์ คือขนาดไฟล์ที่เป็นไปได้ซึ่งมีขนาดใหญ่สุดหารด้วยขนาดของเพจ (ปัจจุบันคือ 4096) ถ้าขนาดของไฟล์ที่ใหญ่ที่สุดคือ 2 GB จากนั้นค่าพารามิเตอร์ *pd_npages* คือ 524288 ตามค่าดีฟอลต์

พารามิเตอร์ v_pinshm

เมื่อคุณตั้งค่าพารามิเตอร์ *v_pinshm* ให้มีค่า 1 นั่นจะเป็นสาเหตุทำให้เพจที่อยู่ในเซ็กเมนต์หน่วยความจำที่แบ่งใช้ถูกตรึงโดย VMM ถ้าแอ็พพลิเคชันซึ่งทำ **shmget( )** ระบุ SHM_PIN เป็นส่วนหนึ่งของแฟล็ก ค่าดีฟอลต์คือ 0

แอ็พพลิเคชันสามารถเลือกที่จะมีความสามารถในการปรับแต่ง ซึ่งระบุแอ็พพลิเคชันที่ควรใช้แฟล็ก SHM_PIN (ตัวอย่างเช่น พารามิเตอร์ *lock_sga* ใน Oracle 8.1.5 และรุ่นต่อมา) การหลีกเลี่ยงการตรึงหน่วยความจำมากเกินไป เนื่องจากในกรณีนี้ไม่มีการแทนที่เพจ ที่สามารถเกิดขึ้นได้ การตรึงจะมีประโยชน์ เนื่องจากจะประหยัดการใช้งานใน I/O แบบอะซิงโครนัสจากเซ็กเมนต์หน่วยความจำที่แบ่งใช้ (ส่วนขยายเคอร์เนล I/O แบบอะซิงโครนัสไม่ต้องการตรึงบัฟเฟอร์)

การปรับ I/O โดยตรง

ข้อได้เปรียบหลักของ I/O แบบโดยตรงคือ ลดการใช้ประโยชน์จาก CPU สำหรับไฟล์ที่อ่านและเขียนโดยกำจัดการทำสำเนาออกจากแคชของไฟล์ VMM ในบัฟเฟอร์ของผู้ใช้

เมื่อคุณกำลังประมวลผล I/O ปกติไปยังไฟล์ I/O จะมาจากบัฟเฟอร์ของแอ็พพลิเคชัน ไปยัง VMM และจากที่นั่นกลับไปยังบัฟเฟอร์ของแอ็พพลิเคชัน เนื้อหาของบัฟเฟอร์จะถูกแคชอยู่ใน RAM ผ่านการใช้หน่วยความจำที่ใช้จริงของ VMM ซึ่งเป็นแคชของบัฟเฟอร์ของไฟล์ ถ้าแคชของไฟล์ที่ฮิตกับอัตราส่วนมีค่าสูง ชนิดของ I/O ที่แคชแล้วจะมีผลกระทบอย่างมากในการปรับปรุงผลการทำงาน I/O ในภาพรวมทั้งหมด แต่แอ็พพลิเคชัน ที่มีแคชที่ฮิตกับอัตราส่วนที่ต่ำหรือแอ็พพลิเคชันที่ทำ I/O ขนาดใหญ่มาก อาจไม่ได้รับผลประโยชน์มากนักจากการใช้ I/O ที่แคชแล้วตามปกติ

ถ้าแคชที่ฮิตกับอัตราส่วนมีค่าต่ำ คำร้องขอการอ่านโดยส่วนใหญ่จะไปยังดิสก์ การเขียนจะเร็วขึ้นพร้อมกับ I/O ที่แคชตามปกติในกรณีส่วนใหญ่ แต่ถ้าไฟล์ถูกเปิดด้วย O_SYNC หรือ O_DSYNC (โปรดดู “การใช้การเรียก sync และ fsync” ในหน้า 231) จากนั้น การเขียนจะไปยังดิสก์ ในกรณีเหล่านี้ I/O โดยตรงสามารถมีแอ็พพลิเคชันที่ได้รับผลประโยชน์เนื่องจากสำเนาข้อมูลถูกจำกัด

ผลประโยชน์อื่นๆ คือ I/O โดยตรงจะอนุญาตให้แอ็พพลิเคชันหลีกเลี่ยงประสิทธิภาพของการทำแคช ของไฟล์อื่นๆ ที่ลดน้อยลง เมื่อไฟล์ถูกอ่านหรือเขียน ไฟล์นั้นจะเพียงพอสำหรับพื้นที่ในหน่วยความจำที่สามารถทำให้ไฟล์อื่นๆ ได้รับการผลักออกจากหน่วยความจำ ถ้าผู้พัฒนาแอ็พพลิเคชันทราบว่า ไฟล์บางไฟล์ มีคุณสมบัติของการใช้ประโยชน์จากแคชที่ต่ำ เฉพาะไฟล์เหล่านี้ จะไม่สามารถเปิดได้ด้วย O_DIRECT

สำหรับ I/O โดยตรงที่ทำงานได้อย่างมีประสิทธิภาพ คำร้องขอ I/O ควรเหมาะสมกับชนิดของระบบไฟล์ ที่กำลังใช้งาน รูทีนย่อย **finfo( )** และ **ffinfo( )** สามารถนำมาใช้เพื่อเคียวรี่อ็อฟเซ็ต ความยาว และข้อกำหนดในการจัดตำแหน่งแอดเดรส สำหรับระบบไฟล์ที่มีขนาดบล็อกที่ไม่เปลี่ยนแปลง ระบบไฟล์ที่แตกแฟรกเมนต์ และระบบไฟล์ที่มีไฟล์ใหญ่ (I/O โดยตรงที่ไม่สนับสนุนระบบไฟล์ที่บีบอัด) ข้อมูลที่เคียวรี่ จะมีโครงสร้าง **diocapbuf** ตามที่กล่าวไว้ใน `/usr/include/sys/finfo.h`

หากต้องการหลีกเลี่ยงปัญหาในเรื่องของความสอดคล้องกัน ถ้ามีการเรียกจำนวนมากเพื่อเปิดไฟล์ และหนึ่งในการเรียกไม่ได้ระบุ O_DIRECT และการเปิดไฟล์อื่นๆ ที่ระบุ O_DIRECT ไฟล์จะยังคงอยู่ในโหมด I/O ที่แคชตามปกติ ในทำนองเดียวกัน ถ้าไฟล์ถูกแม็พลงในหน่วยความจำผ่านการเรียกของระบบ **shmat( )** or **mmap( )** ไฟล์นั้นจะยังคงอยู่ในโหมดที่แคชแล้วตามปกติ ถ้าข้อขัดแย้งล่าสุด การเข้าถึงทางอ้อมจะถูกกำจัด ดังนั้น ระบบไฟล์จะย้ายไฟล์ลงในโหมด I/O โดยตรง (โดยการใช้รูทีนย่อย **close( )**, **munmap( )** หรือ **shmdt( )** อย่างใดอย่างหนึ่ง) การเปลี่ยนแปลงจากโหมดปกติไปเป็นโหมด I/O โดยตรงอาจมีราคาสูง เนื่องจากเพจที่แก้ไขทั้งหมดในหน่วยความจำจะมีการล้างข้อมูลดิสก์ที่จุดนั้น

I/O โดยตรงต้องการวงรอบ CPU ที่น้อยกว่า I/O ปกติ I/O ที่ใช้โดยแอ็พพลิเคชัน ที่ไม่ได้รับผลประโยชน์จากการแคชมากนักที่ถูกจัดเตรียมโดย I/O ปกติสามารถปรับปรุงผลการทำงานได้โดยใช้ I/O โดยตรง ผลประโยชน์ของ I/O โดยตรงนี้ จะมีมากขึ้นในอนาคตตามการเพิ่มความเร็ว CPU ที่ยังคงเพิ่มขึ้น ในความเร็วของหน่วยความจำ

โปรแกรมที่มีผู้สมัครที่ดีสำหรับ I/O โดยตรงจะจำกัด CPU และดำเนินการกับดิสก์ I/O จำนวนมาก แอ็พพลิเคชันในเชิงเทคนิคที่มีลำดับของ I/O ขนาดใหญ่จะมีผู้สมัครที่ดี แอ็พพลิเคชันที่ทำ I/O ขนาดเล็กจำนวนมากจะมีผลประโยชน์จาก ผลการทำงานที่น้อย เนื่องจาก I/O โดยตรงไม่ได้อ่านล่วงหน้าหรือเขียนส่วนหลัง แอ็พพลิเคชันที่มีผลประโยชน์จากการ strip คือผู้สมัครที่ดี

ประสิทธิภาพการอ่าน I/O โดยตรง:

แม้ว่าการใช้ I/O โดยตรงสามารถลดการใช้ CPU ได้ แต่โดยทั่วไป ส่งผลให้เวลาที่ผ่านไปนานขึ้น โดยเฉพาะสำหรับการร้องขอ I/O ขนาดเล็ก เนื่องจากการร้องขอนั้นจะไม่ถูกแคชไว้ในหน่วยความจำ

การอ่าน I/O โดยตรงทำให้เกิดการอ่านแบบซิงโครนัสจากดิสก์ ในขณะที่ตาม นโยบายการแคชปกติ การอ่านอาจดำเนินการจากแคชได้ ลักษณะเช่นนี้อาจส่งผล ให้ประสิทธิภาพด้อยลงถ้าข้อมูลมีโอกาสอยู่ในหน่วยความจำภายใต้นโยบายการแคชปกติ I/O โดยตรงยังเลี่ยงขั้นตอนวิธี VMM read-ahead เนื่องจาก I/Os ไม่ได้เดินทางผ่าน VMM ขั้นตอนวิธี read-ahead มีประโยชน์มาก สำหรับการเข้าถึงไฟล์ในภายหลัง เนื่องจาก VMM สามารถเริ่มต้นการร้องขอดิสก์ และมีหน้าอยู่ในหน่วยความจำอยู่แล้วก่อนหน้าที่แอ็พพลิเคชันร้องขอ หน้า แอ็พพลิเคชันสามารถชดเชยการสูญเสียของ read-ahead นี้ โดยใช้วิธีการอย่างใดอย่างหนึ่งดังต่อไปนี้:

- การออกใช้การร้องขอการอ่านขนาดใหญ่ขึ้น (ต่ำสุด 128 K)
- การออกใช้ I/O read-ahead โดยตรงแบบอะซิงโครนัสโดยใช้หลายเธรด
- การใช้ฟังก์ชันอะซิงโครนัส I/O เช่น **aio_read( )** หรือ **lio_listio( )**

ประสิทธิภาพการบันทึก I/O โดยตรง:

การบันทึก I/O โดยตรงเลี่ยง VMM และไปยังดิสก์โดยตรง ดังนั้นจึงอาจ มีผลกระทบต่อประสิทธิภาพอย่างมาก ใน I/O ที่แคชปกติ การบันทึกสามารถเดินทางไปยังหน่วยความจำ แล้วถูกฟลัชไปยังดิสก์ในภายหลังโดยการดำเนินงาน *sync* หรือ *write behind*

เนื่องจากการบันทึก I/O โดยตรงไม่มีสำเนาอยู่ในหน่วยความจำ เมื่อทำการดำเนินงานซิงค์ จะไม่มีการฟลัชหน้าเหล่านี้ไปยังดิสก์ ดังนั้นจึงลดจำนวนของงานที่ **syncd** daemon ต้องทำ

การจัดระเบียบบันทึกระบบไฟล์และโลจิคัลวอลุ่มบันทึกใหม่

Journaled File System (JFS) และ Enhanced Journaled File System (JFS2) ใช้เทคนิคการเจอร์นัลฐานข้อมูลเพื่อรักษาโครงสร้างระบบไฟล์ ที่สอดคล้องกัน เทคนิคนี้เกี่ยวข้องกับการทำซ้ำธุรกรรมที่ทำใน metadata ระบบไฟล์ลงในบันทึกระบบไฟล์ หมุนเวียน Metadata ระบบไฟล์ ประกอบด้วย superblock, i-nodes, ตัวชี้ข้อมูลทางอ้อม และไดเร็กทอรี

เมื่อหน้าในหน่วยความจำถูกบันทึกจริงลงในดิสก์โดยการเรียก **sync( )** หรือ **fsync( )** commit เร็กคอร์ดจะถูกบันทึกลงในบันทึกเพื่อบ่งชี้ว่า ขณะนี้ข้อมูลอยู่บน ดิสก์ ธุรกรรมบันทึกเกิดขึ้นในสถานการณ์ต่อไปนี้:

- เมื่อไฟล์ถูกสร้างขึ้นหรือลบออก
- เมื่อการเรียก **write( )** เกิดขึ้นสำหรับไฟล์ที่เปิดด้วย O_SYNC และ การบันทึกทำให้เกิดการจัดสรรดิสก์บล็อกใหม่
- เมื่อเรียกรูทีนย่อย **fsync( )** หรือ **sync( )**
- เมื่อการบันทึกทำให้เกิดบล็อกทางอ้อมหรือ double-indirect ที่จะจัดสรร

บันทึกระบบไฟล์ช่วยให้สามารถกู้คืนระบบไฟล์ได้อย่างรวดเร็วและสมบูรณ์ถ้าระบบ ดาวน์ ถ้าแอ็พพลิเคชันกำลังทำซิงโครนัส I/O หรือกำลังสร้างและลบไฟล์จำนวนมาก ในเวลาสั้น อาจมี I/O จำนวนมากไปยังโลจิคัลวอลุ่ม บันทึก ถ้าทั้งโลจิคัลวอลุ่มบันทึกและโลจิคัลวอลุ่มระบบไฟล์ อยู่บนฟิสิคัลดิสก์เดียวกัน อาจก่อให้เกิดปัญหาคอขวด I/O ได้ ข้อแนะนำคือการย้ายอุปกรณ์บันทึกไปยังฟิสิคัลดิสก์อื่น (มีประโยชน์อย่างยิ่งสำหรับเซิร์ฟเวอร์ NFS)

อุปกรณ์แบบ fast-write cached ให้ประสิทธิภาพที่ดีกว่ามากสำหรับโลจิคัล วอลุ่มบันทึก (บันทึกระบบไฟล์หรือบันทึกฐานข้อมูล)

AIX มีอ็อพชัน **mount** ที่เรียกว่า **nointegrity** สำหรับ ระบบไฟล์ JFS ซึ่งจะเลี่ยงการใช้ล็อก JFS สำหรับระบบไฟล์ ที่ติดตั้งด้วยอ็อพชันนี้ การทำเช่นนี้ช่วยให้ประสิทธิภาพดีขึ้น ตราบเท่าที่ผู้ดูแลระบบทราบว่าอาจต้องรันคำสั่ง **fsck** บนระบบไฟล์ ถ้าระบบดาวน์โดยไม่ได้ปิด ตามปกติ

ใช้คำสั่ง **filemon** เพื่อบันทึกข้อมูลเกี่ยวกับ I/Os ลงในบันทึกระบบไฟล์ ถ้าคุณสังเกตว่าระบบไฟล์และอุปกรณ์บันทึกของระบบไฟล์ มีการใช้ประโยชน์อย่างหนักทั้งคู่ อาจดีกว่าที่จะวางแต่ละระบบบน ฟิสิคัลดิสก์แยกต่างหาก (สมมุติว่ามีมากกว่าหนึ่งดิสก์ในกลุ่มวอลุ่ม นั้น)

คุณอาจมีอุปกรณ์บันทึกได้หลายรายการในกลุ่มวอลุ่มหนึ่ง อย่างไรก็ตาม บันทึกสำหรับ ระบบไฟล์ต้องอยู่ในกลุ่มวอลุ่มเดียวกันกับของระบบไฟล์ โลจิคัลวอลุ่มบันทึกหรือโลจิคัลวอลุ่มระบบไฟล์สามารถย้ายไปยังดิสก์อื่นได้ โดยใช้คำสั่ง **migratepv** แม้ในขณะที่ระบบกำลังรัน และถูกใช้งานอยู่

หลักการที่เกี่ยวข้อง:

"ข้อแนะนำเกี่ยวกับกลุ่มวอลุ่ม" ในหน้า 226

ถ้าเป็นไปได้ เพื่อให้ได้การจัดการระบบที่ง่ายขึ้นและประสิทธิภาพที่ดีขึ้น กลุ่มวอลุ่มดีฟอลต์, rootvg, ควรประกอบด้วยฟิสิคัลวอลุ่มที่ ระบบปฏิบัติการมีการติดตั้งเป็นครั้งแรกเท่านั้น

การสร้างบันทึกการทำงานโลจิคัลวอลุ่ม

การวางบันทึกการทำงานโลจิคัลวอลุ่มไว้บนฟิสิคัลวอลุ่มที่แตกต่างจาก โลจิคัลวอลุ่มของระบบไฟล์ที่แอ็คทีฟจะเพิ่มการใช้งานรีซอร์สแบบขนาน คุณสามารถใช้บันทึกการทำงานที่แยกออกจากระบบไฟล์แต่ละระบบ

เมื่อคุณสร้างโลจิคัลวอลุ่ม ผลการทำงานของไดร์ฟ จะแตกต่างกัน ลองพยายามสร้างโลจิคัลวอลุ่มสำหรับระบบไฟล์ที่ใช้งานบ่อยบนไดร์ฟแบบเร็ว (เป็นไปได้สำหรับหนึ่งไดร์ฟที่มีการเขียนแคชแบบเร็ว) ดังต่อไปนี้:

1. สร้างระบบไฟล์ใหม่ที่บันทึกการทำงานโลจิคัลวอลุ่มดังต่อไปนี้:

   ```
   # mklv -t jfslog -y LVname VGname 1 PVname
   ```

   หรือ

   ```
   # mklv -t jfs2log -y LVname VGname 1 PVname
   ```

   หรือ

   ```
   # smitty mklv
   ```

2. จัดรูปแบบบันทึกการทำงานดังต่อไปนี้:

   ```
   # /usr/sbin/logform -V vfstype /dev/LVname
   ```

3. แก้ไข `/etc/filesystems` และ logical volume control block (LVCB) ดังต่อไปนี้:

   ```
   # chfs -a log=/dev/LVname /filesystemname
   ```

4. ถอดออกแล้วจึงประกอบเข้ากับระบบไฟล์

อีกวิธีหนึ่งคือ สร้างบันทึกการทำงานบนวอลุ่มที่แยกออกจากกันนั่นคือ:

- กำหนดกลุ่มวอลุ่มด้วยฟิสิคัลวอลุ่มเดี่ยว
- กำหนดกลุ่มวอลุ่มภายในกลุ่มวอลุ่มใหม่ (นี่เป็นสาเหตุทำให้การจัดสรรกลุ่มวอลุ่ม JFS บันทึกการทำงานบนฟิสิคัลวอลุ่ม)
- เพิ่มฟิสิคัลวอลุ่มที่เหลืออยู่ให้กับกลุ่มวอลุ่ม

- กำหนดระบบไฟล์ที่มีการใช้ประโยชน์สูง (โลจิคัลวอลุ่ม) บนฟิสิคัลวอลุ่มที่เพิ่มขึ้นใหม่

ความก้าวหน้าของดิสก์ I/O

ความก้าวหน้าของดิสก์ I/O มีจุดมุ่งหมายในการปกป้องโปรแกรมที่สร้างจำนวนของเอาต์พุต ที่มีขนาดใหญ่มากจากสิ่งอำนวยความสะดวกสำหรับ I/O ของระบบ และเป็นสาเหตุทำให้เวลาตอบสนองของโปรแกรมที่มีความต้องการน้อยแย่ลง

ความก้าวหน้าของ I/O จะบังคับให้ทำเส้นลายน้ำต่อเซ็กเมนต์ หรือต่อไฟล์ ระดับสูงหรือระดับต่ำสำหรับผลรวมของ I/O ที่คงค้างอยู่ทั้งหมด เมื่อการประมวลผล พยายามบันทึกลงไฟล์ที่มี *เส้นลายน้ำระดับสูง* คงค้างการบันทึกอยู่ การประมวลผลจะวางเพื่อ sleep จนกว่าจะมี I/O เพียงพอต่อการดำเนินการให้เสร็จสิ้น เพื่อให้จำนวนของการบันทึกที่คงค้างอยู่น้อยกว่าหรือเท่ากับ *เส้นลายน้ำระดับต่ำ* ตรรกะของการจัดการคำร้องขอ I/O จะไม่เปลี่ยนแปลง เอาต์พุตจากการประมวลผลวอลุ่มในระดับสูงจะช้าลงในบางส่วน

คุณสามารถตั้งค่าเส้นลายน้ำระดับสูงและระดับต่ำสำหรับระบบที่กว้างขวางด้วยเครื่องมือ SMIT โดยเลือก **สภาวะแวดล้อมของระบบ -> เปลี่ยน / แสดงคุณสมบัติของระบบปฏิบัติการ**(**smitty chgsys**) จากนั้น ป้อนจำนวนของเพจ สำหรับเส้นลายน้ำในระดับสูงและระดับต่ำ หรือสำหรับระบบไฟล์แต่ละระบบโดยใช้อ็อพชันการประกอบเข้าแบบ **maxpout** และ **minpout**

พารามิเตอร์ **maxpout** จะระบุจำนวนของเพจ ที่สามารถกำหนดตารางเวลาในสถานะ I/O ให้กับไฟล์ ก่อนที่เธรดจะหยุดทำงานชั่วคราว พารามิเตอร์ **minpout** จะระบุจำนวนต่ำสุดของเพจที่กำหนดตารางเวลาไว้ ซึ่งเธรดจะเข้าใจได้จากสถานะ การหยุดทำงานชั่วคราว ค่าดีฟอลต์สำหรับ **maxpout** คือ 8193 และค่าดีฟอลต์สำหรับ **minpout** คือ 4096 หากต้องการปิดใช้งานความก้าวหน้า I/O ให้ตั้งค่าพารามิเตอร์ทั้งสองเหล่านั้นให้มีค่าศูนย์

การเปลี่ยนแปลงในค่าระบบที่กว้างขวางของพารามิเตอร์ **maxpout** และ **minpout** จะมีผลกระทบในทันทีโดยไม่ต้องรีบูตระบบ การเปลี่ยนค่าสำหรับพารามิเตอร์ **maxpout** และ **minpout** จะเขียนทับค่าติดตั้งของระบบ อย่างกว้างขวาง คุณสามารถแยกระบบไฟล์ออกจากความก้าวหน้า I/O ได้โดยการประกอบเข้ากับระบบไฟล์ และตั้งค่าพารามิเตอร์ **maxpout** และ **minpout** ให้มีค่า 0 คำสั่งต่อไปนี้คือตัวอย่าง:

```
mount -o minpout=0,maxpout=0 /<file system>
```

การปรับพารามิเตอร์ **maxpout** และ **minpout** อาจช่วยป้องกันเธรดใดๆ ที่ไม่ได้ทำการบันทึกตามลำดับลงในไฟล์จากการครอบครอง รีซอร์สของระบบ

ตารางต่อไปนี้จะสาธิตเวลาตอบสนองของเซสชันของเอดิเตอร์ **vi** สำหรับ IBM eServer™ pSeries รุ่น 7039-651 ที่ปรับแต่งเป็นระบบแบบ 4 ทิศทางพร้อมกับตัวประมวลผลขนาด 1.7 ค่าที่หลากหลายสำหรับพารามิเตอร์ **maxpout** และ **minpout** ขณะที่เขียนลงดิสก์:

ค่าสำหรับ maxpout	ค่าสำหรับ minpout	ขนาดบล็อก dd (10 GB)	เขียน (วินาที)	ทรูพุต (MB/วินาที)	ความคิดเห็น vi
0	0	10000	201	49.8	หลังจาก dd เสร็จสิ้นแล้ว
33	24	10000	420	23.8	ไม่หน่วงเวลา
65	32	10000	291	34.4	ไม่หน่วงเวลา
129	32	10000	312	32.1	ไม่หน่วงเวลา
129	64	10000	266	37.6	ไม่หน่วงเวลา
257	32	10000	316	31.6	ไม่หน่วงเวลา

ค่าสำหรับ maxpout	ค่าสำหรับ minpout	ขนาดบล็อก dd (10 GB)	เขียน (วินาที)	ทรูพุต (MB/วินาที)	ความคิดเห็น vi
257	64	10000	341	29.3	ไม่หน่วงเวลา
257	128	10000	223	44.8	ไม่หน่วงเวลา
513	32	10000	240	41.7	ไม่หน่วงเวลา
513	64	10000	237	42.2	ไม่หน่วงเวลา
513	128	10000	220	45.5	ไม่หน่วงเวลา
513	**256**	**10000**	**206**	**48.5**	**ไม่หน่วงเวลา**
513	384	10000	206	48.5	3 – 6 วินาที
769	512	10000	203	49.3	15–40 วินาที ซึ่งสามารถยาวกว่าเดิมได้
769	640	10000	207	48.3	น้อยกว่า 3 วินาที
1025	32	10000	224	44.6	ไม่หน่วงเวลา
1025	64	10000	214	46.7	ไม่หน่วงเวลา
1025	128	10000	209	47.8	น้อยกว่า 1 วินาที
1025	256	10000	204	49.0	น้อยกว่า 1 วินาที
1025	384	10000	203	49.3	3 วินาที
1025	512	10000	203	49.3	25–40 วินาที ซึ่งสามารถยาวนานกว่าเดิมได้
1025	640	10000	202	49.5	7 – 20 วินาที ซึ่งสามารถยาวกว่าเดิมได้
1025	768	10000	202	49.5	15 – 95 วินาที ซึ่งสามารถยาวนานกว่าเดิมได้
1025	896	10000	209	47.8	3 – 10 วินาที

ช่วงที่ดีที่สุดสำหรับพารามิเตอร์ **maxpout** และ **minpout** จะขึ้นอยู่กับความเร็วของ CPU และระบบ I/O ความก้าวหน้า I/O จะทำงานได้ดีหากค่าของพารามิเตอร์ **maxpout** เท่ากับหรือมากกว่าค่าของพารามิเตอร์ **j2_nPagesPerWriteBehindCluster** ตัวอย่างเช่น ถ้าค่าของพารามิเตอร์ **maxpout** มีค่าเท่ากับ 64 และค่าของพารามิเตอร์ **minpout** มีค่าเท่ากับ 32 ซึ่งจะอยู่ที่ 64 หน้าในสถานะ I/O และ 2 I/O ก่อนที่จะบล็อกการบันทึกถัดไป

การปรับพารามิเตอร์ดีฟอลต์มีดังต่อไปนี้:

พารามิเตอร์	ค่าดีฟอลต์
j2_nPagesPerWriteBehindCluster	32
j2_nBufferPerPagerDevice	512

สำหรับ JFS ที่ปรับปรุงแล้ว คุณสามารถใช้คำสั่ง **ioo -o j2_nPagesPerWriteBehindCluster** เพื่อระบุจำนวนของเพจที่ถูกกำหนดตารางเวลาในหนึ่งครั้ง ค่าดีฟอลต์ของจำนวนเพจสำหรับคลัสเตอร์ JFS ที่ปรับปรุงแล้วคือ 32 ซึ่งหมายความถึง ขนาดดีฟอลต์ของ JFS ที่ปรับปรุงแล้วขนาด 128 KB คุณสามารถใช้คำสั่ง **ioo -o j2_nBufferPerPagerDevice** เพื่อระบุจำนวนของระบบไฟล์ bufstructs ค่าดีฟอลต์คือ 512 สำหรับค่าที่มีผลบังคับใช้ ระบบไฟล์จะต้องประกอบเข้าใหม่อีกครั้ง

สำหรับ JFS ที่ปรับปรุงแล้ว คุณสามารถใช้คำสั่งประกอบเข้า **-o remount** เพื่อเปลี่ยนค่า **maxpout** และ **minpout** ของระบบไฟล์ที่ประกอบแล้ว

ประสิทธิภาพเครือข่าย

AIX นำเสนอโปรโตคอล การสื่อสารที่แตกต่างหลายแบบ ตลอดจนเครื่องมือและวิธีการ มอนิเตอร์และปรับโปรโตคอลดังกล่าว

การปรับประสิทธิภาพ TCP และ UDP

ค่าติดตั้งที่ดีที่สุดของพารามิเตอร์การสื่อสารที่ปรับได้แตกต่างกันไปตาม ชนิดของ LAN ตลอดจนลักษณะ I/O การสื่อสารของระบบหลักและแอ็พพลิเคชันโปรแกรม ส่วนนี้อธิบาย หลักการสากลของการปรับการสื่อสารสำหรับ AIX

ใช้โครงร่างต่อไปนี้สำหรับการตรวจสอบและการปรับการติดตั้งเครือข่าย และเวิร์กโหลด:

- ตรวจสอบให้แน่ใจว่าอะแด็ปเตอร์มีการวางอยู่ในสล็อตที่ถูกต้อง
- ตรวจสอบให้แน่ใจว่าเฟิร์มแวร์ระบบเป็นระดับรีลีสที่ถูกต้อง
- ตรวจสอบให้แน่ใจว่าอะแด็ปเตอร์และสวิตช์เครือข่ายมีความเร็วและโหมด duplex ที่ถูกต้อง
- ตรวจสอบให้แน่ใจว่าเลือกขนาด MTU ที่ถูกต้อง
- ปรับ AIX ที่ปรับได้ สำหรับชนิดเครือข่าย ความเร็ว และโปรโตคอล
- ข้อควรพิจารณาอื่นๆ:
  - อ็อฟชันอะแด็ปเตอร์ offload
    - TCP checksum offload
    - TCP large send หรือการแบ่งเซกเมนต์ใหม่
  - ขัดจังหวะการรวมตัว
  - อินพุตเธรด (Dog threads)

การกำหนดตำแหน่งอะแด็ปเตอร์

ผลการทำงานของเน็ตเวิร์กจะพึ่งพาฮาร์ดแวร์ที่คุณเลือก เช่น ชนิดของอะแด็ปเตอร์ และการกำหนดตำแหน่งอะแด็ปเตอร์ที่อยู่ในเครื่อง

เพื่อมั่นใจว่า ผลการทำงานดีที่สุด คุณต้องวางเน็ตเวิร์กอะแด็ปเตอร์ในสล็อต I/O บัสที่เหมาะสมกับแต่ละอะแด็ปเตอร์มากที่สุด

ขณะที่พยายามกำหนดสล็อต I/O บัสให้เหมาะสมมากที่สุด ให้พิจารณาปัจจัยต่อไปนี้:

- PCI-X เปรียบเทียบกับอะแด็ปเตอร์ PCI
- อะแด็ปเตอร์แบบ 64 บิตเปรียบเทียบกับแบบ 32 บิต
- ความเร็วของนาฬิกาสล็อตบัสที่สนับสนุน (33 MHz, 50/66 MHz หรือ 133 MHz)

แบนด์วิธหรืออัตราข้อมูลของอะแด็ปเตอร์ที่สูงกว่า การกำหนดตำแหน่งสล็อต ที่สำคัญกว่า ตัวอย่างเช่น อะแด็ปเตอร์ PCI-X จะดำเนินการได้ดีที่สุด เมื่อใช้ในสล็อต PCI-X ตามที่อะแด็ปเตอร์รันอยู่ที่ความเร็วของนาฬิกาขนาด 133 MHz บนบัส คุณสามารถวางอะแด็ปเตอร์ PCI-X ในสล็อต PCI แต่อะแด็ปเตอร์เหล่านั้นจะรันช้าลงบนบัสโดยปกติที่ 33 MHz หรือ 66 MHz และจะไม่ดำเนินการพร้อมกันบนเวิร์กโหลดบางส่วน

เช่นเดียวกัน อะแด็ปเตอร์แบบ 64 บิตจะทำงานได้ดีเมื่อติดตั้งอยู่ในสล็อตแบบ 64 บิต คุณสามารถวางอะแด็ปเตอร์แบบ 64 บิตได้ในสล็อตแบบ 32 บิต แต่อะแด็ปเตอร์เหล่านั้นจะดำเนินการด้วย อัตราที่เหมาะสม อะแด็ปเตอร์ MTU ขนาดใหญ่ เช่น กิกะบิตอีเทอร์เน็ตในโหมดของกรอบที่มีขนาดใหญ่ที่สุด จะดำเนินการได้ดีกว่าในสล็อตแบบ 64 บิต

ปัญหาอื่นๆ ที่กระทบกับผลการทำงานคือ จำนวนของอะแด็ปเตอร์ต่อบัส หรือต่อบริดจ์โฮสต์ PCI (PHB) จำนวนของความเร็วอะแด็ปเตอร์สูงสุดอาจถูกจำกัดต่อ PHB ขึ้นอยู่กับแบบจำลองของระบบและชนิดของอะแด็ปเตอร์ คำแนะนำสำหรับการกำหนดตำแหน่งทำให้มั่นใจได้ว่า อะแด็ปเตอร์ถูกกระจายระหว่าง PCI บัสที่หลากหลาย และอาจจำกัดจำนวนของอะแด็ปเตอร์ต่อ PCI บัส โปรดศึกษา *การอ้างอิงการกำหนดตำแหน่งอะแด็ปเตอร์ PCI* สำหรับข้อมูลเพิ่มเติมตามแบบจำลองของเครื่อง และชนิดของอะแด็ปเตอร์

ตารางต่อไปนี้แสดงชนิดของสล็อต PCI และ PCI-X ที่พร้อมใช้งานในเครื่อง IBM System p:

ชนิดของสล็อต	**โค้ดที่ใช้ในหัวข้อนี้**
PCI 32 บิต 33 MHz	A
PCI 32 บิต 50/66 MHz	B
PCI 64 บิต 33 MHz	C
PCI 64 บิต 50/66 MHz	D
PCI-X 32 บิต 33 MHz	E
PCI-X 32 บิต 66 MHz	F
PCI-X 64 บิต 33 MHz	G
PCI-X 64 บิต 66 MHz	H
PCI-X 64 บิต 133 MHz	I

เครื่อง IBM System p5® ที่ใหม่จะมีสล็อต PCI-X เท่านั้น สล็อต PCI-X จะทำงานร่วมกันได้แบบย้อนกลับกับอะแด็ปเตอร์ PCI

ตารางต่อไปนี้แสดงตัวอย่างของอะแด็ปเตอร์ทั่วไป และชนิดของสล็อตที่แนะนำ:

ชนิดของอะแด็ปเตอร์	ชนิดของสล็อตที่ต้องการ (ระดับความสำคัญต่ำสุดถึงระดับความสำคัญสูงสุด)
10/100 Mbps Ethernet PCI Adapter II (อีเทอร์เน็ต 10/100), FC 4962	A-I
IBM PCI 155 Mbps ATM adapter, FC 4953 หรือ 4957	D, H และ I
IBM PCI 622 Mbps MMF ATM adapter, FC 2946	D, G, H และ I
Gigabit Ethernet-SX PCI Adapter, FC 2969	D, G, H และ I
IBM 10/100/1000 Base-T Ethernet PCI Adapter, FC 2975	D, G, H และ I
Gigabit Ethernet-SX PCI-X Adapter (Gigabit Ethernet fibre), FC 5700	G, H และ I
10/100/1000 Base-TX PCI-X Adapter (Gigabit Ethernet), FC 5701	G, H และ I
2-Port Gigabit Ethernet-SX PCI-X Adapter (Gigabit Ethernet fibre), FC 5707	G, H และ I
2-Port 10/100/1000 Base-TX PCI-X Adapter (Gigabit Ethernet), FC 5706	G, H และ I
10 Gigabit-SR Ethernet PCI-X Adapter, FC 5718	I (สล็อต PCI-X 133 เท่านั้น)
10 Gigabit-LR Ethernet PCI-X Adapter, FC 5719	I (สล็อต PCI-X 133 เท่านั้น)

คำสั่ง **lsslot -c pci** และข้อมูลต่อไปนี้:

- ชนิด PCI ของสล็อต
- ความเร็วบัส
- อุปกรณ์ที่อยู่ในสล็อต

ต่อไปนี้คือตัวอย่างของคำสั่ง **lsslot -c pci** สำหรับระบบ p615 แบบ 2 ทางพร้อมกับสล็อตภายในหกช่อง:

```
# lsslot -c pci
# Slot      Description                       Device(s)
U0.1-P1-I1  PCI-X capable, 64 bit, 133 MHz slot  fcs0
U0.1-P1-I2  PCI-X capable, 32 bit, 66 MHz slot   Empty
U0.1-P1-I3  PCI-X capable, 32 bit, 66 MHz slot   Empty
U0.1-P1-I4  PCI-X capable, 64 bit, 133 MHz slot  fcs1
U0.1-P1-I5  PCI-X capable, 64 bit, 133 MHz slot  ent0
U0.1-P1-I6  PCI-X capable, 64 bit, 133 MHz slot  ent2
```

สำหรับอะแด็ปเตอร์กิกะบิตอีเทอร์เน็ต ข้อมูลสถิติที่ระบุเฉพาะอะแด็ปเตอร์ที่ส่วนท้ายของเอาต์พุตคำสั่ง **entstat -d en***[interface-number]* หรือเอาต์พุตคำสั่ง **netstat -v** จะแสดงชนิดของ PCI บัสและความเร็วบัสของอะแด็ปเตอร์ ต่อไปนี้คือตัวอย่างเอาต์พุตของคำสั่ง **netstat -v**:

```
# netstat -v

10/100/1000 Base-TX PCI-X Adapter (14106902) Specific Statistics:
--------------------------------------------------------------------
Link Status: Up
Media Speed Selected: Auto negotiation
Media Speed Running: 1000 Mbps Full Duplex
PCI Mode: PCI-X (100-133)
PCI Bus Width: 64 bit
```

เฟิร์มแวร์ระบบ

เฟิร์มแวร์ระบบรับผิดชอบในการตั้งค่าคอนฟิกพารามิเตอร์สำคัญหลายตัว บนแต่ละอะแด็ปเตอร์ PCI และการตั้งค่าคอนฟิกอ็อพชันใน I/O chips บน I/O และ PCI บัสต่างๆ ในระบบ

ในบางกรณี เฟิร์มแวร์ตั้งค่าพารามิเตอร์เฉพาะให้กับอะแด็ปเตอร์เฉพาะ ตัวอย่างเช่น PCI Latency Timer และ Cache Line Size, และสำหรับอะแด็ปเตอร์ PCI-X ค่า Maximum Memory Read Byte Count (MMRBC) พารามิเตอร์เหล่านี้มีความสำคัญต่อการได้รับประสิทธิภาพที่ดีจากอะแด็ปเตอร์ ถ้าไม่ได้ตั้งค่าพารามิเตอร์เหล่านี้ อย่างถูกต้องเนื่องจากเฟิร์มแวร์ระดับต่ำจะไม่สามารถได้รับประสิทธิภาพสูงสุด โดยการปรับซอฟต์แวร์เพียงอย่างเดียว ตรวจสอบให้แน่ใจว่าคุณอัพเดตเฟิร์มแวร์บนระบบเก่าก่อนที่จะเพิ่มอะแด็ปเตอร์ใหม่ลงในระบบ

คุณสามารถเห็นทั้งแพล็ตฟอร์มและระดับเฟิร์มแวร์ของระบบด้วยคำสั่ง **lscfg -vp|grep -p " ROM"** ดังในตัวอย่างต่อไปนี้:

```
lscfg -vp|grep -p " ROM"

     ...lines omitted...

     System Firmware:
        ROM Level (alterable).......M2P030828
        Version.....................RS6K
        System Info Specific.(YL)...U0.1-P1/Y1
      Physical Location: U0.1-P1/Y1

      SPCN firmware:
        ROM Level (alterable).......0000CMD02252
        Version.....................RS6K
        System Info Specific.(YL)...U0.1-P1/Y3
      Physical Location: U0.1-P1/Y3

      SPCN firmware:
        ROM Level (alterable).......0000CMD02252
        Version.....................RS6K
        System Info Specific.(YL)...U0.2-P1/Y3
      Physical Location: U0.2-P1/Y3

      Platform Firmware:
        ROM Level (alterable).......MM030829
        Version.....................RS6K
        System Info Specific.(YL)...U0.1-P1/Y2
      Physical Location: U0.1-P1/Y2
```

คำแนะนำเกี่ยวกับผลการทำงานของอะแด็ปเตอร์

ระบบปฏิบัติการ AIX มีแนวทางให้ส่วนหนึ่งเพื่อเพิ่มประสิทธิภาพการทำงานอะแด็ปเตอร์สูงสุด

อัตราข้อมูลปริมาณการใช้งานของผู้ใช้สามารถขอรับได้จากโปรแกรมแบบอิงซ็อกเก็ต สำหรับแอ็พพลิเคชันที่เป็นข้อมูลสตรีมผ่านการเชื่อมต่อ TCP ตัวอย่าง เช่น โปรแกรมหนึ่งเรียกใช้ **send( )** และตัวรับเรียกใช้ **recv( )** อัตราคือฟังก์ชันของอัตราบิตเครือข่าย, ขนาด Media Transmission Unit (MTU) (ขนาดเฟรม), โอเวอร์เฮดระดับฟิสิคัล เช่น ช่องว่างระหว่างเฟรม และบิตส่วนนำ, ส่วนหัว DataLink และ ส่วนหัว TCP/IP และตัวประมวลผลความเร็ว Gigahertz อัตราเหล่านี้เป็นค่า กรณีดีที่สุดสำหรับ LAN เดียว และอาจต่ำกว่านี้ถ้าผ่านเราเตอร์ หรือฮอปเครือข่าย หรือรีโมตลิงก์เพิ่มขึ้น

อัตราการสตรีม TCP ทางเดียว (simplex) คืออัตราที่สามารถ เห็นโดยเวิร์กโหลดเช่น FTP ที่ส่งข้อมูลจากระบบ A ไปยัง B ในการทดสอบระหว่างหน่วยความจำกับหน่วยความจำ โปรดดู "คำสั่ง ftp" ในหน้า 326 ฟังก์ชันสื่อบันทึกแบบ Full duplex ดีกว่าสื่อบันทึกแบบ half duplex เนื่องจาก การตอบรับ TCP สามารถส่งกลับมาโดยไม่ต้องมีการยืนยันสำหรับ การเชื่อมต่อเดียวกันที่แพ็กเก็ตข้อมูลกำลังรับส่ง

หมายเหตุ: ในตารางต่อไปนี้ ค่า **Raw bit Rate** คืออัตราบิต สื่อบันทึกฟิสิคัล และไม่แสดงถึงโอเวอร์เฮดสื่อบันทึกฟิสิคัล เช่น ช่องว่าง Inter-Frame, บิตส่วนนำ, เซลล์โอเวอร์เฮด (สำหรับ ATM), ส่วนหัวและส่วนท้าย DataLink ค่าเหล่านี้ลดอัตราบิตที่ใช้ได้ ที่มีผลของการเชื่อมต่อ

ตารางต่อไปนี้แสดงรายการความเร็วเพย์โหลดเครือข่ายสูงสุดและ อัตราการสตรีม TCP ทางเดียว (simplex):

ตารางที่ 5. ความเร็วปริมาณการใช้เน็ตเวิร์กสูงสุดเมื่อเปรียบเทียบกับอัตราการสตรีม TCP แบบสื่อสารทางเดียว

ชนิดเครือข่าย	Raw bit rate (Mbits)	Payload rate (Mb)	Payload rate (MB)
อีเทอร์เน็ตขนาด 10 Mb แบบ Half Duplex	10	6	0.7
อีเทอร์เน็ตขนาด 10 Mb แบบ Full Duplex	10 (20 Mb แบบ full duplex)	9.48	1.13
อีเทอร์เน็ตขนาด 100 Mb แบบ Half Duplex	100	62	7.3
อีเทอร์เน็ตขนาด 100 Mb แบบ Full Duplex	100 (200 Mb แบบ full duplex)	94.8	11.3
อีเทอร์เน็ตขนาด 1000 Mb แบบ Full Duplex, MTU 1500	1000 (2000 Mb แบบ full duplex)	948	113.0
อีเทอร์เน็ตขนาด 1000 Mb แบบ Full Duplex, MTU 9000	1000 (2000 Mb แบบ full duplex)	989	117.9
10 Gb Ethernet, Full Duplex, MTU 1500 (ที่เปิดใช้งาน RFC1323)	10000	7200 (peak 9415)[1]	858 (peak 1122)[1]
10 Gb Ethernet, Full Duplex, MTU 9000 (ที่เปิดใช้งาน RFC1323)	10000	9631 (peak 9891)[1]	1148 (peak 1179)[1]
FDDI, MTU 4352 (ดีฟอลต์)	100	92	11.0
Asynchronous Transfer Mode (ATM) 155, MTU 1500	155	125	14.9
ATM 155, MTU 9180 (ดีฟอลต์)	155	133	15.9
ATM 622, MTU 1500	622	364	43.4
ATM 622, MTU 9180 (ดีฟอลต์)	622	534	63.6

[1] ค่าในตารางระบุอัตราสำหรับ อะแด็ปเตอร์เฉพาะบนพาร์ติชันเฉพาะ ประสิทธิภาพการทำงานสำหรับอะแด็ปเตอร์ 10 Gigabit Ethernet ใน virtual Ethernet Adapter (ใน VIOS) หรือ Shared Ethernet Adapters (SEA) หรือสำหรับพาร์ติชันแบ่งใช้ (LPAR ที่แบ่งใช้) ไม่ แสดงในตารางเนื่องจากประสิทธิภาพการทำงานได้รับผลจาก ตัวแปรอื่น และการปรับค่าที่อยู่นอกขอบเขตของตารางนี้

เวิร์กโหลดการสตรีม TCP แบบสองทิศทาง (duplex) มีข้อมูลสตรีม แบบสองทิศทาง ตัวอย่างเช่น การรันคำสั่ง **ftp** จากระบบ A ไประบบ B และอีกอินสแตนซ์ของคำสั่ง **ftp** จาก B ไป A พร้อมกัน ถูกพิจารณาเป็นการสตรีม duplex TCP ชนิดเหล่านี้ของเวิร์ก

โหลดจะใช้ประโยชน์ของสื่อบันทึกแบบ full duplex ที่สามารถส่งและรับข้อมูลในเวลาเดียวกันได้ สื่อบันทึกบางอย่าง เช่น Fibre-Distributed Data Interface (FDDI) หรือ Ethernet ในโหมด Half Duplex ไม่สามารถส่ง และรับข้อมูลพร้อมกัน และดำเนินการได้ไม่ดีเมื่อรัน เวิร์กโหลด duplex เวิร์กโหลด Duplex ไม่ปรับสเกลอัตราเป็นสองเท่า ของเวิร์กโหลดทางเดียวเนื่องจากแพ็กเก็ตการตอบกลับของ TCP ที่ มาจากผู้รับต้องยืนยันด้วยแพ็กเก็ตข้อมูลที่ กำลังส่งในทางเดียวกัน ตารางต่อไปนี้แสดงอัตราการสตรีม TCP แบบสองทิศทาง (duplex):

ตารางที่ 6. ความเร็วปริมาณการใช้เน็ตเวิร์กสูงสุด เมื่อเปรียบเทียบกับอัตราการสตรีม TCP แบบ duplex

ชนิดเครือข่าย	Raw bit rate (Mbits)	Payload rate (Mb)	Payload rate (MB)
อีเทอร์เน็ตขนาด 10 Mb แบบ Half Duplex	10	5.8	0.7
อีเทอร์เน็ตขนาด 10 Mb แบบ Full Duplex	10 (20 Mb แบบ full duplex)	18	2.2
อีเทอร์เน็ตขนาด 100 Mb แบบ Half Duplex	100	58	7.0
อีเทอร์เน็ตขนาด 100 Mb แบบ Full Duplex	100 (200 Mb แบบ full duplex)	177	21.1
อีเทอร์เน็ตขนาด 1000 Mb แบบ Full Duplex, MTU 1500	1000 (2000 Mb แบบ full duplex)	1811 (1667 peak) [1]	215 (222 peak) [1]
อีเทอร์เน็ตขนาด 1000 Mb แบบ Full Duplex, MTU 9000	1000 (2000 Mb แบบ full duplex)	1936 (1938 peak) [1]	231 (231 peak) [1]
อีเทอร์เน็ตขนาด 10 Gb แบบ Full Duplex, MTU 1500	10000 (20000 Mb แบบ full duplex)	14400 (18448 peak) [1]	1716 (2200 peak) [1]
อีเทอร์เน็ตขนาด 10 Gb แบบ Full Duplex, MTU 9000	10000 (20000 Mb แบบ full duplex)	18000 (19555 peak) [1]	2162 (2331 peak) [1]
FDDI, MTU 4352 (ดีฟอลต์)	100	97	11.6
ATM 155, MTU 1500	155 (310 Mb แบบ full duplex)	180	21.5
ATM 155, MTU 9180 (ดีฟอลต์)	155 (310 Mb แบบ full duplex)	236	28.2
ATM 622, MTU 1500	622 (1244 Mb แบบ full duplex)	476	56.7
ATM 622, MTU 9180 (ดีฟอลต์)	622 (1244 Mb แบบ full duplex)	884	105

[1] ค่าในตารางระบุอัตราสำหรับ อะแด็ปเตอร์เฉพาะบนพาร์ติชันเฉพาะ ประสิทธิภาพการทำงานสำหรับอะแด็ปเตอร์ 10 Gigabit Ethernet ใน virtual Ethernet Adapter (ใน VIOS) หรือ Shared Ethernet Adapters (SEA) หรือสำหรับพาร์ติชันแบ่งใช้ (LPAR ที่แบ่งใช้) ไม่ แสดงในตารางเนื่องจากประสิทธิภาพการทำงานได้รับผลจาก ตัวแปรอื่น และการปรับค่าที่อยู่นอกขอบเขตของตารางนี้

หมายเหตุ:

1. หมายเลขพีคแสดงปริมาณงานในกรณีที่ดีที่สุดที่มีหลายเซสชัน TCP ที่กำลังรันในแต่ละทิศทาง อัตราอื่นๆ สำหรับเซสชัน TCP แบบเดี่ยว อัตราเซสชันเดียวขึ้นอยู่กับความถี่ ตัวประมวลผล อะแด็ปเตอร์ที่เจาะจง และชนิดสล็อต PCI ที่ใช้
2. อัตรา duplex ของ 1000 Mbit Ethernet (Gigabit Ethernet) สำหรับอะแด็ปเตอร์ PCI-eXtended (PCI-X) หรืออะแด็ปเตอร์สล็อต peripheral component interconnect express (PCIe) ผลการทำงานจะต่ำกว่าเวิร์กโหลดแบบ duplex สำหรับอะแด็ปเตอร์ PCI หรืออะแด็ปเตอร์ PCI-X ในสล็อต PCI อัตรา 10 Gb Ethernet ที่ ถูกระบุนั้นสำหรับอะแด็ปเตอร์ PCIe เท่านั้น

3. อัตราข้อมูลสำหรับ TCP/IP ที่ใช้อินเตอร์เน็ตโปรโตคอลเวอร์ชัน 4 (IPv4) อ็อพชัน **RFC1323** ถูกเปิดใช้งานสำหรับอะแด็ปเตอร์ ต่อไปนี้:
   - อะแด็ปเตอร์ที่มีขนาด MTU เป็น 4096 และใหญ่กว่า
   - อะแด็ปเตอร์ 10 Gigabit Ethernet หรือเร็วกว่า
4. คอลัมน์ Payload rate (Mb) เป็นหน่วยเมกะบิตต่อวินาที โดย 1 Mb คือ 1,000,000 บิต คอลัมน์ Payload rate (MB) เป็นหน่วยเมกะไบต์ ต่อวินาที โดย 1 MB คือ 1,048,576 ไบต์

ค่าติดตั้งอะแด็ปเตอร์และอุปกรณ์

อ็อพชันอะแด็ปเตอร์และอุปกรณ์ทั้งหลายมีความสำคัญสำหรับทั้งการดำเนินการที่ถูกต้อง และผลการทำงานที่ดีที่สุด

โดยปกติแล้ว อุปกรณ์ AIX มีค่าดีฟอลต์ที่ควรทำงานได้ดีสำหรับการติดตั้งส่วนใหญ่ ดังนั้น ค่าอุปกรณ์จึงไม่จำเป็นต้องเปลี่ยนแปลง อย่างไรก็ตาม ในบางบริษัทมีนโยบายที่ต้องการค่าติดตั้งเน็ตเวิร์กที่ระบุเฉพาะ หรืออุปกรณ์เน็ตเวิร์กบางตัวอาจต้องการค่าติดตั้งบางอย่างเหล่านี้เพื่อทำการเปลี่ยนแปลง

ความเร็วอะแด็ปเตอร์และค่าติดตั้งในโหมด duplex

ค่ากำหนดดีฟอลต์สำหรับ AIX Auto_Negotiation ซึ่งจัดการกับความเร็วและค่าติดตั้ง duplex สำหรับอัตราข้อมูลสูงสุด ที่เป็นไปได้ สำหรับโหมด Auto_Negotiation ที่ทำงานได้อย่างถูกต้อง คุณต้องปรับแต่งจุดปลายอื่นๆ (สวิตช์) สำหรับโหมด Auto_Negotiation

คุณสามารถปรับแต่งอะแด็ปเตอร์อีเทอร์เน็ตสำหรับโหมดต่อไปนี้:

- 10_Half_Duplex
- 10_Full_Duplex
- 100_Half_Duplex
- 100_Full_Duplex
- Auto_Negotiation

สิ่งสำคัญคือ คุณปรับแต่งทั้งอะแด็ปเตอร์และจุดปลายอื่นๆ ของสายเคเบิลด้วยวิธีเดียวกัน (โดยปกติคือสวิตช์อีเทอร์เน็ต หรืออะแด็ปเตอร์ตัวอื่นหากรันอยู่ในคอนฟิกูเรชันแบบจุดต่อจุด โดยไม่มีสวิตช์อีเทอร์เน็ต) ถ้าจุดปลายหนึ่งบจุด ถูกตั้งค่าความเร็วที่ระบุเฉพาะและโหมด duplex ด้วยตนเอง จุดปลายอื่นๆ ควรตั้งค่าความเร็วและโหมด duplex ที่เหมือนกัน การตั้งค่าหนึ่งจุดปลายและHaving ค่าติดตั้งอื่นๆ ในโหมด Auto_Negotiation ด้วยตนเองจะส่งผลทำให้เกิดปัญหา ซึ่งให้ลิงก์ทำงานได้ช้าลง

การดำเนินการที่ดีที่สุดคือ การใช้โหมด Auto_Negotiation เมื่อเป็นไปได้ เนื่องจากเป็นค่ากำหนดดีฟอลต์สำหรับสวิตช์อีเทอร์เน็ตส่วนใหญ่ อย่างไรก็ตาม สวิตช์อีเทอร์เน็ต 10/100 บางตัวไม่สนับสนุนโหมด Auto_Negotiation ของโหมด duplex ชนิดของสวิตช์เหล่านี้ต้องการให้คุณตั้งค่าจุดปลายทั้งสองด้านให้มีความเร็วและโหมด duplex ตามความต้องการ

หมายเหตุ: อะแด็ปเตอร์ 10 Gigabit Ethernet ไม่สนับสนุนโหมด Auto_Negotiation เนื่องจากทำงานที่ความเร็วเดียวสำหรับสื่อบันทึกไฟเบอร์ SR และ LR เท่านั้น

คุณต้องใช้คำสั่งที่เป็นค่าเฉพาะกับแต่ละสวิตช์อีเทอร์เน็ตเพื่อแสดงค่าติดตั้งพอร์ต และเปลี่ยนความเร็วพอร์ตและค่าติดตั้งโหมด duplex ภายในสวิตช์อีเทอร์เน็ต โปรดอ้างถึงเอกสารคู่มือสวิตช์ของผู้ขายของคุณ สำหรับคำสั่ง

สำหรับ AIX คุณสามารถใช้คำสั่ง **smitty devices** เพื่อเปลี่ยนค่าติดตั้งของอะแด็ปเตอร์ คุณสามารถใช้คำสั่ง **netstat –v** หรือคำสั่ง **entstat –d en***X* โดยที่ *X* คือหมายเลขอีเทอร์เน็ตอินเตอร์เฟส เพื่อแสดงค่าติดตั้งและโหมดที่ถูกจัดการ ต่อไปนี้คือส่วนหนึ่งของตัวอย่างของเอาต์พุตคำสั่ง **entstat –d en3**:

```
10/100/1000 Base-TX PCI-X Adapter (14106902) Specific Statistics:
---------------------------------------------------------------------
Link Status: Up
Media Speed Selected: Auto negotiation
Media Speed Running: 1000 Mbps Full Duplex
```

ค่าติดตั้งอะแด็ปเตอร์ MTU

อุปกรณ์ทั้งหมดบนฟิสิคัลเน็ตเวิร์กเดียวกัน หรือโลจิคัลเน็ตเวิร์ก หากใช้การแท็ก VLAN ต้องมีขนาด Media Transmission Unit (MTU) ที่เหมือนกัน นี่คือขนาดสูงสุดของกรอบ (หรือแพ็กเก็ต) ที่สามารถส่งออกบนสายสื่อสารได้

เน็ตเวิร์กอะแด็ปเตอร์ที่หลากหลายจะสนับสนุนขนาดของ MTU ที่ต่างกัน ดังนั้น ควรตรวจสอบให้แน่ใจว่า คุณใช้ขนาด MTU ที่เหมือนกันสำหรับอุปกรณ์ทั้งหมดบนเน็ตเวิร์ก ตัวอย่างเช่น คุณไม่สามารถมีอะแด็ปเตอร์กิกะบิตอีเทอร์เน็ตซึ่งใช้โหมดของกรอบขนาดใหญ่ที่มีขนาด MTU 9000 ไบต์ ขณะที่อะแด็ปเตอร์อื่นๆ บนเน็ตเวิร์กใช้ค่าดีฟอลต์ของขนาด MTU 1500 ไบต์ อะแด็ปเตอร์อีเทอร์เน็ต 10/100 ไม่สนับสนุนโหมดของกรอบขนาดใหญ่ ดังนั้น อะแด็ปเตอร์เหล่านั้นจะไม่ทำงานร่วมกับอ็อพชันกิกะบิตอีเทอร์เน็ตนี้ คุณยังต้องปรับแต่งสวิตช์อีเทอร์เน็ต เพื่อใช้กรอบขนาดใหญ่ หากกรอบขนาดใหญ่ได้รับการสนับสนุนบน สวิตช์อีเทอร์เน็ตของคุณ

ซึ่งเป็นสิ่งสำคัญในการเลือกขนาด MTU ของอะแด็ปเตอร์ไว้แต่เนิ่นๆ ในการติดตั้งเน็ตเวิร์ก ดังนั้น คุณจึงสามารถปรับแต่งอุปกรณ์และสวิตช์ทั้งหมดได้อย่างถูกต้อง และ อ็อพชันการปรับ AIX จำนวนมาก จะขึ้นอยู่กับขนาด MTU ที่เลือกไว้

ผลกระทบต่อประสิทธิภาพการทำงานจากขนาด MTU

ขนาด MTU ของเครือข่ายมีผลกระทบอย่างมากต่อประสิทธิภาพการทำงาน

การใช้ MTU ขนาดใหญ่ช่วยให้ระบบปฏิบัติการสามารถส่งแพ็กเก็ตขนาดใหญ่ ในจำนวนที่น้อยลงเพื่อให้ได้ผลผลิตเครือข่ายจำนวนเดียวกัน แพ็กเก็ตขนาดใหญ่ขึ้น ช่วยลดการประมวลผลที่ต้องใช้ในระบบปฏิบัติการได้เป็นอย่างมาก โดยสมมุติว่า เวิร์กโฟลว์อนุญาตให้ส่งข้อความขนาดใหญ่ได้ ถ้าเวิร์กโหลดกำลังส่ง ข้อความขนาดเล็กเพียงอย่างเดียว ขนาด MTU ที่ใหญ่ขึ้นจะไม่ช่วยอะไร

เมื่อเป็นไปได้ ให้ใช้ขนาด MTU ใหญ่ที่สุดที่อะแด็ปเตอร์และเครือข่ายสนับสนุน ตัวอย่างเช่น สำหรับอะแด็ปเตอร์ Asynchronous Transfer Mode (ATM) ขนาด MTU ดีฟอลต์ 9180 มีประสิทธิภาพมากกว่าการใช้ MTU ขนาด 1500 ไบต์ (โดยปกติแล้วใช้โดย LAN Emulation) สำหรับอีเทอร์เน็ตหนึ่งและ 10 กิกะบิต ถ้าเครื่องทั้งหมดบนเครือข่ายมีอะแด็ปเตอร์อีเทอร์เน็ตหนึ่งกิกะบิตและไม่มี อะแด็ปเตอร์ 10/100 บนเครือข่าย สิ่งที่ดีที่สุดคือการใช้โหมด jumbo frame ตัวอย่างเช่น โดยทั่วไป การเชื่อมต่อระหว่างเซิร์ฟเวอร์ภายในคอมพิวเตอร์แล็บ สามารถทำโดยใช้ jumbo frames

การเลือกโหมด jumbo frame บนกิกะบิตอีเทอร์เน็ต

คุณต้องเลือกโหมด jumbo frame เป็นอ็อพชันอุปกรณ์

การพยายามเปลี่ยนขนาด MTU โดยใช้คำสั่ง **ifconfig** ไม่ทำงาน ใช้ SMIT เพื่อแสดงค่าติดตั้งอะแด็ปเตอร์ด้วยขั้นตอน ต่อไปนี้:

1. เลือก `Devices`
2. เลือก `Communications`
3. เลือก `Adapter Type`

4. เลือก Change/Show Characteristics of an Ethernet Adapter
5. เปลี่ยนอ็อพชัน Transmit Jumbo Frames จาก no เป็น yes

จอภาพ SMIT มีลักษณะคล้ายข้อมูลต่อไปนี้:

```
                                Change/Show Characteristics of an Ethernet Adapter

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                [Entry Fields]
  Ethernet Adapter                               ent0
  Description                                    10/100/1000 Base-TX PCI-X Adapter (14106902)
  Status                                         Available
  Location                                       1H-08
  Receive descriptor queue size                 [1024]                                            +#
  Transmit descriptor queue size                [512]                                             +#
  Software transmit queue size                  [8192]                                            +#
  Transmit jumbo frames                          yes                                               +
  Enable hardware transmit TCP resegmentation    yes                                               +
  Enable hardware transmit and receive checksum  yes                                               +
  Media Speed                                    Auto_Negotiation                                  +
  Enable ALTERNATE ETHERNET address              no                                                +
  ALTERNATE ETHERNET address                    [0x000000000000]                                   +
  Apply change to DATABASE only                  no                                                +

F1=Help                     F2=Refresh                  F3=Cancel                   F4=List
Esc+5=Reset                 Esc+6=Command               Esc+7=Edit                  Esc+8=Image
Esc+9=Shell                 Esc+0=Exit                  Enter=Do
```

การปรับประสิทธิภาพเครือข่ายโดยใช้คำสั่ง no

อ็อพชันเครือข่ายหรือคำสั่ง **no** แสดง เปลี่ยน และจัดการอ็อพชันเครือข่ายสากล

อ็อพชันคำสั่ง **no** ที่ใช้เพื่อเปลี่ยนพารามิเตอร์ การปรับมีดังต่อไปนี้:

อ็อพชัน คำนิยาม

-a พิมพ์ค่าที่ปรับได้ทั้งหมดและค่าปัจจุบันของค่าเหล่านั้น

-d [*ค่าที่ปรับได้*]
ตั้งค่าที่ปรับได้ซึ่งระบุกลับไปเป็นค่าดีฟอลต์

-D ตั้งค่าอ็อพชันทั้งหมดกลับไปเป็นค่าดีฟอลต์

-o *ค่าที่ปรับได้*=[**ค่าใหม่**]
แสดงค่าหรือตั้งค่าที่ปรับได้ซึ่งระบุเป็นค่าใหม่ที่ ระบุ

-h [*ค่าที่ปรับได้*]
แสดงวิธีใช้เกี่ยวกับพารามิเตอร์ที่ปรับได้ซึ่งระบุ ถ้ามีการระบุพารามิเตอร์ดังกล่าว มิฉะนั้น แสดงข้อความการใช้คำสั่ง **no**

-r ใช้พร้อมกับอ็อพชัน **-o** เพื่อเปลี่ยนค่าที่ปรับได้จากชนิด รีบูต เป็นชนิด ถาวร ในไฟล์ nextboot

-p ใช้พร้อมกับอ็อพชัน **-o** เพื่อทำให้ค่าที่ปรับได้แบบไดนามิกเป็นแบบถาวรใน ไฟล์ nextboot

-L *[ค่าที่ปรับได้]*
ใช้พร้อมกับอ็อพชัน **-o** เพื่อแสดงรายการลักษณะของค่าที่ปรับได้หนึ่งค่าหรือทั้งหมด โดยแสดงหนึ่งลักษณะต่อบรรทัด

ข้อมูลต่อไปนี้เป็นตัวอย่างของคำสั่ง **no**:

```
NAME                    CUR    DEF    BOOT   MIN    MAX    UNIT          TYPE     DEPENDENCIES
-------------------------------------------------------------------------------------------------

General Network Parameters

-------------------------------------------------------------------------------------------------
sockthresh              85     85     85     0      100    %_of_thewall      D
-------------------------------------------------------------------------------------------------
fasttimo                200    200    200    50     200    millisecond       D
-------------------------------------------------------------------------------------------------
inet_stack_size         16     16     16     1             kbyte             R
-------------------------------------------------------------------------------------------------
...lines omitted....


where:
CUR = current value

DEF = default value

BOOT = reboot value

MIN = minimal value

MAX = maximum value

UNIT = tunable unit of measure

TYPE = parameter type: D (for Dynamic), S (for Static), R for Reboot),B (for Bosboot), M (for Mount),
                       I (for Incremental) and C (for Connect)

DEPENDENCIES = list of dependent tunable parameters, one per line
```

แอ็ตทริบิวต์เครือข่ายบางอย่างเป็นแอ็ตทริบิวต์แบบรันไทม์ซึ่งสามารถเปลี่ยนแปลง ได้ตลอดเวลา แอ็ตทริบิวต์อื่นเป็นแอ็ตทริบิวต์แบบเวลาโหลดซึ่งต้องตั้งค่าก่อนที่จะโหลดส่วนขยายเคอร์นัล **netinet**

หมายเหตุ: เมื่อคุณใช้คำสั่ง **no** เพื่อเปลี่ยนพารามิเตอร์ ไดนามิกพารามิเตอร์จะถูกเปลี่ยนในหน่วยความจำ และการเปลี่ยนแปลงมีผล เฉพาะเมื่อบูตระบบครั้งถัดไปเท่านั้น ณ จุดนั้น พารามิเตอร์ทั้งหมดมีการตั้งค่าเป็นค่าติดตั้งรีบูตของ พารามิเตอร์เหล่านั้น เมื่อต้องการเปลี่ยนไดนามิกพารามิเตอร์เป็นแบบถาวร ให้ใช้อ็อพชัน **-r**หรือ **-p** ของ คำสั่ง **no** เพื่อตั้งค่าอ็อพชันในไฟล์ `nextboot` อ็อพชันพารามิเตอร์แบบรีบูตจำเป็นต้องรีบูตระบบเพื่อให้มีผลบังคับใช้

หากต้องการข้อมูลเพิ่มเติมเกี่ยวกับคำสั่ง **no** ให้ดู คำสั่ง no ใน *Commands Reference, Volume 4*

TCP fastpath loopback

ใช้อ็อพชัน transmission control protocol (TCP) fastpath loopback เพื่อให้ได้ประสิทธิภาพที่ดีขึ้นสำหรับทราฟฟิก loopback

พารามิเตอร์ที่สามารถปรับแต่งได้แบบเครือข่าย `tcp_fastlo` อนุญาตให้ทราฟฟิก TCP loopback สามารถลดระยะห่างสำหรับสแต็ก TCP/IP ทั้งหมด (เครือข่ายและอินเตอร์เฟส) เพื่อให้ได้ประสิทธิภาพที่ดีกว่า

แอ็พพลิเคชันไม่ต้องการการเปลี่ยนแปลงใดๆ เมื่อใช้อ็อพชันนี้ เมื่อเปิดใช้งาน ทราฟฟิก TCP loopback จะถูกจัดการเหมือนกับการใช้งานโดเมน UNIX

อ็อพชันที่สอง `tcp_fastlo_crosswpar` จะเปิดใช้งาน TCP fastpath เพื่อให้ทำงานระหว่างเวิร์กโหลดพาร์ติชัน (wpar) อ็อพชัน `tcp_fastlo` ต้องถูกเปิดใช้งานเพื่อให้อ็อพชัน `tcp_fastlo_crosswpar` ทำงานได้

เมื่อต้องการเปิดใช้งาน fastpath ของทราฟฟิก TCP loopback ให้ใช้คำสั่ง **no** โดยการป้อน:

```
# no -o tcp_fastlo=1
```

อ็อพชันนี้เป็นแบบไดนามิกและมีประสิทธิภาพสำหรับการเชื่อมต่อ TCP ในอนาคต

เมื่อต้องการเปิดใช้งาน fastpath ของทราฟฟิก TCP loopback ระหว่างเวิร์กโหลดพาร์ติชัน (wpar) ให้ใช้คำสั่ง **no** โดยการป้อน:

```
# no -o tcp_fastlo_crosswpar=1
```

หมายเหตุ: อ็อพชันสองอ็อพชัน คือ `tcp_fastlo` และ `tcp_fastlo_crosswpar` จะถูกปิดใช้งานในปัจจุบัน (ตั้งค่าเป็น 0) โดยดีฟอลต์ อ็อพชันเหล่านี้จะถูกสงวนไว้สำหรับรีลีสของ AIX ในอนาคต

ทราฟฟิก TCP fastpath loopback จะรับผิดชอบในสถิติที่เฉพาะโดยคำสั่ง **netstat** เมื่อการเชื่อมต่อ TCP ถูกเปิด ซึ่งจะไม่รับผิดชอบกับอินเตอร์เฟส loopback อย่างไรก็ตาม TCP fastpath loopback จะไม่ใช้อุปกรณ์ TCP/IP และ loopback เพื่อสร้างและยกเลิกการเชื่อมต่อ fast path ดังนั้น แพ็กเก็ตเหล่านี้จะรับผิดชอบในการทำงานปกติ

การหลีกเลี่ยงอินเตอร์รัปต์

การจัดการกับอินเตอร์รัปต์จะมีค่ามากกว่าในแง่ของวงรอบ CPU ของโฮสต์

หากต้องการจัดการกับอินเตอร์รัปต์ ระบบต้องบันทึกสถานะของเครื่องก่อน พิจารณาว่า อินเตอร์รัปต์มาจากที่ใด แล้วจึงดำเนินการกับภารกิจการดูแลจัดการ และเรียก interrupt handler ของไดร์เวอร์อุปกรณ์ที่ถูกต้อง ไดร์เวอร์อุปกรณ์ จะดำเนินการกับการใช้ในระดับสูง เช่น การอ่านเรจิสเตอร์สถานะการอินเตอร์รัปต์บนอะแด็ปเตอร์ ซึ่งจะเปรียบเทียบกับความเร็วเครื่องได้ช้า ใช้การล็อก SMP ขอรับและล้างบัฟเฟอร์ เป็นต้น

ไดร์เวอร์อุปกรณ์ AIX ส่วนใหญ่ไม่ใช้การส่งผ่านอินเตอร์รัปต์ที่สมบูรณ์ ซึ่งหลีกเลี่ยงอินเตอร์รัปต์ สำหรับแพ็กเก็ตการส่งออก การส่งการประมวลผลที่สมบูรณ์จะถูกจัดการบนการดำเนินการส่งถัดไป ดังนั้น การหลีกเลี่ยงการส่งที่แยกออกจะเสร็จสิ้นการอินเตอร์รัปต์ คุณสามารถใช้คำสั่ง เช่น **netstat -v, entstat, atmstat** หรือ **fddistat** เพื่อดูสถานะของจำนวนแพ็กเก็ตที่ส่งผ่าน หรือแพ็กเก็ตที่รับและจำนวนอินเตอร์รัปต์ที่ส่งผ่านและอินเตอร์รัปต์ที่รับ จากข้อมูลสถิติ คุณสามารถมองเห็นได้ว่า อินเตอร์รัปต์ที่ส่งผ่าน จะถูกหลีกเลี่ยง อะแด็ปเตอร์และไดร์เวอร์กลุ่มที่สามบางตัวอาจไม่ปฏิบัติตามระเบียบนี้

การเปิดใช้งานการใช้เธรดบนอะแด็ปเตอร์ LAN

ด้วยการเปิดใช้คุณลักษณะ *dog threads* ไดร์เวอร์จะจัดคิวแพ็กเก็ตขาเข้า ในเธรด และเธรดจัดการการเรียก IP, TCP และโค้ดซ็อกเก็ต

ตามค่าดีฟอลต์ ไดร์เวอร์จะเรียก IP โดยตรง ซึ่งจะเรียกโปรโตคอล ที่สแต็กไปยังระดับของซ็อกเก็ตขณะที่รันบนช่วงของอินเตอร์รัปต์ ซึ่งจะช่วยลดความยาวพาธของคำสั่ง แต่เพิ่มเวลาในการพักอินเตอร์รัปต์ บนระบบ SMP CPU เดี่ยวสามารถเกิดคอขวดสำหรับแพ็กเก็ตที่รับจาก อะแด็ปเตอร์แบบเร็ว เธรดสามารถรันบน CPU อื่นๆ ซึ่งอาจไม่ได้ทำงาน การเปิดใช้งาน dog thread สามารถเพิ่มความสามารถของระบบในบางกรณี ซึ่งอัตราแพ็กเก็ตขาเข้าจะสูงมาก และอนุญาตให้แพ็กเก็ตขาเข้าประมวลผลในแบบขนาน โดย CPU จำนวนมาก

ด้านล่างของคุณลักษณะ dog thread นั่นคือ การเพิ่มเวลาแฝงภายใต้โหลดและยังเพิ่มการใช้ประโยชน์จากโฮสต์ CPU เนื่องจากแพ็กเก็ตที่รอคิวไปยังเธรด และเธรดได้ถูกจัดส่งแล้ว

หมายเหตุ: คุณลักษณะนี้ไม่ได้สนับสนุนบนยูนิโพรเซสเซอร์ เนื่องจากจะเพิ่มความยาวพาธและลดความเร็วของผลการทำงานลง

นี่คือ คุณลักษณะสำหรับด้านอินพุต (รับ) ของอะแด็ปเตอร์ LAN ซึ่งสามารถปรับแต่งค่าที่ระดับของอินเตอร์เฟสด้วยคำสั่ง **ifconfig** (**ifconfig** *interface* **thread** หรือ **ifconfig** *interface hostname* **up thread**)

หากต้องการปิดใช้งานคุณลักษณะ ให้ใช้คำสั่ง **ifconfig** *interface* **–thread** ตามที่แสดงในตัวอย่างต่อไปนี้:

```
# ifconfig en0 thread

# ifconfig en0
en0: flags=5e080863,e0<UP,BROADCAST,NOTRAILERS,RUNNING,SIMPLEX,MULTICAST,GROUPRT,64BIT,CHECKSUM_OFFLOAD,PSEG,THREAD,CHAIN>
        inet 192.1.0.1 netmask 0xffffff00 broadcast 192.1.0.255

# ifconfig en0 -thread

# ifconfig en0
en0: flags=5e080863,c0<UP,BROADCAST,NOTRAILERS,RUNNING,SIMPLEX,MULTICAST,GROUPRT,64BIT,CHECKSUM_OFFLOAD,PSEG,THREAD,CHAIN>
        inet 192.1.0.1 netmask 0xffffff00 broadcast 192.1.0.255
```

คำสั่ง **netstat –s** และแสดงตัวนับบางตัวเพื่อแสดงจำนวนของแพ็กเก็ตการประมวลผลโดยเธรด และ ถ้าเธรดที่รอคิวจะปล่อยแพ็กเก็ตขาเข้าใดๆ ต่อไปนี้คือตัวอย่างของคำสั่ง **netstat –s** :

```
# netstat -s| grep hread

        352 packets processed by threads
        0 packets dropped by threads
```

คำแนะนำเมื่อพิจารณาการใช้ dog thread ดังต่อไปนี้:

- มี CPU มากกว่าอะแด็ปเตอร์ที่จำเป็นในการติดตั้ง โดยปกติแล้ว จะแนะนำให้มีอย่างน้อยสองเท่าของ CPU ที่มากกว่าอะแด็ปเตอร์
- ระบบที่มีผลประโยชน์ของความเร็ว CPU น้อย เครื่องที่มีความเร็ว CPU ช้ากว่าอาจถูกให้ความช่วยเหลือมากที่สุด
- คุณลักษณะนี้คือการปรับปรุงผลการทำงานเมื่อมีอัตราแพ็กเก็ตอินพุต สูง ซึ่งจะปรับปรุงผลการทำงานมากกว่าบน MTU 1500 เปรียบเทียบกับ MTU 9000 (กรอบขนาดใหญ่มาก) บนกิกะบิตที่เป็นอัตราแพ็กเก็ตจะสูงกว่าบนเน็ตเวิร์กขนาดเล็ก MTU

  dog thread รันได้ดีที่สุดเมื่อค้นหาการทำงานมากขึ้นบนคิว และไม่มีการส่งกลับไปเป็น sleep (รอสำหรับอินพุต) นี่จะช่วยลดการใช้ไดรเวอร์ ที่ตื่นตัวกับเธรด และระบบที่จัดส่งเธรด
- dog thread ยังสามารถลดจำนวนของเวลาที่ CPU เฉพาะ ใช้กับอินเตอร์รัปต์ที่พรางตัว ซึ่งสามารถลด CPU เพื่อกลับสู่การทำงานในระดับผู้ใช้ ได้เร็วขึ้น
- dog thread ยังสามารถลดผลการทำงานประมาณ 10 เปอร์เซ็นต์ ถ้าอัตราแพ็กเก็ตไม่เร็วเท่าที่จะอนุญาตให้เธรดรันต่อไป 10 เปอร์เซ็นต์นี้คือจำนวนเฉลี่ยของ CPU ที่เพิ่มขึ้นซึ่งจำเป็นต่อกำหนดตารางเวลา และจัดส่งเธรด

Interface-Specific Network Options

Interface–Specific Network Options (ISNO) อนุญาตให้ IP เน็ตเวิร์กอินเตอร์เฟส ถูกปรับแต่งเองสำหรับผลการทำงานที่ดีที่สุด

ค่าที่ตั้งค่าไว้สำหรับอินเตอร์เฟสแต่ละตัวจะใช้ผ่านค่าของระบบที่มีอยู่ก่อน ซึ่งตั้งค่าด้วยคำสั่ง **no** คุณลักษณะจะถูกเปิดใช้งาน (ดีฟอลต์) หรือถูกปิดใช้งานสำหรับระบบทั้งหมดด้วยคำสั่ง **no** อ็อพชัน **use_isno** อ็อพชันการปิดใช้งาน ISNO สำหรับจุดเดี่ยวนี้จะถูกรวมไว้ในเครื่องมือการวินิจฉัย เพื่อกำจัดข้อผิดพลาดในการปรับที่เป็นไปได้หากผู้ดูแลระบบต้องการแยกปัญหาเกี่ยวกับ ผลการทำงาน

โปรแกรมเมอร์และนักวิเคราะห์ผลการทำงานควรจดบันทึกว่า ค่า ISNO ไม่ควรแสดงอยู่ในซ็อกเก็ต (หมายความว่า ค่าเหล่านั้นไม่สามารถอ่านได้โดยการเรียกของระบบ **getsockopt()**) จนกว่าการเชื่อมต่อ TCP ได้ถูกดำเนินการ เน็ตเวิร์กอินเตอร์เฟสที่ระบุเฉพาะ ซึ่งซ็อกเก็ตที่ใช้จริงจะไม่ทราบจนกว่าการเชื่อมต่อจะเสร็จสิ้น ดังนั้น ซ็อกเก็ตจะสะท้อนถึงค่าดีฟอลต์ของระบบจากคำสั่ง **no** หลังจากที่การเชื่อมต่อ TCP ถูกยอมรับ และเน็ตเวิร์กอินเตอร์เฟสที่รู้จัก ค่า ISNO อาจถูกวางลงในซ็อกเก็ต

พารามิเตอร์ต่อไปนี้ได้เพิ่มไว้สำหรับเน็ตเวิร์กอินเตอร์เฟสที่สนับสนุนแต่ละตัว และจะมีประสิทธิพลสำหรับการเชื่อมต่อ TCP (และไม่ใช่ UDP):

- **rfc1323**
- **tcp_nodelay**
- **tcp_sendspace**
- **tcp_recvspace**
- **tcp_mssdflt**

เมื่อตั้งค่าสำหรับอินเตอร์เฟสที่ระบุเฉพาะแล้ว ค่าเหล่านี้จะแทนที่ค่าอ็อพชัน **no** ที่ตั้งค่าไว้สำหรับระบบ พารามิเตอร์เหล่านี้จะพร้อมใช้งานสำหรับอินเตอร์เฟส TCP/IP หลัก (โทเค็นริง FDDI อีเทอร์เน็ต 10/100 และกิกะบิตอีเทอร์เน็ต) ยกเว้น **css#** IP interface บนสวิตช์ SP เนื่องจากวิธีการแก้ปัญหาแบบง่ายๆ ผู้ใช้สวิตช์ SP สามารถตั้งค่าอ็อพชันการปรับที่เหมาะสมสำหรับสวิตช์โดยใช้คำสั่ง **no** จากนั้นใช้ ISNO เพื่อตั้งค่าตามความต้องการสำหรับอินเตอร์เฟสระบบอื่นๆ

อ็อพชันเหล่านี้จะถูกตั้งค่าไว้สำหรับอินเตอร์เฟส TCP/IP (เช่น en0 หรือ tr0) และไม่ใช่กับเน็ตเวิร์กอะแด็ปเตอร์ (ent0 หรือ tok0)

AIX จะตั้งค่าดีฟอลต์สำหรับอินเตอร์เฟสแบบ กิกะบิตอีเทอร์เน็ต สำหรับทั้ง MTU 1500 และสำหรับโหมดของกรอบที่มีขนาดใหญ่ (MTU 9000) ตราบเท่าที่คุณปรับแต่งอินเตอร์เฟสผ่านหน้าจอ **SMIT tcpip** อ็อพชัน ISNO ควรถูกตั้งค่าตามค่าดีฟอลต์ ซึ่งจัดเตรียม ผลการทำงานที่ดีขึ้น

สำหรับ 10/100 อีเทอร์เน็ตและอะแด็ปเตอร์โทเค็นริง ค่าดีฟอลต์ ISNO จะไม่ถูกตั้งค่าโดยระบบ ตามที่อะแด็ปเตอร์เหล่านั้นควรทำงานได้ดีด้วยค่าดีฟอลต์ของระบบ **no** แบบโกลบอล อย่างไรก็ตาม แอ็ตทริบิวต์ ISNO สามารถตั้งค่าได้หากต้องการเพื่อแทนที่ค่าดีฟอลต์แบบโกลบอล

ตัวอย่างต่อไปนี้แสดงค่า ISNO ที่เป็นค่าดีฟอลต์สำหรับ **tcp_sendspace** และ **tcp_recvspace** สำหรับ GigE ในโหมด MTU 1500:

```
# ifconfig en0
en0: flags=5e080863,c0<UP,BROADCAST,NOTRAILERS,RUNNING,SIMPLEX,MULTICAST,GROUPRT,64BIT,CHECKSUM_OFFLOAD,PSEG,CHAIN>
        inet 10.0.0.1 netmask 0xffffff00 broadcast 192.0.0.255
        tcp_sendspace 131072 tcp_recvspace 65536
```

สำหรับโหมดกรอบขนาดใหญ่มาก ค่าดีฟอลต์ ISNO สำหรับ **tcp_sendspace, tcp_recvspace** และ **rfc1323** จะถูกตั้งค่าดังต่อไปนี้:

```
 # ifconfig en0
en0: flags=5e080863,c0<UP,BROADCAST,NOTRAILERS,RUNNING,SIMPLEX,MULTICAST,GROUPRT,64BIT,CHECKSUM_OFFLOAD,PSEG,CHAIN>
       inet 192.0.0.1 netmask 0xffffff00 broadcast 192.0.0.255
       tcp_sendspace 262144 tcp_recvspace 131072 rfc1323 1
```

ใช้ค่าติดตั้งต่อไปนี้เพื่อเปิดใช้งาน **rfc1323** หากขนาดของ MTU คือ 4096 ไบต์หรือมากกว่า และเพื่อตั้งค่า **tcp_sendspace** และ **tcp_recvspace** ให้มีขนาดอย่างน้อย 128 KB สำหรับอะแด็ปเตอร์ที่มีความเร็วสูง (กิกะบิตหรือเร็วกว่า) อะแด็ปเตอร์ที่มีความเร็วสูง ถูกตั้งค่า 256 KB ค่า "เปล่า" หมายความถึงอ็อพชันที่ไม่ได้ตั้งค่า ดังนั้น ค่านั้นจึงสืบทอดค่าติดตั้ง "no" แบบโกลบอล

อินเตอร์เฟส	ความเร็ว	MTU	tcp_sendspace	tcp_recvspace	rfc1323	tcp_nodelay	tcp_mssdflt
lo0 (loopback)	N/A	16896	131072	131072	1		
อีเทอร์เน็ต	10 หรือ 100 (Mbit)						
อีเทอร์เน็ต	1000 (Gigabit)	1500	131072	65536	1		
อีเทอร์เน็ต	1000 (Gigabit)	9000	262144	131072	1		
อีเทอร์เน็ต	10 GigE	1500	262144	262144	1		
อีเทอร์เน็ต	10 GigE	9000	262144	262144	1		
Ether Channel	ปรับแต่งตามค่าความเร็ว/MTU ของอินเตอร์เฟส						
อีเทอร์เน็ตเสมือน	N/A	อื่นๆ	262144	262144	1		
InfiniBand	N/A	2044	131072	131072	1		

คุณสามารถตั้งค่าอ็อพชัน ISNO ได้ด้วยเมธอดดังต่อไปนี้:

- SMIT
- คำสั่ง **chdev**
- คำสั่ง **ifconfig**

การใช้ SMIT หรือคำสั่ง **chdev** เพื่อเปลี่ยนค่าในฐานข้อมูล ODM บนดิสก์ ดังนั้นค่าเหล่านั้นจะเป็นค่าแบบถาวร คำสั่ง **ifconfig** จะเปลี่ยนค่าเฉพาะในหน่วยความจำเท่านั้น ดังนั้น ค่าเหล่านั้นจะกลับสู่ค่าก่อนหน้าที่เก็บอยู่ใน ODM สำหรับการรีบูตในครั้งถัดไป

การแก้ไขอ็อพชัน ISNO ด้วย SMIT:

คุณสามารถเปลี่ยนอ็อพชัน ISNO ด้วย SMIT ได้

ป้อนคำสั่งต่อไปนี้ที่บรรทัดรับคำสั่ง:

```
# smitty tcpip
```

1. เลือกอ็อพชัน Futher Configuration
2. เลือกอ็อพชัน Network Interfaces
3. เลือก Network Interface Selection
4. เลือก Change/Show Characteristics of a Network Interface

5. เลือกอินเตอร์เฟสด้วยเคอร์เซอร์ของคุณ ตัวอย่างเช่น en0

จากนั้น คุณจะมองเห็นหน้าจอต่อไปนี้:

```
Change / Show a Standard Ethernet Interface

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                [Entry Fields]
  Network Interface Name                         en0
  INTERNET ADDRESS (dotted decimal)             [192.0.0.1]
  Network MASK (hexadecimal or dotted decimal)  [255.255.255.0]
  Current STATE                                  up                      +
  Use Address Resolution Protocol (ARP)?         yes                     +
  BROADCAST ADDRESS (dotted decimal)            []
  Interface Specific Network Options
    ('NULL' will unset the option)
    rfc1323                                     []
    tcp_mssdflt                                 []
    tcp_nodelay                                 []
    tcp_recvspace                               []
    tcp_sendspace                               []


F1=Help                  F2=Refresh               F3=Cancel                F4=List
Esc+5=Reset              Esc+6=Command            Esc+7=Edit               Esc+8=Image
Esc+9=Shell              Esc+0=Exit               Enter=Do
```

โปรดสังเกตว่า ดีฟอลต์ของระบบ ISNO จะไม่แสดง แม้ว่าจะถูกตั้งค่าไว้ภายในก็ตาม สำหรับตัวอย่างนี้ ให้แทนที่ค่าดีฟอลต์สำหรับ **tcp_sendspace** และลดจำนวนลงเป็น 65536

ทำการสำรองข้อมูลอินเตอร์เฟสด้วย **smitty tcpip** และเลือก Minimum Configuration and Startup จากนั้น เลือก en0 และใช้ค่าดีฟอลต์ที่ตั้งค่าไว้ เมื่อติดตั้งอินเตอร์เฟสในครั้งแรก

ถ้าคุณใช้คำสั่ง **ifconfig** เพื่อแสดงอ็อพชัน ISNO คุณสามารถมองเห็นค่าของแอ็ตทริบิวต์ **tcp_sendspace** ที่มีค่าเป็น 65536 ในปัจจุบัน ต่อไปนี้คือตัวอย่าง:

```
# ifconfig en0
en0: flags=5e080863,c0<UP,BROADCAST,NOTRAILERS,RUNNING,SIMPLEX,MULTICAST,GROUPRT,64BIT,CHECKSUM_OFFLOAD,PSEG,CHAIN>
         inet 192.0.0.1 netmask 0xffffff00 broadcast 192.0.0.255
         tcp_sendspace 65536 tcp_recvspace 65536
```

เอาต์พุตคำสั่ง **lsattr** ยังแสดงค่าดีฟอลต์ของระบบที่ถูกแทนที่สำหรับแอ็ตทริบิวต์นี้:

```
# lsattr -E -l en0
alias4                      IPv4 Alias including Subnet Mask           True
alias6                      IPv6 Alias including Prefix Length         True
arp           on            Address Resolution Protocol (ARP)          True
authority                   Authorized Users                           True
broadcast                   Broadcast Address                          True
mtu           1500          Maximum IP Packet Size for This Device     True
netaddr       192.0.0.1     Internet Address                           True
netaddr6                    IPv6 Internet Address                      True
netmask       255.255.255.0 Subnet Mask                                True
prefixlen                   Prefix Length for IPv6 Internet Address    True
remmtu        576           Maximum IP Packet Size for REMOTE Networks True
```

```
rfc1323                     Enable/Disable TCP RFC 1323 Window Scaling True
security      none          Security Level                             True
state         up            Current Interface Status                   True
tcp_mssdflt                 Set TCP Maximum Segment Size               True
tcp_nodelay                 Enable/Disable TCP_NODELAY Option          True
tcp_recvspace               Set Socket Buffer Space for Receiving      True
tcp_sendspace 65536         Set Socket Buffer Space for Sending        True
```

การแก้ไขอ็อพชัน ISNO ด้วยคำสั่ง chdev และ ifconfig:

คุณสามารถใช้คำสั่งต่อไปนี้เพื่อตรวจสอบระบบในครั้งแรก และตรวจสอบอินเตอร์เฟสที่สนับสนุน จากนั้นตั้งค่าและตรวจสอบค่าใหม่

- ตรวจสอบให้แน่ใจว่า อ็อพชัน **use_isno** ถูกเปิดใช้งานโดยใช้คำสั่งต่อไปนี้ :

  ```
  # no -a | grep isno
  use_isno = 1
  ```

- ตรวจสอบให้แน่ใจว่า อินเตอร์เฟสสนับสนุน ISNO ใหม่ทั้งหมดห้าอินเตอร์เฟสโดยใช้คำสั่ง **lsattr -El**:

  ```
  # lsattr -E -l en0 -H
  attribute           value description                         user_settable
  :
  rfc1323             Enable/Disable TCP RFC 1323 Window Scaling True
  tcp_mssdflt         Set TCP Maximum Segment Size               True
  tcp_nodelay         Enable/Disable TCP_NODELAY Option          True
  tcp_recvspace       Set Socket Buffer Space for Receiving      True
  tcp_sendspace       Set Socket Buffer Space for Sending        True
  ```

- ตั้งค่าที่ระบุเฉพาะอินเตอร์เฟสโดยใช้คำสั่ง **ifconfig** หรือ **chdev** อย่างใดอย่างหนึ่ง คำสั่ง **ifconfig** จะตั้งค่าชั่วคราว (เหมาะสมกับการใช้เพื่อการทดสอบมากที่สุด) คำสั่ง **chdev** จะปรับเปลี่ยน ODM ดังนั้นค่าที่กำหนดเองจะส่งคืนหลังจากที่ระบบรีบูต

  ตัวอย่างเช่น ในการตั้งค่า **tcp_recvspace** และ **tcp_sendspace** ให้มีค่า 64 KB และเปิดใช้งาน **tcp_nodelay** ให้ใช้หนึ่งในเมธอด ต่อไปนี้:

  ```
  # ifconfig en0 tcp_recvspace 65536 tcp_sendspace 65536 tcp_nodelay 1
  ```

  หรือ

  ```
  # chdev -l en0 -a tcp_recvspace=65536 -a tcp_sendspace=65536 -a tcp_nodelay=1
  ```

- ตรวจสอบค่าติดตั้งโดยใช้คำสั่ง **ifconfig** หรือ **lsattr**:

  ```
  # ifconfig en0
  en0: flags=5e080863,c0<UP,BROADCAST,NOTRAILERS,RUNNING,SIMPLEX,MULTICAST,GROUPRT,64BIT,CHECKSUM_OFFLOAD,PSEG,CHAIN>
          inet 9.19.161.100 netmask 0xffffff00 broadcast 9.19.161.255
          tcp_sendspace 65536 tcp_recvspace 65536 tcp_nodelay 1
  ```

  หรือ

  ```
  # lsattr -El en0
  rfc1323             Enable/Disable TCP RFC 1323 Window Scaling True
  tcp_mssdflt         Set TCP Maximum Segment Size               True
  tcp_nodelay   1     Enable/Disable TCP_NODELAY Option          True
  tcp_recvspace 65536 Set Socket Buffer Space for Receiving      True
  tcp_sendspace 65536         Set Socket Buffer Space for Sending        True
  ```

การปรับเวิร์กโหลด TCP

มีค่า AIX ที่ปรับได้อยู่หลายค่า ซึ่งอาจกระทบต่อประสิทธิภาพ TCP

แอ็พพลิเคชันจำนวนมากใช้ Transport Control Protocol (TCP) ที่เชื่อถือได้ รวมถึงคำสั่ง **ftp** และ **rcp**

หมายเหตุ: คำสั่ง **no -o** เตือนคุณว่า เมื่อคุณเปลี่ยนอ็อพชันการปรับที่มีผลกระทบต่อการเชื่อมต่อ TCP/IP การเปลี่ยนแปลงจะมีผลเฉพาะสำหรับการเชื่อมต่อที่สร้างขึ้นหลังจากทำการเปลี่ยน แปลงแล้วเท่านั้น นอกจากนี้ คำสั่ง **no -o** จะรีสตาร์ทกระบวนการ **inetd** daemon เมื่ออ็อพชันเปลี่ยนแปลง ซึ่งอาจกระทบต่อกระบวนการซึ่ง **inetd** daemon กำลังรอฟังการเชื่อมต่อใหม่

การปรับเวิร์กโหลดการสตรีม TCP:

เวิร์กโหลดการสตรีมย้ายข้อมูลจำนวนมากจากปลายด้านหนึ่งไปยัง ปลายอีกด้านหนึ่ง ตัวอย่างของเวิร์กโหลดการสตรีมคือการโอนย้ายไฟล์ เวิร์กโหลดแบ็คอัพหรือการเรียกคืน หรือการโอนย้ายข้อมูลจำนวนมาก ส่วนประกอบหลักที่น่าสนใจในเวิร์กโหลดเหล่านี้คือแบนด์วิธ แต่คุณยังสามารถดู ที่เวลาแฝงจากปลายถึงปลายได้ด้วย

ตัวปรับหลักที่มีผลต่อประสิทธิภาพ TCP สำหรับแอ็พพลิเคชันการสตรีม มีดังต่อไปนี้:

- **tcp_recvspace**
- **tcp_sendspace**
- **rfc1323**
- **การค้นหา MTU ของพาธ**
- **tcp_nodelayack**
- **sb_max**
- อะแด็ปเตอร์อ็อพชัน เช่น checksum offload และ TCP Large Send

ตารางต่อไปนี้แสดงขนาดที่แนะนำสำหรับค่าที่ปรับได้เพื่อให้ได้ ประสิทธิภาพสูงสุด ตามข้อมูลชนิดของอะแด็ปเตอร์และขนาด MTU:

อุปกรณ์	ความเร็ว	ขนาด MTU	tcp_sendspace	tcp_recvspace	sb_max [1]	rfc1323
Token Ring	4 หรือ 16 Mbit	1492	16384	16384	32768	0
อีเทอร์เน็ต	10 Mbit	1500	16384	16384	32768	0
อีเทอร์เน็ต	100 Mbit	1500	16384	16384	65536	0
อีเทอร์เน็ต	กิกะบิต	1500	131072	65536	131072	0
อีเทอร์เน็ต	กิกะบิต	9000	131072	65535	262144	0
อีเทอร์เน็ต	กิกะบิต	9000	262144	131072[2]	524288	1
อีเทอร์เน็ต	10 กิกะบิต	1500	131072	65536	131072	0
อีเทอร์เน็ต	10 กิกะบิต	9000	262144	131072	262144	1
ATM	155 Mbit	1500	16384	16384	131072	0
ATM	155 Mbit	9180	65535	65535[3]	131072	0
ATM	155 Mbit	65527	655360	655360[4]	1310720	1

อุปกรณ์	ความเร็ว	ขนาด MTU	tcp_sendspace	tcp_recvspace	sb_max [1]	rfc1323
FDDI	100 Mbit	4352	45056	45056	90012	0
Fibre Channel	2 กิกะบิต	65280	655360	655360	1310720	1

(1) แนะนำให้ใช้ค่าดีฟอลต์ 1048576 สำหรับ *sb_max* ที่ปรับได้ ค่าที่แสดงในตาราง คือค่าต่ำสุดที่ยอมรับได้สำหรับ *sb_max* ที่ปรับได้

(2) ประสิทธิภาพจะ ดีขึ้นเล็กน้อยเมื่อใช้อ็อพชันเหล่านี้ โดยมี rfc1323 ที่เปิดใช้งาน บน jumbo frames บนกิกะบิตอีเทอร์เน็ต

(3) บางชุดของ พื้นที่ว่างการส่งและการรับ TCP จะส่งผลให้ผลผลิตต่ำมาก (1 Mbit หรือน้อยกว่า) เพื่อหลีกเลี่ยงปัญหานี้ ให้ตั้งค่า *tcp_sendspace* ที่ปรับได้ เป็นค่าต่ำสุด ซึ่งเป็นสามเท่าของขนาด MTU หรือเป็นค่าที่มากกว่าหรือเท่ากับค่า *tcp_recvspace* ของเครื่องรับ

(4) TCP มีค่า 16 บิตเท่านั้นที่จะใช้สำหรับขนาดหน้าต่าง ค่านี้จะแปลเป็นขนาดหน้าต่างสูงสุด 65536 ไบต์ สำหรับอะแด็ปเตอร์ที่มีขนาด MTU ใหญ่ (ตัวอย่างเช่น 32 KB หรือ 64 KB) ประสิทธิภาพการสตรีม TCP อาจแย่มาก ตัวอย่างเช่น บนอุปกรณ์ที่มีขนาด MTU 64 KB และมีการตั้งค่า *tcp_recvspace* เป็น 64 KB TCP สามารถส่งได้เพียงหนึ่งแพ็กเก็ตเท่านั้น จากนั้นหน้าต่างจะปิด TCP ต้องรอ ACK กลับจากเครื่องรับก่อนสามารถส่ง อีกครั้งได้ ปัญหานี้สามารถแก้ไขได้ด้วยวิธีอย่างใดอย่างหนึ่งต่อไปนี้:

- เปิดใช้งาน *rfc1323* ซึ่งเพิ่มประสิทธิภาพ TCP และทำให้สามารถทะลุ ขีดจำกัด 16 บิต ส่งผลให้สามารถใช้ขนาดหน้าต่างที่ใหญ่กว่า 64 KB ได้ จากนั้น คุณสามารถตั้งค่า *tcp_recvspace* ที่ปรับได้ เป็นค่าจำนวนมาก เช่น 10 เท่าของขนาด MTU ซึ่งอนุญาตให้ TCP สตรีมข้อมูลและให้ประสิทธิภาพที่ดี
- ลดขนาด MTU ของอะแด็ปเตอร์ ตัวอย่างเช่น ใช้คำสั่ง **ifconfig at0 mtu 16384** เพื่อตั้งค่าขนาด ATM MTU เป็น 16 KB ซึ่งส่งผลให้ TCP คำนวณค่า MSS ที่เล็กลง ด้วยขนาด MTU 16 KB TCP สามารถส่งข้อมูลได้ สี่แพ็กเก็ตสำหรับขนาดหน้าต่าง 64 KB

คำแนะนำทั่วไปสำหรับการปรับเวิร์กโหลดการสตรีม TCP มีดังต่อไปนี้:

- ตั้งค่าพื้นที่ว่างการส่งและการรับ TCP เป็น 10 เท่าของขนาด MTU เป็นอย่างน้อย
- คุณควรเปิดใช้งาน *rfc1323* เมื่อขนาด MTU มากกว่า 8 KB เพื่อให้ค่าพื้นที่ว่างการรับ TCP มากขึ้น
- สำหรับอะแด็ปเตอร์ความเร็วสูง ค่าพื้นที่ว่างการส่งและการรับ TCP ที่ใหญ่ขึ้น ช่วยให้ประสิทธิภาพดีขึ้น
- สำหรับอะแด็ปเตอร์ความเร็วสูง ค่า *tcp_sendspace* ที่ปรับได้ ควรจะเป็น 2 เท่าของค่า *tcp_recvspace*
- มีการตั้งค่า *rfc1323* สำหรับอินเตอร์เฟส lo0 โดยค่าดีฟอลต์ ขนาด MTU ดีฟอลต์ของ **lo0** สูงกว่า 1500 ดังนั้น *tcp_sendspace* และ *tcp_recvspace* ที่ปรับได้ จึงมีการตั้งค่าเป็น 128K

คำสั่ง **ftp** และ **rcp** เป็นตัวอย่างของ TCP แอ็พพลิเคชันที่ได้รับประโยชน์จากการปรับ *tcp_sendspace* และ *tcp_recvspace* ที่ปรับได้

tcp_recvspace ที่ปรับได้:

tcp_recvspace ที่ปรับได้ระบุจำนวนไบต์ของข้อมูล ที่ระบบการรับสามารถบัฟเฟอร์ในเคอร์นัลบนซ็อกเก็ตคิวการรับ ได้

tcp_recvspace ที่ปรับได้ยังใช้โดยโปรโตคอล TCP เพื่อตั้งค่าขนาดหน้าต่าง TCP ซึ่ง TCP ใช้เพื่อจำกัดจำนวนไบต์ของข้อมูลที่จะส่งไปยังเครื่องรับ เพื่อให้มั่นใจว่าเครื่องรับมีพื้นที่ว่างเพียงพอที่จะบัฟเฟอร์ ข้อมูล *tcp_recvspace* ที่ปรับได้ เป็นพารา

มิเตอร์ที่สำคัญสำหรับประสิทธิภาพ TCP เนื่องจาก TCP ต้องสามารถส่ง ผ่านหลายแพ็กเก็ตเข้าในเครือข่ายได้ เพื่อให้แน่ใจว่าไปป์ไลน์เครือข่ายเต็ม ถ้า TCP ไม่สามารถทำให้มีแพ็กเก็ตที่เพียงพอในไปป์ไลน์ ประสิทธิภาพจะด้อยลง

คุณสามารถตั้งค่า *tcp_recvspace* ที่ปรับได้โดยใช้วิธีการ ดังต่อไปนี้:

- การเรียกระบบ **setsockopt( )** จากโปรแกรม
- คำสั่ง **no -o tcp_recvspace=[*value*]**
- พารามิเตอร์ *tcp_recvspace* ISNO

คำแนะนำทั่วไปสำหรับ *tcp_recvspace* ที่ปรับได้คือ ควรตั้งเป็นค่าที่น้อยกว่าขนาด MTU อย่างน้อย 10 เท่า คุณสามารถกำหนดค่า *tcp_recvspace* ที่ปรับได้โดยการหารค่า *bandwidth-delay product* ด้วย 8 ซึ่งคำนวณด้วยสูตรต่อไปนี้:

```
bandwidth-delay product = capacity(bits)= bandwidth(bits/second) x round-trip time (seconds)
```

การหารค่าความสามารถด้วย 8 ให้ค่าประเมินที่ดีของขนาดหน้าต่าง TCP ที่ต้องใช้ เพื่อรักษาไปป์ไลน์เครือข่ายให้เต็ม ความล่าช้าในการเดินทางไปกลับยิ่งนาน และความเร็วเครือข่ายยิ่งมาก ค่า *bandwidth-delay product* ยิ่งมาก ส่งผลให้หน้าต่าง TCP ใหญ่ขึ้น ตัวอย่าง ของกรณีนี้คือเครือข่าย 100 Mbit ที่มีเวลาเดินทางไปกลับ 0.2 มิลลิวินาที คุณสามารถคำนวณค่า *bandwidth-delay product* ด้วยสูตรข้างบน:

```
bandwidth-delay product = 100000000 x 0.0002 = 20000
20000/8 = 2500
```

ดังนั้น ในตัวอย่างนี้ ขนาดหน้าต่าง TCP ต้องมีค่าอย่างน้อย 2500 ไบต์ บน 100 Mbit และกิกะบิตอีเทอร์เน็ตบน LAN เดียว คุณอาจต้องการ ตั้งค่า *tcp_recvspace* และ *tcp_sendspace* ที่ ปรับได้เป็นอย่างน้อย 2 หรือ 3 เท่าของค่า *bandwidth-delay product* ที่คำนวณได้ เพื่อให้ได้ประสิทธิภาพการทำงานสูงสุด

tcp_sendspace ที่ปรับได้:

tcp_sendspace ที่ปรับได้ระบุจำนวนข้อมูล ที่แอ็พพลิเคชันการส่งสามารถบัฟเฟอร์ไว้ในเคอร์นัลได้ ก่อนแอ็พพลิเคชัน จะถูกบล็อกในการเรียกการส่ง

บัฟเฟอร์การส่งซ็อกเก็ต TCP ใช้เพื่อบัฟเฟอร์ข้อมูลแอ็พพลิเคชันไว้ในเคอร์นัล โดยใช้ mbufs/คลัสเตอร์ ก่อนข้อมูลนั้นจะถูกส่งไปยังเครื่องรับโดยโปรโตคอล TCP ขนาดดีฟอลต์ของบัฟเฟอร์การส่งมีการระบุโดยค่า *tcp_sendspace* ที่ปรับได้ หรือโปรแกรมสามารถใช้รูทีนย่อย **setsockopt( )** เพื่อแทนที่ ค่า

คุณควรตั้งค่า *tcp_sendspace* ที่ปรับได้เป็นค่าเท่ากับค่า *tcp_recvspace* เป็นอย่างน้อย และสำหรับอะแด็ปเตอร์ความเร็ว สูง ค่า *tcp_sendspace* ควรเป็นสองเท่า ของค่า *tcp_recvspace* เป็นอย่างน้อย

ถ้าแอ็พพลิเคชันระบุ O_NDELAY หรือ O_NONBLOCK บนซ็อกเก็ตซึ่ง ทำให้เกิด I/O ที่ไม่บล็อก จากนั้น ถ้าบัฟเฟอร์การส่งเต็ม แอ็พพลิเคชันจะส่งคืน พร้อมกับข้อผิดพลาด EWOULDBLOCK/EAGAIN แทนการพักไว้ แอ็พพลิเคชันต้องมีการโค้ดเพื่อจัดการข้อผิดพลาดนี้ (โซลูชันที่แนะนำคือ พักไว้ชั่วครู่หนึ่ง แล้วส่งใหม่อีกครั้ง)

rfc1323 ที่ปรับได้:

rfc1323 ที่ปรับได้ช่วยให้สามารถใช้อ็อพชันการปรับสเกล หน้าต่าง TCP ได้

อ็อพชันการปรับสเกลหน้าต่าง TCP คืออ็อพชันที่ตกลงกับ TCP ดังนั้นจึงต้อง มีการเปิดใช้งานบนจุดปลายทั้งสองด้านของการเชื่อมต่อ TCP เพื่อให้เกิดผล โดยค่าดีฟอลต์ ขนาดหน้าต่าง TCP ถูกจำกัดไว้ที่ 65536 ไบต์ (64 K) แต่สามารถตั้งค่าให้สูงขึ้น

ได้ ถ้ามีการตั้งค่า *rfc1323* เป็น 1 ถ้าคุณกำลังตั้งค่า *tcp_recvspace* ให้มากกว่า 65536 ให้ตั้งค่า *rfc1323* เป็น 1 บนแต่ละด้านของการเชื่อมต่อ ถ้าคุณไม่ได้ตั้งค่า *rfc1323* บนทั้งสองด้านของ การเชื่อมต่อ ค่าที่บังคับใช้สำหรับ *tcp_recvspace* ที่ปรับได้จะเป็น 65536 อ็อพชันนี้เพิ่มไบต์เพิ่มเติมอีก 12 ไบต์ลงในส่วนหัวโปรโตคอล TCP ซึ่งจะหักออกจากข้อมูล payload ของผู้ใช้ ดังนั้นบนอะแด็ปเตอร์ MTU ขนาดเล็ก อ็อพชันนี้จึงอาจส่งผลกระทบต่อประสิทธิภาพเล็กน้อย

ถ้าคุณกำลังส่งข้อมูลผ่านทางอะแด็ปเตอร์ที่มีขนาด MTU ใหม่ (เช่น 32 K หรือ 64 K) ประสิทธิภาพการสตรีม TCP อาจไม่เต็มที่ ยกเว้นว่ามีการเปิดใช้งาน อ็อพชันนี้เนื่องจากแพ็กเก็ตหนึ่งจะใช้ขนาดหน้าต่าง TCP ทั้งหมด ดังนั้น TCP จึงไม่สามารถสตรีมหลายแพ็กเก็ตได้เนื่องจากจะต้องรอ การยอมรับ TCP และการอัพเดตหน้าต่างจากเครื่องรับสำหรับ แต่ละแพ็กเก็ต โดยการเปิดใช้งานอ็อพชัน *rfc1323* โดยใช้คำสั่ง **no -o rfc1323=1** ขนาดหน้าต่างของ TCP สามารถตั้งค่าได้สูงถึง 4 GB หลังจากตั้งค่าอ็อพชัน *rfc1323* เป็น 1 แล้ว คุณสามารถเพิ่มพารามิเตอร์ *tcp_recvspace* เป็นค่าที่สูงขึ้นมาก เช่น 10 เท่าของขนาด MTU ได้

ถ้าระบบการส่งและการรับไม่สนับสนุนอ็อพชัน *rfc1323* การลดขนาด MTU เป็นวิธีหนึ่งในการปรับปรุงประสิทธิภาพการสตรีมสำหรับ อะแด็ปเตอร์ MTU ขนาดใหญ่ ตัวอย่างเช่น แทนการใช้ขนาด MTU เป็น 65536 ซึ่งจำกัด TCP ให้มีแพ็กเก็ตที่ค้างอยู่เพียงแพ็กเก็ตเดียว ให้เลือกขนาด MTU ที่เล็กลงเป็น 16384 เพื่อให้ TCP มีแพ็กเก็ตที่ค้างอยู่จำนวน 4 แพ็กเก็ตพร้อมกับค่า *tcp_recvspace* เป็น 65536 ไบต์ ซึ่งช่วยให้ประสิทธิภาพดีขึ้น อย่างไรก็ตาม โหนดทั้งหมดบนเครือข่าย ต้องใช้ขนาด MTU เดียวกัน

การค้นหา MTU ของ TCP พาธ:

อ็อพชันโปรโตคอลการค้นหา MTU ของ TCP พาธมีการเปิดใช้งานโดยค่าดีฟอลต์ ใน AIX อ็อพชันนี้ช่วยให้โปรโตคอลสแต็คทราบขนาด MTU ต่ำสุดบนเครือข่ายใดๆ ที่อยู่ในพาธระหว่างสองโฮสต์ ในปัจจุบัน และมีการควบคุมโดยอ็อพชันเครือข่าย **tcp_pmtu_discover=1**

การใช้การค้นหา TCP Path MTU จะใช้แพ็กเก็ต TCP ของ การเชื่อมต่อเองแทนข้อความ ICMP ECHO ส่วนขยายเคอร์นัล TCP/IP เก็บรักษาตาราง ที่เรียกว่าตาราง PMTU เพื่อจัดเก็บข้อมูลการค้นหา PMTU ที่เกี่ยวข้อง รายการสำหรับปลายทางแต่ละแห่งมีการสร้างขึ้นในตาราง PMTU เมื่อสร้างการ เชื่อมต่อ TCP ไปยังปลายทางนั้น ค่า PMTU คือค่าอินเตอร์เฟส MTU ขาออก

แพ็กเก็ต TCP มีการส่งโดยมีชุดบิต Don't Fragment, หรือ DF, ในส่วนหัว IP แพ็กเก็ต TCP เข้าถึงเราเตอร์เครือข่ายที่มีค่า MTU ที่น้อยกว่า ขนาดของแพ็กเก็ต TCP เราเตอร์ส่งกลับข้อความแสดงข้อผิดพลาด ICMP ซึ่งบ่งชี้ว่าข้อความไม่สามารถส่งต่อได้เนื่องจากไม่สามารถแบ่ง เฟรกเมนต์ได้ ถ้าเราเตอร์ที่ส่งข้อความแสดงข้อผิดพลาดคอมไพล์ด้วย RFC 1191 ค่า MTU ของเครือข่ายจะมีอยู่ในข้อความแสดงข้อผิดพลาด ICMP มิฉะนั้น สำหรับแพ็กเก็ต TCP ที่จะส่งผ่านใหม่ ค่าที่น้อยกว่าของขนาด MTU ต้องถูกกำหนด จากตารางค่า MTU ที่รู้จักกันดีภายในส่วนขยายเคอร์นัล TCP/IP AIX จากนั้น อัพเดตค่า PMTU ของปลายทางในตาราง PMTU ด้วยขนาด MTU ที่เล็กกว่าและส่งผ่านแพ็กเก็ต TCP ใหม่ การเชื่อมต่อ TCP ในลำดับต่อมาใดๆ ไปยังปลายทางนั้นใช้ค่า PMTU ที่อัพเดต

คุณสามารถใช้คำสั่ง **pmtu** เพื่อดูหรือลบรายการ PMTU ข้อมูลต่อไปนี้เป็นตัวอย่างของคำสั่ง **pmtu**:

```
# pmtu display

  dst           gw            If    pmtu      refcnt    redisc_t    exp

--------------------------------------------------------------------------
```

```
10.10.1.3     10.10.1.5         en1     1500        2          9        0

10.10.2.5     10.10.2.33        en0     1500        1          0        0
```

รายการ PMTU ที่ไม่ได้ใช้ซึ่งเป็นรายการ `refcnt` ที่มีค่าเป็น 0 จะถูกลบออกเพื่อป้องกันไม่ให้ตาราง PMTU ใหญ่เกินไป รายการที่ไม่ได้ใช้ จะถูกลบออกหลังผ่านไป *pmtu_expire* นาทีตั้งแต่ค่า `refcnt` เท่ากับ 0 อ็อพชันเครือข่าย *pmtu_expire* มีค่าดีฟอลต์เป็น 10 นาที เพื่อป้องกันไม่ให้รายการ PMTU หมดอายุ คุณสามารถตั้งค่า *pmtu_expire* เป็น 0

Route cloning ไม่จำเป็นสำหรับการใช้การค้นหา MTU ของ TCP พาธ ซึ่งหมายความว่าตารางการเราต์จะเล็กลงและจัดการได้มากขึ้น

tcp_nodelayack ที่ปรับได้:

อ็อพชัน *tcp_nodelayack* พร้อมต์ TCP ให้ส่งการตอบรับทันที แทนที่จะถ่วงเวลาออกไป 200 ms ซึ่งเป็นค่าปกติ การส่งการตอบรับทันทีอาจทำให้โอเวอร์เฮดสูงขึ้นเล็กน้อย แต่ในบางกรณี สามารถช่วยปรับปรุงประสิทธิภาพได้มาก

ปัญหาประสิทธิภาพพบเมื่อ TCP ถ่วงเวลาการส่งการตอบรับออกไป 200 ms เนื่องจากผู้ส่งกำลังรอการตอบรับจากผู้รับ และผู้รับกำลังรอข้อมูลเพิ่มเติมจากผู้ส่ง ซึ่งอาจส่งผลให้การสตรีม ผลผลิตต่ำ ถ้าคุณสงสัยปัญหานี้ คุณควรเปิดใช้งานอ็อพชัน *tcp_nodelayack* เพื่อดูว่าอ็อพชันช่วยปรับปรุง ประสิทธิภาพการสตรีมหรือไม่ ถ้าไม่ช่วย ให้ปิดใช้งานอ็อพชัน *tcp_nodelayack*

sb_max ที่ปรับได้:

sb_max ที่ปรับได้ตั้งค่าขีดจำกัดสูงสุด ของจำนวนซ็อกเก็ตบัฟเฟอร์ที่จัดคิวสำหรับแต่ละซ็อกเก็ต ซึ่งควบคุม จำนวนพื้นที่ว่างบัฟเฟอร์ที่ใช้โดยบัฟเฟอร์ที่จัดคิวสำหรับซ็อกเก็ตของผู้ส่ง หรือซ็อกเก็ตของผู้รับ

ระบบพิจารณาซ็อกเก็ตบัฟเฟอร์ที่ใช้ตามขนาดของบัฟเฟอร์ ไม่ใช่เนื้อหาของบัฟเฟอร์

ถ้าไดรเวอร์อุปกรณ์วางข้อมูล 100 ไบต์ไว้ในบัฟเฟอร์ขนาด 2048 ไบต์ ระบบจะ พิจารณาจำนวน 2048 ไบต์เป็นพื้นที่ว่างซ็อกเก็ตบัฟเฟอร์ที่จะใช้ เป็นเรื่องทั่วไปสำหรับ ไดรเวอร์อุปกรณ์ที่จะได้รับบัฟเฟอร์เข้าในบัฟเฟอร์ที่มีขนาดใหญ่เพียงพอที่จะได้รับแพ็กเก็ตขนาดสูงสุดของอะแด็ปเตอร์ ซึ่งมักทำให้สิ้นเปลืองพื้นที่ว่างบัฟเฟอร์ แต่ไดรเวอร์อุปกรณ์อาจต้องการรอบ CPU มากขึ้นเพื่อคัดลอกข้อมูลไปยังบัฟเฟอร์ที่เล็กลง

หมายเหตุ: ใน AIX ค่าดีฟอลต์ สำหรับ *sb_max* ที่ปรับได้คือ 1048576 ซึ่งถือว่าเป็น ขนาดใหญ่ โปรดดู การปรับเวิร์กโหลดการสตรีม TCP สำหรับค่า *sb_max* ที่แนะนำ ถ้าคุณต้องการเปลี่ยนพารามิเตอร์นี้

TCP checksum offload:

อ็อพชัน TCP checksum offload ช่วยให้อะแด็ปเตอร์เครือข่ายสามารถ คำนวณ TCP checksum ในการส่งผ่านและการรับ ซึ่งทำให้ AIX host CPU ไม่ต้องคำนวณ checksum

ประโยชน์แตกต่างกันไปตามขนาดแพ็กเก็ต แพ็กเก็ตขนาดเล็กได้รับประโยชน์น้อยมากหรือไม่ได้รับ ประโยชน์จากอ็อพชันนี้เลย ในขณะที่แพ็กเก็ตขนาดใหญ่ได้รับประโยชน์มากกว่า บนอะแด็ปเตอร์ PCI-X GigE ประโยชน์สำหรับ MTU 1500 โดยปกติแล้ว คือการลดการใช้ CPU ลง ประมาณ 5% และสำหรับ MTU 9000 (Jumbo Frames) คือการลดการใช้ CPU ลงประมาณ 15%

TCP streaming ผลผลิตด้วย MTU 1500 ช้าลงบนเครื่องที่มีตัวประมวลผล เร็วกว่า 400 MHz ถ้าเปิดใช้งานอ็อพชัน TCP checksum offload เนื่องจากระบบโฮสต์สามารถรัน checksum ได้เร็วว่าอะแด็ปเตอร์กิกะบิต อีเทอร์เน็ต PCI, FC2969 และ FC 2975 ดังนั้น โดยค่าดีฟอลต์ อ็อพชันนี้จึงเป็น off บนอะแด็ปเตอร์เหล่านี้ เมื่ออะแด็ปเตอร์เหล่านี้ใช้ jumbo frames อะแด็ปเตอร์สามารถรันที่ ความเร็วของการเชื่อมต่อแม้ว่าต้องคำนวณ checksum

อะแด็ปเตอร์ PCI-X กิกะบิตอีเทอร์เน็ตสามารถรันที่ความเร็วของการเชื่อมต่อด้วยอ็อพชัน TCP checksum offload ที่เปิดใช้งานและลดการประมวลผลโฮสต์ CPU ดังนั้น จึงมีการเปิดใช้งานโดยค่าดีฟอลต์

TCP large send offload:

อ็อพชัน TCP large send offload อนุญาตให้เลเยอร์ AIX TCP สร้าง ข้อความ TCP ได้ยาวสูงสุด 64 KB อะแด็ปเตอร์ส่งข้อความในการเรียกใช้ หนึ่งครั้งต่ำลงไปในสแต็กผ่าน IP และไดร์เวอร์อุปกรณ์ Ethernet

จากนั้นอะแด็ปเตอร์แบ่งข้อความออกเป็นหลายเฟรม TCP เพื่อ ส่งข้อมูลบนสายเคเบิล แพ็กเก็ต TCP ที่ส่งบนสายเคเบิลอาจเป็นเฟรม 1500 ไบต์สำหรับ Media Transmission Unit (MTU) ของ 1500 หรือสูงสุดเฟรม 9000 ไบต์สำหรับ MTU ของ 9000 (เฟรมขนาดใหญ่มาก)

หากไม่มีอ็อพชัน TCP large send offload เพื่อให้อ็อพชัน TCP ส่งข้อมูล 64 KB จะใช้การเรียกใช้ 44 ครั้งลงไปในสแต็กโดยใช้แพ็กเก็ต 1500 ไบต์ ด้วยอ็อพชัน TCP large send อ็อพชัน TCP สามารถส่ง ข้อมูลได้สูงสุด 64K ไบต์ในหนึ่งการเรียกใช้ลงไปในสแต็ก ซึ่งลด การประมวลผลโฮสต์ และผลลัพธ์ในการใช้งานตัวประมวลผลต่ำลงบน ตัวประมวลผลโฮสต์ จากนั้นอะแด็ปเตอร์อีเทอร์เน็ตทำการออฟโหลดการแบ่งเซกเมนต์ TCP เป็นแบ่งเซกเมนต์ข้อมูลเป็นแพ็กเก็ตขนาด MTU (โดยปกติ 1500 ไบต์) ประโยชน์แตกต่างกันไปขึ้นอยู่กับขนาด TCP large send เฉลี่ย ตัวอย่างเช่น การลด CPU ตัวประมวลผลโฮสต์ลง 60 ถึง 75% สามารถทำได้ด้วยอะแด็ปเตอร์ PCI-eXtended (PCI-X) Gigabit Ethernet ที่มีขนาด MTU เป็น 1500 สำหรับเฟรมขนาดใหญ่มาก (MTU 9000) จะ ประหยัดได้น้อยเนื่องจากระบบส่งเฟรมขนาดที่ใหญกว่าอยู่แล้ว ตัวอย่าง เช่น การลด CPU ตัวประมวลผลโฮสต์ลง 40% โดยปกติด้วยเฟรม ขนาดใหญ่มาก

อ็อพชัน large send offload ถูกเปิดใช้งานโดยดีฟอลต์ บนอะแด็ปเตอร์อีเทอร์เน็ตที่สนับสนุนอ็อพชันเมื่อคุณทำกลังทำงานในโหมดงานเฉพาะ อ็อพชันนี้เพิ่มประสิทธิภาพการทำงานบน 10 Gigabit Ethernet และอะแด็ปเตอร์ทำงานเร็วขึ้นสำหรับเวิร์กโหลดที่จัดการการสตรีมข้อมูล (เช่น file transfer protocol (FTP), RCP, การสำรองข้อมูลเทป และแอ็พพลิเคชัน การย้ายข้อมูลจำนวนมากที่คล้ายๆ กัน) อุปกรณ์อะแด็ปเตอร์อีเทอร์เน็ตเสมือน และ shared Ethernet adapter (SEA) ถือเป็นข้อยกเว้นโดยอ็อพชัน large send offload ถูกปิดใช้งานโดยดีฟอลต์เนื่องจากปัญหาความสามารถในการ ทำงานกับระบบปฏิบัติการ Linux หรือ IBM i การเปิดใช้งาน Large Send และคุณลักษณะประสิทธิภาพการทำงานอื่นๆ สามารถ ทำใน AIX และอะแด็ปเตอร์ อีเทอร์เน็ตเสมือน หรือสภาวะแวดล้อม SEA

อ็อพชัน large send เป็นแอ็ตทริบิวต์อุปกรณ์ที่ ถูกระบุเป็น large_send คุณสามารถเห็นแอ็ตทริบิวต์ อุปกรณ์ send offload โดยใช้คำสั่งต่อไปนี้โดย X คือหมายเลขอุปกรณ์:

```
lsattr -E -l entX
```

อ็อพชันการกำจัดของอะแด็ปเตอร์:

บางอะแด็ปเตอร์นำเสนออ็อพชันที่สามารถเปิดใช้งานหรือปิดใช้งาน งานออฟโหลดจากระบบ AIX ไปยังอะแด็ปเตอร์

ตารางที่ 7. อะแด็ปเตอร์และอ็อพชันที่พร้อมใช้งาน และค่ากำหนดดีฟอลต์ของระบบ

ชนิดของอะแด็ปเตอร์	ฟีเจอร์โค้ด	การกำจัดเช็คซัม TCP	ค่ากำหนดดีฟอลต์	TCP การส่งขนาดใหญ่	ค่ากำหนดดีฟอลต์
GigE, PCI, SX & TX	2969, 2975	ใช่	OFF	ใช่	OFF
GigE, PCI-X, SX และ TX	5700, 5701	ใช่	ON	ใช่	ON
GigE dual port PCI-X, TX และ SX	5706, 5707	ใช่	ON	ใช่	ON
10 GigE PCI-X LR และ SR	5718, 5719	ใช่	ON	ใช่	ON
อีเทอร์เน็ต 10/100	4962	ใช่	ON	ใช่	OFF
ATM 155, UTP & MMF	4953, 4957	ใช่ (ส่งอย่างเดียว)	ON	ไม่ใช่	N/A
ATM 622, MMF	2946	ใช่	ON	ไม่ใช่	N/A

การปรับเวิร์กโหลดการร้องขอและการตอบกลับ TCP

เวิร์กโหลดการร้องขอและการตอบกลับ TCP คือเวิร์กโหลดที่เกี่ยวข้องกับ การแลกเปลี่ยนข้อมูลสองทาง

ตัวอย่างของเวิร์กโหลดการร้องขอและการตอบกลับคือแอ็พพลิเคชันหรือไคลเอ็นต์/เซิร์ฟเวอร์แอ็พพลิเคชันชนิด Remote Procedure Call (RPC) เช่น การร้องขอเว็บเบราเซอร์ ไปยังเว็บเซิร์ฟเวอร์ ระบบไฟล์ NFS (ที่ใช้ TCP เป็นโปรโตคอลการขนส่ง) หรือโปรโตคอลการจัดการล็อกของฐานข้อมูล การร้องขอดังกล่าวมักเป็นข้อความขนาดเล็ก และการตอบกลับที่ใหญ่กว่า แต่ยังอาจเป็นการร้องขอขนาดใหญ่และการตอบกลับขนาดเล็กได้ด้วย

ส่วนประกอบหลักที่น่าสนใจในเวิร์กโหลดเหล่านี้คือเวลาแฝงในการเดินทางไปกลับ ของเครือข่าย การร้องขอหรือการตอบกลับเหล่านี้จำนวนมากใช้ข้อความขนาดเล็ก ดังนั้นแบนด์วิธเครือข่ายจึงไม่ใช่เรื่องสำคัญ

ฮาร์ดแวร์มีผลกระทบอย่างมากต่อเวลาแฝง ตัวอย่างเช่น ชนิดของเครือข่าย ชนิดและประสิทธิภาพของสวิตช์เครือข่ายหรือเราต์เตอร์ ความเร็วของ ตัวประมวลผลที่ใช้ในแต่ละโหนดของเครือข่าย เวลาแฝงของอะแด็ปเตอร์และบัส ทั้งหมดนี้ล้วนมีผลกระทบต่อเวลาในการเดินทางไปกลับ

การปรับอ็อพชันเพื่อให้เวลาแฝงต่ำสุด (การตอบกลับที่ดีที่สุด) โดยปกติแล้ว ทำให้ CPU โอเวอร์เฮดสูงขึ้นเนื่องจากระบบส่งแพ็กเก็ตมากขึ้น ได้รับการขัดจังหวะ มากขึ้น เป็นต้น เพื่อลดเวลาแฝงและเวลาการตอบกลับให้เหลือน้อยที่สุด นี่เป็นการแลกเปลี่ยน ประสิทธิภาพที่เป็นจริงตลอดกาล

ตัวปรับหลักสำหรับแอ็พพลิเคชันการร้องขอและการตอบกลับมีดัง ต่อไปนี้:

- *tcp_nodelay* หรือ *tcp_nagle_limit*
- *tcp_nodelayack*
- ค่าติดตั้ง ขัดจังหวะการรวมตัว ของอะแด็ปเตอร์

หมายเหตุ: เวิร์กโหลดการร้องขอ/การตอบกลับบางรายการมีข้อมูลจำนวนมากใน ทิศทางเดียว เวิร์กโหลดประเภทนี้อาจต้องมีการปรับชุดของการสตรีม และเวลาแฝง ขึ้นอยู่กับเวิร์กโหลด

อ็อพชัน tcp_nodelay or tcp_nagle_limit:

ใน AIX อ็อพชันซ็อกเก็ต TCP_NODELAY มีการปิดใช้งานโดยค่าดีฟอลต์ ซึ่งอาจทำให้เกิดความล่าช้าอย่างมากสำหรับเวิร์กโหลด การร้องขอ/การตอบกลับ ที่อาจส่งข้อมูลเพียงสองสามไบต์แต่ต้องเสียเวลารอการตอบกลับ TCP ใช้การตอบรับที่ล่าช้า เนื่องจากหวังจะรออาศัยการตอบรับ TCP บนแพ็กเก็ตการตอบกลับ โดยปกติ ความล่าช้านาน 200 ms

การใช้ TCP ส่วนใหญ่ใช้ขั้นตอนวิธี nagle ที่การเชื่อมต่อ TCP สามารถมีเซกเมนต์ขนาดเล็กที่ค้างอยู่ซึ่งยังไม่ได้รับการตอบรับได้เพียงหนึ่งเซกเมนต์เท่านั้น ลักษณะนี้ส่งผลให้ TCP ถ่วงเวลาการส่งแพ็กเก็ตเพิ่มเติมจนกว่าจะได้รับ การตอบรับ หรือจนกว่าจะสามารถรวมกลุ่มข้อมูลเพิ่มเติมและส่งเซกเมนต์ ขนาดเต็มได้

แอ็พพลิเคชันที่ใช้เวิร์กโหลดการร้องขอ/การตอบกลับควรใช้การเรียก **setsockopt( )** เพื่อเปิดใช้งานอ็อพชัน TCP_NODELAY ตัวอย่างเช่น ยูทิลิตี **telnet** และ **rlogin,** Network File System (NFS), และเว็บเซิร์ฟเวอร์ใช้อ็อพชัน TCP_NODELAY อยู่แล้ว เพื่อปิดใช้งาน nagle อย่างไรก็ตาม บางแอ็พพลิเคชันไม่ได้ใช้อ็อพชันนี้ ซึ่งอาจส่งผลให้ประสิทธิภาพไม่ดีขึ้นอยู่กับขนาด MTU เครือข่ายและขนาดของการส่ง (การบันทึก) ไปยังซ็อกเก็ต

เมื่อจัดการกับแอ็พพลิเคชันที่ไม่ได้เปิดใช้งาน TCP_NODELAY คุณสามารถ ใช้อ็อพชันการปรับดังต่อไปนี้เพื่อปิดใช้งาน nagle:

- **tcp_nagle_limit**
- อ็อพชัน **tcp_nodelay** ISNO
- **tcp_nodelayack**
- **fasttimo**
- ขัดจังหวะการรวมตัวบนอะแด็ปเตอร์

อ็อพชัน tcp_nagle_limit:

อ็อพชันเครือข่าย *tcp_nagle_limit* เป็นอ็อพชันเครือข่าย สากล และมีการตั้งค่าเป็น 65536 โดยค่าดีฟอลต์

TCP ปิดใช้งานขั้นตอนวิธี nagle สำหรับเซกเมนต์ที่มีขนาดเท่ากับหรือใหญ่กว่า ค่านี้ ดังนั้นคุณสามารถปรับ threshold ซึ่งเปิดใช้งาน nagle ได้ ตัวอย่างเช่น เมื่อต้องการปิดใช้งาน nagle ทั้งหมด ให้ตั้งค่า *tcp_nagle_limit* เป็น 1 เมื่อต้องการอนุญาตให้ TCP รวมการส่งและส่งแพ็กเก็ตที่มีไบต์อย่างน้อย 256 ไบต์ ให้ตั้งค่า *tcp_nagle_limit* เป็น 256

อ็อพชัน tcp_nodelay ISNO:

ที่ระดับอินเตอร์เฟส มีอ็อพชัน *tcp_nodelay* ISNO เพื่อเปิดใช้งาน TCP_NODELAY

การตั้งค่า *tcp_nodelay* เป็น 1 ส่งผลให้ TCP ซึ่งปิดใช้งาน nagle ไม่ล่าช้า และส่งแต่ละแพ็กเก็ตของ แต่ละการส่งหรือการบันทึกแอ็พพลิเคชัน

อ็อพชัน tcp_nodelayack:

คุณสามารถใช้อ็อพชันเครือข่าย *tcp_nodelayack* เพื่อปิดใช้งานการตอบรับที่ล่าช้า ซึ่งโดยปกติ นาน 200 ms

การไม่ถ่วงเวลาการตอบรับสามารถลดเวลาแฝงและช่วยให้ผู้ส่ง (ซึ่งอาจเปิดใช้งาน nagle) ได้รับการตอบรับ แล้วส่งเซกเมนต์บางส่วนถัดไปได้เร็วขึ้น

อ็อพชัน fasttimo:

คุณสามารถใช้เน็ตเวิร์กอ็อพชัน **fasttimo** เพื่อลดตัวจับเวลาลงให้เหลือ 100 หรือ 50 มิลลิวินาที จาก 200 มิลลิวินาที ซึ่งเป็นค่าดีฟอลต์

เนื่องจาก TCP ใช้ตัวจับเวลานี้สำหรับฟังก์ชันอื่นๆ ที่ตัวจับเวลาทำไว้สำหรับการเปิดการเชื่อมต่อ TCP ทั้งหมด การลดตัวจับเวลานี้จะเพิ่มการใช้งานให้กับระบบมากขึ้น เนื่องจากการเชื่อมต่อ TCP ทั้งหมดจะมีการสแกนที่บ่อยขึ้น อ็อพชันข้างต้นเป็นตัวเลือกที่ดีที่สุด และคุณควรใช้อ็อพชัน **fasttimo** เป็นตัวช่วยสุดท้ายในการปรับระบบ

การรวมอินเตอร์รัปต์:

หากต้องการหลีกเลี่ยงการลอยตัวของระบบโฮสต์ด้วยอินเตอร์รัปต์ แพ็กเก็ต จำนวนมากที่ถูกเก็บรวบรวม และอินเตอร์รัปต์เดี่ยวที่ถูกสร้างขึ้นสำหรับแพ็กเก็ตจำนวนมาก เหตุการณ์นี้เรียกว่า *การรวมอินเตอร์รัปต์*

สำหรับการดำเนินการรับ อินเตอร์รัปต์จะแจ้งให้โฮสต์ CPU ที่แพ็กเก็ตได้รับบนคิวอินพุตของอุปกรณ์โดยไม่มีรูปแบบของอินเตอร์รัปต์ตรรกะระดับกลางบนอะแด็ปเตอร์ ซึ่งอาจนำไปสู่การอินเตอร์รัปต์สำหรับแพ็กเก็ตขาเข้าแต่ละชุด อย่างไรก็ตาม เนื่องจากอัตราแพ็กเก็ตขาเข้าเพิ่มขึ้น ไดร์เวอร์อุปกรณ์จะเสร็จสิ้น การประมวลผลหนึ่งแพ็กเก็ต และตรวจสอบว่า แพ็กเก็ตอยู่บนคิวการรับ ก่อนที่จะออกจากไดร์เวอร์และล้างข้อมูลอินเตอร์รัปต์ จากนั้น ไดร์เวอร์ค้นหาว่า มีแพ็กเก็ตเพิ่มเติมที่ต้องจัดการและจบการจัดการกับแพ็กเก็ตจำนวนมากต่ออินเตอร์รัปต์ ซึ่งเป็นอัตราแพ็กเก็ตที่เพิ่มขึ้น นั่นหมายความว่า ระบบมีประสิทธิภาพมากขึ้นเนื่องจากโหลดเพิ่มขึ้น

อย่างไรก็ตาม อะแด็ปเตอร์บางตัวมีคุณลักษณะเพิ่มเติมที่สามารถจัดเตรียมการควบคุม เมื่ออินเตอร์รัปต์ที่ได้รับถูกสร้างขึ้น สิ่งนี้เรียกว่า การรวมอินเตอร์รัปต์ หรืออินเตอร์รัปต์ตรรกะระดับกลาง ซึ่งอนุญาตให้หลายๆ แพ็กเก็ต ที่ต้องรับ และสร้างหนึ่งอินเตอร์รัปต์สำหรับหลายๆ แพ็กเก็ต ตัวจับเวลาจะเริ่มต้น เมื่อแพ็กเก็ตแรกมาถึง จากนั้น อินเตอร์รัปต์จะถูกหน่วงเวลา *n* มิลลิวินาที หรือจนกว่าจะ *m* แพ็กเก็ตจะมาถึง เมธอดจะผันแปรตามอะแด็ปเตอร์ ซึ่งเป็นคุณลักษณะของไดร์เวอร์อุปกรณ์ที่อนุญาตให้ผู้ใช้ควบคุม

ภายใต้โหลด การรวมอินเตอร์รัปต์จะเพิ่มเวลาแฝงให้กับเวลาที่แพ็กเก็ตมาถึง แพ็กเก็ตจะอยู่ในโฮสต์หน่วยความจำ แต่โฮสต์จะไม่รู้จักแพ็กเก็ต จนกระทั่งเวลาผ่านไป อย่างไรก็ตาม ภายใต้โหลดแพ็กเก็ตที่สูงกว่า ระบบจะดำเนินการได้อย่างมีประสิทธิภาพเพิ่มขึ้นโดยใช้วงรอบ CPU เพียงเล็กน้อย เนื่องจากอินเตอร์รัปต์จำนวนน้อย จะถูกสร้างขึ้น และประมวลผลโฮสต์ในหลายๆ แพ็กเก็ตต่ออินเตอร์รัปต์

สำหรับอะแด็ปเตอร์ AIX ที่สอดแทรกคุณลักษณะระดับกลางของอินเตอร์รัปต์ คุณควรตั้งค่าให้มีระดับกลาง เพื่อลดการใช้อินเตอร์รัปต์โดยไม่เพิ่มจำนวนของเวลาแฝงเพิ่มเติม สำหรับแอ็พพลิเคชันที่อาจต้องมีเวลาแฝงจำนวนน้อยที่สุด คุณควรปิดหรือเปลี่ยนอ็อพชัน เพื่ออนุญาตให้อินเตอร์รัปต์เพิ่มขึ้นต่อวินาทีสำหรับเวลาแฝงที่ลดลง

อะแด็ปเตอร์ Gigabit Ethernet นำเสนอคุณลักษณะระดับกลางของอินเตอร์รัปต์ อะแด็ปเตอร์ FC 2969 และ FC 2975 GigE PCI จะมีเมธอดของค่าหน่วงเวลา และจำนวนบัฟเฟอร์ อะแด็ปเตอร์จะเริ่มต้นตัวจับเวลา เมื่อแพ็กเก็ตแรกมาถึง และอินเตอร์รัปต์จะเกิดขึ้นเมื่อตัวจับเวลาหมดอายุ หรือเมื่อ *n* บัฟเฟอร์ในโฮสต์ที่ถูกใช้

อะแด็ปเตอร์ FC 5700, FC 5701, FC 5706 และ FC 5707 GigE PCI-X ใช้เมธอดของอัตราอินเตอร์รัปต์ที่ช้าลง ซึ่งสร้างอินเตอร์รัปต์ที่ความถี่ที่ระบุ ซึ่งอนุญาตสำหรับจัดกลุ่มของแพ็กเก็ตตามเวลา อัตราอินเตอร์รัปต์ดีฟอลต์คือ 10 000 อินเตอร์รัปต์ต่อวินาที สำหรับการใช้อินเตอร์รัปต์ที่ต่ำ คุณสามารถตั้งค่าอัตราอินเตอร์รัปต์ให้มีค่าต่ำสุด 2 000 อินเตอร์รัปต์ต่อวินาที สำหรับเวิร์กโหลดที่เรียกเวลาแฝงต่ำและเวลาตอบสนองเร็วขึ้น คุณสามารถตั้งค่าอัตราอินเตอร์รัปต์ให้มีค่าสูงสุด 20 000 อินเตอร์รัปต์ การตั้งค่าอัตราอินเตอร์รัปต์ ให้มีค่า 0 จะปิดใช้งานการอินเตอร์รัปต์อย่างสมบูรณ์

อะแด็ปเตอร์ 10 Gigabit Ethernet PCI-X (FC 5718 and 5719) มีอ็อพชันการรวมอินเตอร์รัปต์ (*rx_int_delay*) ที่มีการหน่วงเวลา 0.82 มิลลิวินาที ความยาวแท้จริงของเวลาหน่วงจะถูกพิจารณาโดยการคูณ 0.82 ด้วยค่าที่ตั้งค่าอยู่ใน *rx_int_delay* อ็อพชันนี้จะถูกปิดใช้งาน ตามค่าดีฟอลต์ (*rx_int_delay*=0) เนื่องจากการทดสอบที่สรุปว่า อัตราอินพุตที่สูงกว่าของอะแด็ปเตอร์เหล่านี้ การรวมอินเตอร์รัปต์จะไม่ช่วยในเรื่องของผลการทำงาน.

ตารางที่ 8. คุณสมบัติของอะแด็ปเตอร์ 10 Gigabit Ethernet PCI-X

ชนิดอะแด็ปเตอร์	โค้ดคุณลักษณะ	แอ็ตทริบิวต์ ODM	ค่าดีฟอลต์	ขอบเขต
10 Gigabit Ethernet PCI-X (LR or SR)	5718, 5719	rx_int_delay	0	0-512

การปรับ UDP

User Datagram Protocol (UDP) คือ datagram protocol ที่ใช้โดย Network File System (NFS), เซิร์ฟเวอร์ชื่อ (named), Trivial File Transfer Protocol (TFTP), และโปรโตคอลวัตถุประสงค์พิเศษอื่น

เนื่องจาก UDP คือ datagram protocol ข้อความทั้งหมด (datagram) ต้อง ถูกคัดลอกเข้าในเคอร์นัลในการดำเนินงานส่งเป็นหนึ่ง atomic การดำเนินงาน Datagram ยังมีการรับเป็นหนึ่งข้อความที่สมบูรณ์บนการเรียกระบบ **recv** หรือ **recvfrom** ด้วย คุณต้องตั้งค่าพารามิเตอร์ *udp_sendspace* และ *udp_recvspace* เพื่อจัดการความต้องการบัฟเฟอร์สำหรับแต่ละซ็อกเก็ต

UDP datagram ใหญ่สุดที่สามารถส่งได้คือ 64 KB บวกขนาดส่วนหัว UDP (8 ไบต์) และขนาดส่วนหัว IP (20 ไบต์สำหรับ IPv4 หรือ 40 ไบต์สำหรับส่วนหัว IPv6)

ค่าที่ปรับได้ต่อไปนี้มีผลกระทบต่อประสิทธิภาพ UDP:

- *udp_sendspace*
- *udp_recvspace*
- UDP packet chaining
- อะแด็ปเตอร์อ็อพชัน เช่น ขัดจังหวะการรวมตัว

udp_sendspace ที่ปรับได้:

ตั้งค่า udp_sendspace ที่ปรับได้ เป็นค่าที่เท่ากับหรือมากกว่า UDP datagram ที่ใหญ่ที่สุด ซึ่งจะส่ง

เพื่อให้ง่าย ควรตั้งค่าพารามิเตอร์นี้เป็น 65536 ซึ่งใหญ่เพียงพอจะจัดการ UDP packet ที่ใหญ่ที่สุดซึ่งเป็นไปได้ ไม่มีประโยชน์ที่จะตั้งค่านี้ให้ ใหญ่ขึ้น

udp_recvspace ที่ปรับได้:

udp_recvspace ที่ปรับได้ควบคุมจำนวนของ พื้นที่ว่างสำหรับข้อมูลขาเข้าที่จัดคิวบนแต่ละ UDP socket หลังจากใช้งานถึงขีดจำกัด *udp_recvspace* สำหรับซ็อกเก็ตแล้ว แพ็กเก็ตขาเข้าจะถูกทิ้งไป

สถิติของแพ็กเก็ตที่ทิ้งมีการแสดงรายละเอียดในเอาต์พุตคำสั่ง **netstat -p udp** ภายใต้คอลัมน์ `socket buffer overflows` หากต้องการข้อมูลเพิ่มเติม ให้ดู คำสั่ง netstat ใน *Commands Reference, Volume 4*

คุณควรจะตั้งค่า *udp_recvspace* ให้สูง เนื่องจากข้อเท็จจริงที่ว่า หลาย UDP datagrams อาจมาถึงและรอซ็อกเก็ต ที่จะให้แอ็พพลิเคชันอ่าน นอกจากนี้ UDP แอ็พพลิเคชันจำนวนมากใช้ซ็อกเก็ตเฉพาะเพื่อรับแพ็กเก็ต ซ็อกเก็ตนี้ใช้เพื่อรับแพ็กเก็ตจากจากไคลเอ็นต์ทั้งหมดที่พูดคุยกับเซิร์ฟเวอร์แอ็พพลิเคชัน ดังนั้น พื้นที่ว่างการรับ ต้องใหญ่พอที่จะจัดการกับ datagrams

จำนวนมากที่อาจมาจาก หลายไคลเอ็นต์ และถูกจัดคิวบนซ็อกเก็ตเพื่อรอที่จะอ่าน ถ้า ค่านี้ต่ำเกินไป แพ็กเก็ตขาเข้าจะถูกทิ้งไป และผู้ส่งต้องส่งผ่านแพ็กเก็ต อีกครั้ง ซึ่งอาจทำให้ประสิทธิภาพไม่ดี

เนื่องจากระบบย่อยการสื่อสารพิจารณาบัฟเฟอร์ที่ใช้ ไม่ใช่เนื้อหาของ บัฟเฟอร์ คุณจึงต้องคำนึงถึงบัฟเฟอร์เมื่อตั้งค่า *udp_recvspace* ตัวอย่างเช่น 8 KB datagram จะถูกแบ่งเฟรกเมนต์เป็น 6 แพ็กเก็ตซึ่งใช้ 6 บัฟเฟอร์การรับ ทั้งหมดนี้จะเป็นบัฟเฟอร์ 2048 ไบต์สำหรับอีเทอร์เน็ต ดังนั้น จำนวนทั้งหมด ของซ็อกเก็ตบัฟเฟอร์ที่ใช้โดยหนึ่ง 8 KB datagram นี้เป็นดังนี้:

```
6*2048=12,288 bytes
```

คุณสามารถเห็นว่าต้องปรับ *udp_recvspace* ให้สูงขึ้น ขึ้นอยู่กับประสิทธิผลของการบัฟเฟอร์ขาเข้า ซึ่งจะ แตกต่างกันไปตาม ขนาด datagram และตามไดรเวอร์อุปกรณ์ การส่ง 64 byte datagram จะใช้บัฟเฟอร์ 2 KB สำหรับแต่ละ 64 byte datagram.

จากนั้น คุณต้องพิจารณาจำนวนของ datagrams ที่อาจจะจัดคิวบนหนึ่งซ็อกเก็ต นี้ ตัวอย่างเช่น เซิร์ฟเวอร์ NFS ได้รับ UDP packets ที่ซ็อกเก็ตซึ่งรู้จักกันดีหนึ่ง จากไคลเอ็นต์ทั้งหมด ถ้าความลึกคิวของซ็อกเก็ตนี้เป็น 30 แพ็กเก็ต คุณจะใช้ 30 * 12, 288 = 368,640 สำหรับ *udp_recvspace* ถ้า NFS ใช้ 8 KB datagrams NFS เวอร์ชัน 3 ใช้ได้มากถึง 32 KB datagrams

ค่าเริ่มต้นที่แนะนำสำหรับ *udp_recvspace* คือ 10 เท่าของค่าของ *udp_sendspace* เนื่องจาก UDP อาจไม่สามารถส่งผ่านแพ็กเก็ตไปยังแอ็พพลิเคชันก่อนอีกแพ็กเก็ตหนึ่งเข้ามา นอกจากนี้ หลายโหนดสามารถส่งไปยังหนึ่งโหนดในเวลาเดียวกัน เพื่อให้มีพื้นที่ว่าง staging ค่านี้จึงมีการกำหนดให้สามารถทับซ้อนได้ 10 แพ็กเก็ตก่อนแพ็กเก็ต ลำดับต่อมาจะถูกทิ้ง สำหรับแอ็พพลิเคชันแบบขนานขนาดใหญ่ที่ใช้ UDP อาจต้อง เพิ่มค่า

หมายเหตุ: ค่า *sb_max* ซึ่งระบุขนาดซ็อกเก็ตบัฟเฟอร์สูงสุดสำหรับซ็อกเก็ตบัฟเฟอร์ใดๆ ควรจะเป็น สองเท่าของขนาดของบัฟเฟอร์การส่งและการรับ UDP และ TCP ที่ใหญ่ที่สุด เป็นอย่างน้อย

UDP packet chaining:

เมื่อ UDP Datagrams ที่จะส่งผ่านใหญ่กว่าขนาดอะแด็ปเตอร์ MTU ชั้น IP protocol จะแบ่งเฟรกเมนต์ datagram เป็นเฟรกเมนต์ขนาด MTU อีเทอร์เน็ตอินเตอร์เฟสมีคุณลักษณะ UPD packet chaining คุณลักษณะนี้มีการเปิดใช้งาน โดยค่าดีฟอลต์ ใน AIX

UDP packet chaining ทำให้ IP สร้างสายโซ่ทั้งหมดของเฟรกเมนต์และ ส่งผ่านสายโซ่นั้นบนไดรเวอร์อุปกรณ์อีเทอร์เน็ตในการเรียกครั้งเดียว การทำเช่นนี้ปรับปรุง ประสิทธิภาพโดยการลดจำนวนการเรียกผ่านทาง ARP และชั้นอินเตอร์เฟส ไปยังไดรเวอร์ และยังลดการเรียก **lock** และ **unlock** ในสภาพแวดล้อม SMP ด้วย อีกทั้งยังช่วย cache affinity ของ code loops การเปลี่ยนแปลง เหล่านี้ลดการใช้ประโยชน์ CPU ของผู้ส่ง

คุณสามารถดูอ็อพชัน UDP packet chaining ด้วยคำสั่ง **ifconfig** ตัวอย่างต่อไปนี้แสดงเอาต์พุตคำสั่ง **ifconfig** สำหรับอินเตอร์เฟส `en0` โดยที่แฟล็ก `CHAIN` บ่งชี้ว่า มีการเปิดใช้งาน packet chaining:

```
# ifconfig en0
en0: flags=5e080863,80<UP,BROADCAST,NOTRAILERS,RUNNING,SIMPLEX,MULTICAST,GROUPRT,64BIT,CHECKSUM_OFFLOAD,PSEG,CHAIN>
         inet 192.1.6.1 netmask 0xffffff00 broadcast 192.1.6.255
         tcp_sendspace 65536 tcp_recvspace 65536 tcp_nodelay 1
```

Packet chaining สามารถปิดใช้งานโดยคำสั่งต่อไปนี้:

```
# ifconfig en0 -pktchain

# ifconfig en0
en0: flags=5e080863,80<UP,BROADCAST,NOTRAILERS,RUNNING,SIMPLEX,MULTICAST,GROUPRT,64BIT,CHECKSUM_OFFLOAD,PSEG>
         inet 192.1.6.1 netmask 0xffffff00 broadcast 192.1.6.255
         tcp_sendspace 65536 tcp_recvspace 65536 tcp_nodelay 1
```

Packet chaining สามารถเปิดใช้งานใหม่ด้วยคำสั่งต่อไปนี้:

```
# ifconfig en0 pktchain

# ifconfig en0
en0: flags=5e080863,80<UP,BROADCAST,NOTRAILERS,RUNNING,SIMPLEX,MULTICAST,GROUPRT,64BIT,CHECKSUM_OFFLOAD,PSEG,CHAIN>
         inet 192.1.6.1 netmask 0xffffff00 broadcast 192.1.6.255
         tcp_sendspace 65536 tcp_recvspace 65536 tcp_nodelay 1
```

การรวมอินเตอร์รัปต์:

หากต้องการหลีกเลี่ยงการลอยตัวของระบบโฮสต์ด้วยอินเตอร์รัปต์ แพ็กเก็ต จำนวนมากที่ถูกเก็บรวบรวม และอินเตอร์รัปต์เดี่ยวที่ถูกสร้างขึ้นสำหรับแพ็กเก็ตจำนวนมาก เหตุการณ์นี้เรียกว่า *การรวมอินเตอร์รัปต์*

สำหรับการดำเนินการรับ อินเตอร์รัปต์จะแจ้งให้โฮสต์ CPU ที่แพ็กเก็ตได้รับบนคิวอินพุตของอุปกรณ์โดยไม่มีรูปแบบของอินเตอร์รัปต์ตรรกะระดับกลางบนอะแด็ปเตอร์ ซึ่งอาจนำไปสู่การอินเตอร์รัปต์สำหรับแพ็กเก็ตขาเข้าแต่ละชุด อย่างไรก็ตาม เนื่องจากอัตราแพ็กเก็ตขาเข้าเพิ่มขึ้น ไดร์เวอร์อุปกรณ์จะเสร็จสิ้น การประมวลผลหนึ่งแพ็กเก็ต และตรวจสอบว่า แพ็กเก็ตอยู่บนคิวการรับ ก่อนที่จะออกจากไดร์เวอร์และล้างข้อมูลอินเตอร์รัปต์ จากนั้น ไดร์เวอร์ค้นหาว่า มีแพ็กเก็ตเพิ่มเติมที่ต้องจัดการและจบการจัดการกับแพ็กเก็ตจำนวนมากต่ออินเตอร์รัปต์ ซึ่งเป็นอัตราแพ็กเก็ตที่เพิ่มขึ้น นั่นหมายความว่า ระบบมีประสิทธิภาพมากขึ้นเนื่องจากโหลดเพิ่มขึ้น

อย่างไรก็ตาม อะแด็ปเตอร์บางตัวมีคุณลักษณะเพิ่มเติมที่สามารถจัดเตรียมการควบคุม เมื่ออินเตอร์รัปต์ที่ได้รับถูกสร้างขึ้น สิ่งนี้เรียกว่า การรวมอินเตอร์รัปต์ หรืออินเตอร์รัปต์ตรรกะระดับกลาง ซึ่งอนุญาตให้หลายๆ แพ็กเก็ต ที่ต้องรับ และสร้างหนึ่งอินเตอร์รัปต์สำหรับหลายๆ แพ็กเก็ต ตัวจับเวลาจะเริ่มต้น เมื่อแพ็กเก็ตแรกมาถึง จากนั้น อินเตอร์รัปต์จะถูกหน่วงเวลา *n* มิลลิวินาที หรือจนกว่าจะ *m* แพ็กเก็ตจะมาถึง เมธอดจะผันแปรตามอะแด็ปเตอร์ ซึ่งเป็นคุณลักษณะของไดร์เวอร์อุปกรณ์ที่อนุญาตให้ผู้ใช้ควบคุม

ภายใต้โหลด การรวมอินเตอร์รัปต์จะเพิ่มเวลาแฝงให้กับเวลาที่แพ็กเก็ตมาถึง แพ็กเก็ตจะอยู่ในโฮสต์หน่วยความจำ แต่โฮสต์จะไม่รู้จักแพ็กเก็ต จนกระทั่งเวลาผ่านไป อย่างไรก็ตาม ภายใต้โหลดแพ็กเก็ตที่สูงกว่า ระบบจะดำเนินการได้อย่างมีประสิทธิภาพเพิ่มขึ้นโดยใช้วงรอบ CPU เพียงเล็กน้อย เนื่องจากอินเตอร์รัปต์จำนวนน้อย จะถูกสร้างขึ้น และประมวลผลโฮสต์ในหลายๆ แพ็กเก็ตต่ออินเตอร์รัปต์

สำหรับอะแด็ปเตอร์ AIX ที่สอดแทรกคุณลักษณะระดับกลางของอินเตอร์รัปต์ คุณควรตั้งค่าให้มีระดับกลาง เพื่อลดการใช้อินเตอร์รัปต์โดยไม่เพิ่มจำนวนของเวลาแฝงเพิ่มเติม สำหรับแอ็พพลิเคชันที่อาจต้องมีเวลาแฝงจำนวนน้อยที่สุด คุณควรปิดหรือเปลี่ยนอ็อพชัน เพื่ออนุญาตให้อินเตอร์รัปต์เพิ่มขึ้นต่อวินาทีสำหรับเวลาแฝงที่ลดลง

อะแด็ปเตอร์ Gigabit Ethernet นำเสนอคุณลักษณะระดับกลางของอินเตอร์รัปต์ อะแด็ปเตอร์ FC 2969 และ FC 2975 GigE PCI จะมีเมธอดของค่าหน่วงเวลา และจำนวนบัฟเฟอร์ อะแด็ปเตอร์จะเริ่มต้นตัวจับเวลา เมื่อแพ็กเก็ตแรกมาถึง และอินเตอร์รัปต์จะเกิดขึ้นเมื่อตัวจับเวลาหมดอายุ หรือเมื่อ *n* บัฟเฟอร์ในโฮสต์ที่ถูกใช้

อะแด็ปเตอร์ FC 5700, FC 5701, FC 5706 และ FC 5707 GigE PCI-X ใช้เมธอดของอัตราอินเตอร์รัปต์ที่ช้าลง ซึ่งสร้างอินเตอร์รัปต์ที่ความถี่ที่ระบุ ซึ่งอนุญาตสำหรับจัดกลุ่มของแพ็กเก็ตตามเวลา อัตราอินเตอร์รัปต์ดีฟอลต์คือ 10 000 อินเตอร์รัปต์

ต่อวินาที สำหรับการใช้อินเตอร์รัปต์ที่ต่ำ คุณสามารถตั้งค่าอัตราอินเตอร์รัปต์ให้มีค่าต่ำสุด 2 000 อินเตอร์รัปต์ต่อวินาที สำหรับเวิร์กโหลดที่เรียกเวลาแฝงต่ำและเวลาตอบสนองเร็วขึ้น คุณสามารถตั้งค่าอัตราอินเตอร์รัปต์ให้มีค่าสูงสุด 20 000 อินเตอร์รัปต์ การตั้งค่าอัตราอินเตอร์รัปต์ ให้มีค่า 0 จะปิดใช้งานการอินเตอร์รัปต์อย่างสมบูรณ์

อะแด็ปเตอร์ 10 Gigabit Ethernet PCI-X (FC 5718 and 5719) มีอ็อพชันการรวมอินเตอร์รัปต์ (*rx_int_delay*) ที่มีการหน่วงเวลา 0.82 มิลลิวินาที ความยาวแท้จริงของเวลาหน่วงจะถูกพิจารณาโดยการคูณ 0.82 ด้วยค่าที่ตั้งค่าอยู่ใน *rx_int_delay* อ็อพชันนี้จะถูกปิดใช้งาน ตามค่าดีฟอลต์ (*rx_int_delay*=0) เนื่องจากการทดสอบที่สรุปว่า อัตราอินพุตที่สูงกว่าของอะแด็ปเตอร์เหล่านี้ การรวมอินเตอร์รัปต์จะไม่ช่วยในเรื่องของผลการทำงาน.

ตารางที่ 9. คุณสมบัติของอะแด็ปเตอร์ 10 Gigabit Ethernet PCI-X

ชนิดอะแด็ปเตอร์	โค้ดคุณลักษณะ	แอ็ตทริบิวต์ ODM	ค่าดีฟอลต์	ขอบเขต
10 Gigabit Ethernet PCI-X (LR or SR)	5718, 5719	rx_int_delay	0	0-512

การปรับรีซอร์สอะแด็ปเตอร์

เนื่องจากอะแด็ปเตอร์และไดรเวอร์มีอยู่หลายชนิด จึงเป็นการยากที่จะ อธิบายอะแด็ปเตอร์แอ็ตทริบิวต์ครบทุกชนิด ข้อมูลต่อไปนี้เน้นแอ็ตทริบิวต์ทั่วไป ที่อะแด็ปเตอร์เครือข่ายและไดรเวอร์ส่วนใหญ่มี ซึ่งสามารถส่งผลกระทบ ต่อประสิทธิภาพระบบ

ไดรเวอร์การสื่อสารส่วนใหญ่มีชุดของพารามิเตอร์ที่ปรับได้ ซึ่งใช้ควบคุมรีซอร์สการส่งผ่านและการรับ โดยปกติ พารามิเตอร์เหล่านี้ควบคุมขีดจำกัด คิวการส่งผ่านและคิวการรับ แต่ยังอาจควบคุมจำนวนและขนาด ของบัฟเฟอร์หรือรีซอร์สอื่นด้วย พารามิเตอร์เหล่านี้จำกัดจำนวนของ บัฟเฟอร์หรือแพ็กเก็ตที่อาจจัดคิวสำหรับการส่งผ่าน หรือจำกัดจำนวน ของบัฟเฟอร์การรับที่มีอยู่สำหรับการรับแพ็กเก็ต พารามิเตอร์เหล่านี้สามารถ ปรับได้เพื่อให้แน่ใจถึงการจัดคิวที่เพียงพอที่ระดับอะแด็ปเตอร์ เพื่อจัดการ โหลดสูงสุดซึ่งสร้างขึ้นโดยระบบหรือเครือข่าย

ข้อมูลต่อไปนี้เป็นคำแนะนำ ทั่วไปบางอย่าง:

- เมื่อต้องการแสดงข้อมูลรายละเอียดเกี่ยวกับรีซอร์สอะแด็ปเตอร์และข้อผิดพลาด ใดๆ ที่อาจเกิดขึ้น ให้ใช้คำสั่งต่อไปนี้ โดยขึ้นอยู่กับชนิดอะแด็ปเตอร์ ที่คุณใช้:
  - **netstat -v**
  - **entstat**
  - **atmstat**
  - **fddistat**
  - **tokstat**
- มอนิเตอร์รายงานบันทึกข้อผิดพลาดระบบโดยใช้คำสั่ง **errpt** และ **errpt -a**
- โปรดจำว่าควรเปลี่ยนพารามิเตอร์เฉพาะถ้าใช้เงื่อนไขใดๆ ต่อไปนี้เท่านั้น:
  - มีหลักฐานบ่งชี้การขาดแคลนรีซอร์ส
  - มีคิวโอเวอร์รัน
  - การวิเคราะห์ประสิทธิภาพบ่งชี้ว่าต้องการการปรับระบบบางอย่าง

คิวการส่งผ่าน:

สำหรับการส่งผ่าน ไดรเวอร์อุปกรณ์สามารถให้ขีดจำกัด *คิวการส่งผ่าน*

อาจมีทั้งขีดจำกัดคิวฮาร์ดแวร์และคิวซอฟต์แวร์ขึ้นอยู่กับ ไดรเวอร์และอะแด็ปเตอร์ บางไดรเวอร์มีคิวฮาร์ดแวร์เพียงอย่างเดียว บางไดรเวอร์มี ทั้งคิวฮาร์ดแวร์และซอฟต์แวร์ บางไดรเวอร์ควบคุมคิวฮาร์ดแวร์ภายในระบบ และอนุญาตให้แก้ไขได้เฉพาะขีดจำกัดคิวซอฟต์แวร์อย่างเดียว โดยทั่วไป ไดรเวอร์อุปกรณ์จะจัดคิวแพ็กเก็ตการส่งผ่านในอะแด็ปเตอร์ฮาร์ดแวร์คิวโดยตรง ถ้า CPU ระบบเร็วเมื่อเปรียบเทียบกับความเร็วของเครือข่าย หรือบน ระบบ SMP ระบบอาจจัดทำแพ็กเก็ตการส่งผ่านได้เร็วกว่าจำนวนที่สามารถ ส่งผ่านบนเครือข่ายได้ ซึ่งจะทำให้คิวฮาร์ดแวร์ เต็ม

หลังจากคิวฮาร์ดแวร์เต็มแล้ว บางไดรเวอร์จะนำเสนอคิวซอฟต์แวร์ แล้วจะจัดคิวในคิวซอฟต์แวร์ ถ้าใช้งานถึงขีดจำกัดคิวการส่งผ่าน ซอฟต์แวร์แล้ว แพ็กเก็ตการส่งผ่านจะถูกทิ้งไป ลักษณะนี้อาจกระทบต่อ ประสิทธิภาพเนื่องจากโปรโตคอลระดับบนต้องไทม์เอาต์ และส่งผ่าน แพ็กเก็ตใหม่ อย่างไรก็ตาม ในบางเวลา อะแด็ปเตอร์ต้องทิ้งแพ็กเก็ตเนื่องจากการให้ พื้นที่ว่างมากเกินไปอาจส่งผลให้ส่งแพ็กเก็ตข้อมูลเก่า

ตารางที่ 10. ตัวอย่างของขนาดคิวการส่งผ่านอะแด็ปเตอร์ PCI

ชนิดอะแด็ปเตอร์	โค้ดคุณลักษณะ	แอ็ตทริบิวต์ ODM	ค่าดีฟอลต์	ขอบเขต
IBMอะแด็ปเตอร์ 10/100 Mbps Ethernet PCI	2968	tx_que_size	8192	16-16384
10/100 Mbps Ethernet Adapter II	4962	tx_que_sz	8192	512-16384
กิกะบิตอีเทอร์เน็ต PCI (SX หรือ TX)	2969, 2975	tx_que_size	8192	512-16384
กิกะบิตอีเทอร์เน็ต PCI (SX หรือ TX)	5700, 5701, 5706, 5707	tx_que_sz	8192	512-16384
10 กิกะบิตอีเทอร์เน็ต PCI-X (LR หรือ SR)	5718, 5719	tx_que_sz	8192	512-16384
ATM 155 (MMF หรือ UTP)	4953, 4957	sw_txq_size	2048	50-16384
ATM 622 (MMF)	2946	sw_txq_size	2048	128-32768
FDDI	2741, 2742, 2743	tx_queue_size	256	3-2048

สำหรับอะแด็ปเตอร์ที่มีขีดจำกัดคิวฮาร์ดแวร์ การเปลี่ยนค่าเหล่านี้ จะส่งผลให้มีการใช้หน่วยความจำจริงมากขึ้นบนเครื่องรับ เนื่องจากการควบคุม บล็อกและบัฟเฟอร์ที่เชื่อมโยงกับหน่วยความจำนั้น ด้วยเหตุนี้ ควรเพิ่มขีดจำกัดนี้เฉพาะเมื่อจำเป็นเท่านั้น หรือสำหรับระบบขนาดใหญ่ที่สามารถปฏิเสธการเพิ่มการใช้หน่วยความจำได้ สำหรับขีดจำกัดคิวการส่งผ่านซอฟต์แวร์ การเพิ่มขีดจำกัดเหล่านี้ไม่ได้เพิ่ม การใช้หน่วยความจำ เพียงแต่ช่วยให้สามารถจัดคิวแพ็กเก็ตซึ่งถูกจัดสรรโดยโปรโตคอลชั้นสูงกว่าแล้วเท่านั้น

คำอธิบายการส่งผ่าน:

บางไดรเวอร์อนุญาตให้คุณปรับขนาดของ transmit ring หรือ จำนวนของคำอธิบายการส่งผ่าน

คิวการส่งผ่านฮาร์ดแวร์ควบคุมจำนวนสูงสุดของบัฟเฟอร์ที่สามารถ จัดคิวสำหรับอะแด็ปเตอร์เพื่อการส่งผ่านพร้อมกัน โดยปกติ คำอธิบายหนึ่งชี้ไปยังบัฟเฟอร์เดียวเท่านั้นและข้อความจะถูกส่งใน หลายบัฟเฟอร์ ไดรเวอร์จำนวนมากไม่อนุญาตให้คุณเปลี่ยนพารามิเตอร์

ชนิดของอะแด็ปเตอร์	ฟีเจอร์โค้ด	แอ็ตทริบิวต์ ODM	ค่าดีฟอลต์	ขอบเขต
กิกะบิตอีเทอร์เน็ต PCI-X, SX หรือ TX	5700, 5701, 5706, 507	txdesc_que_sz	512	128-1024, ผลคูณของ 128

การรับรีซอร์ส:

อะแด็ปเตอร์บางตัวอนุญาตให้คุณตั้งค่าคอนฟิกจำนวนของรีซอร์สที่ใช้ สำหรับการรับแพ็กเก็ตจากเครือข่าย จำนวนนี้อาจรวมจำนวนของบัฟเฟอร์การรับ (และแม้กระทั่งขนาดของบัฟเฟอร์) หรือจำนวนของคำอธิบายการรับ DMA

บางไดรเวอร์มีหลายพูลบัฟเฟอร์การรับที่มีบัฟเฟอร์ขนาดแตกต่างกัน ซึ่งอาจต้องปรับสำหรับเวิร์กโหลดต่างๆ บางไดรเวอร์จัดการรีซอร์ส เหล่านี้ภายในไดรเวอร์และไม่อนุญาตให้คุณเปลี่ยนแปลง

อาจต้องเพิ่มรีซอร์สการรับเพื่อจัดการปริมาณงานสูงสุด บนเครือข่าย ไดรเวอร์อุปกรณ์อินเตอร์เฟสเครือข่ายวางแพ็กเก็ตขาเข้าไว้บน คิวการรับ ถ้ารายการคำอธิบายการรับหรือ ring เต็ม หรือไม่มีบัฟเฟอร์ แพ็กเก็ตจะถูกทิ้ง ส่งผลให้ผู้ส่งต้อง ส่งผ่านใหม่ คิวคำอธิบายการรับสามารถปรับได้โดยใช้เครื่องมือ SMIT หรือคำสั่ง **chdev** (โปรดดู “การเปลี่ยนพารามิเตอร์เน็ตเวิร์ก” ในหน้า 314) ขนาดคิวสูงสุดเป็นค่าเฉพาะสำหรับ อะแด็ปเตอร์การสื่อสารแต่ละชนิด และโดยปกติ สามารถดูได้โดยใช้ปุ่ม **F4** หรือ **List** ในเครื่องมือ SMIT

ตารางที่ 11. ตัวอย่างของขนาดคิวการรับอะแด็ปเตอร์ PCI

ชนิดอะแด็ปเตอร์	โค้ดคุณลักษณะ	แอ็ตทริบิวต์ ODM	ค่าดีฟอลต์	ขอบเขต
IBMอะแด็ปเตอร์ 10/100 Mbps Ethernet PCI	2968	rx_que_size rx_buf_pool_size	256 384	16, 32 ,64, 128, 26 16-2048
อะแด็ปเตอร์ 10/100 Mbps Ethernet PCI II	4962	rx_desc_que_sz rxbuf_pool_sz	512 1024	100-1024 512-2048
กิกะบิตอีเทอร์เน็ต PCI (SX หรือ TX)	2969, 2975	rx_queue_size	512	512 (คงที่)
กิกะบิตอีเทอร์เน็ต PCI-X (SX หรือ TX)	5700, 5701, 5706, 5707, 5717, 5768, 5271, 5274, 5767 และ 5281	rxbuf_pool_sz rxdesc_que_sz	2048 1024	512-16384,1 128-3840,128
10 กิกะบิต PCI-X (SR หรือ LR)	5718, 5719	rxdesc_que_sz rxbuf_pool_sz	1024 2048	128-1024, by 128 512-2048
ATM 155 (MMF หรือ UTP)	4953, 4957	rx_buf4k_min	x60	x60-x200 (96-512)
ATM 622 (MMF)	2946	rx_buf4k_min rx_buf4k_max	256 [2] 0 [1]	0-4096 0-14000
FDDI	2741, 2742, 2743	RX_buffer_cnt	42	1-512

หมายเหตุ:

1. แอ็ตทริบิวต์ **rx_buf4k_max** ของอะแด็ปเตอร์ ATM คือ จำนวนบัฟเฟอร์สูงสุดในพูลบัฟเฟอร์การรับ เมื่อมีการตั้งค่าเป็น 0 ไดรเวอร์จะกำหนดจำนวนตามจำนวนของหน่วยความจำบนระบบ (ตัวอย่างเช่น **rx_buf4k_max= thewall** * 6 / 320)

แต่มีขีดจำกัดสูงสุด 9500 บัฟเฟอร์สำหรับอะแด็ปเตอร์ ATM 155 และ 16360 บัฟเฟอร์สำหรับอะแด็ปเตอร์ ATM 622 บัฟเฟอร์จะถูกรีลีส (ลดลงถึง **rx_buf4k_min**) เมื่อไม่ต้องการ

2. แอ็ตทริบิวต์ **rx_buf4k_min** ของอะแด็ปเตอร์ ATM คือ จำนวนต่ำสุดของบัฟเฟอร์ที่ว่างในพูล ไดรเวอร์พยายามรักษาบัฟเฟอร์ที่ว่าง ในพูลให้มีจำนวนนี้เท่านั้น พูลสามารถขยายได้สูงสุดถึงค่า **rx_buf4k_max**

คำสั่งในการเคียวรี่และเปลี่ยนแอ็ตทริบิวต์อุปกรณ์:

ยูทิลิตีสถานะทั่วไปสามารถนำมาใช้เพื่อแสดงข้อจำกัดของคิวการส่งผ่าน และจำนวนของข้อผิดพลาดเกี่ยวกับ *ไม่มีรีซอร์ส* หรือ *ไม่มีบัฟเฟอร์*

คุณสามารถใช้คำสั่ง **netstat -v** หรือไปยังยูทิลิตีข้อมูลสถิติเกี่ยวกับอะแด็ปเตอร์ได้โดยตรง (**entstat** สำหรับอีเทอร์เน็ต **tokstat** สำหรับโทเค็นริง **fddistat** สำหรับ FDDI **atmstat** สำหรับ ATM และอื่นๆ)

สำหรับเอาต์พุตตัวอย่าง **entstat** โปรดดู "ข้อมูลสถิติของอะแด็ปเตอร์" ในหน้า 358 เมธอดอื่นๆ คือ การใช้ยูทิลิตี **netstat -i** ถ้าระบบแสดงจำนวนนับที่ไม่ใช่ศูนย์ในคอลัมน์ *Oerrs* สำหรับอินเตอร์เฟส นั่นคือผลลัพธ์ของเอาต์พุตคิวโอเวอร์โฟลว์

การดูค่าติดตั้งอะแด็ปเตอร์เครือข่าย:

คุณสามารถใช้คำสั่ง **lsattr -E -l** *adapter-name* หรือคุณสามารถใช้คำสั่ง SMIT (**smitty commodev**) เพื่อแสดง การตั้งค่าคอนฟิกอะแด็ปเตอร์

อะแด็ปเตอร์ที่แตกต่างกันมีชื่อที่แตกต่างกันสำหรับตัวแปรเหล่านี้ ตัวอย่างเช่น อาจมีชื่อว่า *sw_txq_size*, *tx_que_size*, หรือ *xmt_que_size* สำหรับพารามิเตอร์คิวการส่งผ่าน พารามิเตอร์ขนาดคิวการรับและบัฟเฟอร์พูลการรับอาจมีชื่อว่า *rec_que_size*, *rx_que_size*, หรือ *rv_buf4k_min* เป็นต้น

ข้อมูลต่อไปนี้ เป็นตัวอย่างจากเอาต์พุตของคำสั่ง **lsattr -E -l atm0** บนอะแด็ปเตอร์ IBM PCI 622 Mbps ATM เอาต์พุตแสดงว่า *sw_txq_size* มีการตั้งค่า เป็น 2048 และบัฟเฟอร์การรับ *rx_buf4K_min* มีการตั้งค่าเป็น 256

```
# lsattr -E -l atm0
adapter_clock  0           Provide SONET Clock                          True
alt_addr       0x0         ALTERNATE ATM MAC address (12 hex digits)    True
busintr        99          Bus Interrupt Level                          False
interface_type 0           Sonet or SDH interface                       True
intr_priority  3           Interrupt Priority                           False
max_vc         1024        Maximum Number of VCs Needed                 True
min_vc         64          Minimum Guaranteed VCs Supported             True
regmem         0xe0008000  Bus Memory address of Adapter Registers      False
rx_buf4k_max   0           Maximum 4K-byte pre-mapped receive buffers   True
rx_buf4k_min   256         Minimum 4K-byte pre-mapped receive buffers   True
rx_checksum    yes         Enable Hardware Receive Checksum             True
rx_dma_mem     0x4000000   Receive  bus memory address range            False
sw_txq_size    2048        Software Transmit Queue size                 True
tx_dma_mem     0x2000000   Transmit bus memory address range            False
uni_vers       auto_detect SVC UNI Version                              True
use_alt_addr   no          Enable ALTERNATE ATM MAC address             True
virtmem        0xe0000000  Bus Memory address of Adapter Virtual Memory False
```

ข้อมูลต่อไปนี้เป็นตัวอย่างของค่าติดตั้งของอะแด็ปเตอร์ PCI-X กิกะบิต อีเทอร์เน็ตโดยใช้คำสั่ง **lsattr -E -l ent0** เอาต์พุตนี้แสดงว่า *tx_que_size* มีการตั้งค่าเป็น 8192, *rxbuf_pool_sz* มีการตั้งค่า เป็น 2048, และ *rx_que_size* มีการตั้งค่าเป็น 1024

```
# lsattr -E -l ent0

alt_addr       0x000000000000   Alternate ethernet address                    True
busintr        163              Bus interrupt level                           False
busmem         0xc0080000       Bus memory address                            False
chksum_offload yes              Enable hardware transmit and receive checksum True
compat_mode    no               Gigabit Backward compatibility                True
copy_bytes     2048             Copy packet if this many or less bytes        True
flow_ctrl      yes              Enable Transmit and Receive Flow Control      True
intr_priority  3                Interrupt priority                            False
intr_rate      10000            Max rate of interrupts generated by adapter   True
jumbo_frames   no               Transmit jumbo frames                         True
large_send     yes              Enable hardware TX TCP resegmentation         True
media_speed    Auto_Negotiation Media speed                                   True
rom_mem        0xc0040000       ROM memory address                            False
rx_hog         1000             Max rcv buffers processed per rcv interrupt   True
rxbuf_pool_sz  2048             Rcv buffer pool, make 2X rxdesc_que_sz        True
rxdesc_que_sz  1024             Rcv descriptor queue size                     True
slih_hog       10               Max Interrupt events processed per interrupt  True
tx_que_sz      8192             Software transmit queue size                  True
txdesc_que_sz  512              TX descriptor queue size                      True
use_alt_addr   no               Enable alternate ethernet address             True
```

การเปลี่ยนพารามิเตอร์เน็ตเวิร์ก

เมื่อมีความเป็นไปได้ ให้ใช้คำสั่ง smitty เพื่อเปลี่ยนพารามิเตอร์เน็ตเวิร์ก

หากต้องการเลือกชนิดของอุปกรณ์โดยเฉพาะ ให้ใช้คำสั่ง smitty commodev จากนั้น เลือกชนิดของอะแด็ปเตอร์จากรายการที่แสดง ต่อไปนี้คือ ตัวอย่างของคำสั่ง smitty commodev เพื่อเปลี่ยนพารามิเตอร์เน็ตเวิร์ก สำหรับอะแด็ปเตอร์อีเทอร์เน็ต:

```
                        Change/Show Characteristics of an Ethernet Adapter

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                [Entry Fields]
  Ethernet Adapter                             ent2
  Description                                  10/100/1000 Base-TX PCI-X Adapter (14106902)
  Status                                       Available
  Location                                     1V-08
  Receive descriptor queue size                [1024]                                          +#
  Transmit descriptor queue size               [512]                                           +#
  Software transmit queue size                 [8192]                                          +#
  Transmit jumbo frames                        no                                              +
  Enable hardware transmit TCP resegmentation  yes                                             +
  Enable hardware transmit and receive checksum yes                                            +
  Media Speed                                  Auto_Negotiation                                +
  Enable ALTERNATE ETHERNET address            no                                              +
  ALTERNATE ETHERNET address                   [0x000000000000]                                +
  Apply change to DATABASE only                no                                              +

F1=Help                  F2=Refresh               F3=Cancel                F4=List
Esc+5=Reset              Esc+6=Command            Esc+7=Edit               Esc+8=Image
Esc+9=Shell              Esc+0=Exit               Enter=Do
```

หากต้องการเปลี่ยนค่าพารามิเตอร์ใดๆ ให้ทำดังต่อไปนี้:

1. **ดึงอินเตอร์เฟสด้วยการรันคำสั่งต่อไปนี้:**

   ```
   # ifconfig en0 detach
   ```

   โดยที่ *en0* คือชื่อของอะแด็ปเตอร์
2. **ใช้ SMIT เพื่อแสดงค่าติดตั้งอะแด็ปเตอร์ เลือก อุปกรณ์ -> การสื่อสาร -> *ชนิดของอะแด็ปเตอร์* -> เปลี่ยน/แสดง...**

3. เลื่อนเคอร์เซอร์ไปยังฟิลด์ที่คุณต้องการเปลี่ยน และกด **F4** เพื่อดูช่วงต่ำสุดและสูงสุดสำหรับฟิลด์ (หรือชุดของขนาดที่ระบุเฉพาะ ซึ่งได้รับการสนับสนุน)
4. เลือกขนาดที่เหมาะสม และกด Enter เพื่ออัพเดตฐานข้อมูล ODM
5. พ่วงต่ออะแด็ปเตอร์อีกครั้งโดยรันคำสั่งต่อไปนี้:

   ```
   # ifconfig en0 hosthame up
   ```

เมธอดสำรองในการเปลี่ยนค่าพารามิเตอร์เหล่านี้คือ การรันคำสั่งต่อไปนี้:

```
# chdev -l [ifname] -a [attribute-name]=newvalue
```

ตัวอย่างเช่น หากต้องการเปลี่ยน **tx_que_size** ข้างต้นบน en0 ให้เป็น 128 ให้ใช้ลำดับของคำสั่งดังนี้ หมายเหตุ ไดร์เวอร์นี้จะสนับสนุนเฉพาะขนาดที่แตกต่างกันสี่ขนาดเท่านั้น ดังนั้น ให้ใช้คำสั่ง SMIT เพื่อดูค่าเหล่านั้นจะเป็นการดีกว่า

```
# ifconfig en0 detach
# chdev -l ent0 -a tx_que_size=128
# ifconfig en0 hostname up
```

การปรับขนาดเซกเมนต์สูงสุดของ TCP

แพ็กเก็ตขนาดสูงสุดที่ TCP ส่งอาจส่งผลกระทบอย่างมากต่อแบนด์วิธ เนื่องจากการส่งขนาดแพ็กเก็ตที่ใหญ่ที่สุดเท่าที่เป็นไปได้บนเครือข่าย มีประสิทธิภาพมากกว่า

TCP ควบคุมขนาดสูงสุดนี้ ซึ่งเรียกกันว่า Maximum Segment Size (MSS) สำหรับแต่ละการเชื่อมต่อ TCP สำหรับเครือข่ายแบบติดตั้งโดยตรง TCP คำนวณ MSS โดยใช้ขนาด MTU ของอินเตอร์เฟสเครือข่าย แล้วลบด้วยส่วนหัวโปรโตคอล เพื่อให้ได้ค่าขนาดของข้อมูลในแพ็กเก็ต TCP ตัวอย่างเช่น อีเทอร์เน็ตที่มี MTU ของ 1500 จะได้ผลลัพธ์เป็น MSS ของ 1460 หลังจากลบ 20 ไบต์ ของส่วนหัว IPv4 และ 20 ไบต์ของส่วนหัว TCP

โปรโตคอล TCP มีกลไกสำหรับปลายทั้งสองด้านของการเชื่อมต่อ เพื่อเผยแพร่ MSS ที่จะใช้บนการเชื่อมต่อเมื่อสร้างการเชื่อมต่อขึ้น ผู้ใช้ขั้นปลายแต่ละรายใช้ฟิลด์ OPTIONS ในส่วนหัว TCP เพื่อเผยแพร่ MSS ที่นำเสนอ MSS ที่เลือกคือค่าที่น้อยกว่าที่ให้โดยทั้งสอง ด้าน ถ้าปลายด้านหนึ่งไม่ได้ให้ MSS ระบบจะใช้ 536 ไบต์ ซึ่งไม่ดีสำหรับประสิทธิภาพ

ปัญหาคือแต่ละ TCP endpoint ทราบเฉพาะ MTU ของเครือข่าย ที่เชื่อมต่ออยู่เท่านั้น และไม่ทราบขนาด MTU ของเครือข่ายอื่นที่อาจมีอยู่ระหว่าง ปลายทั้งสองด้าน ดังนั้น TCP จะทราบ MSS ที่ถูกต้องถ้า ปลายทั้งสองด้านอยู่บนเครือข่ายเดียวกันเท่านั้น ด้วยเหตุนี้ TCP จึงจัดการเผยแพร่ MSS แตกต่างกันขึ้นอยู่กับการตั้งค่าคอนฟิกเครือข่าย ถ้าต้องการหลีกเลี่ยง การส่งแพ็กเก็ตที่อาจต้องการการแบ่งเฟรกเมนต์ IP บนเครือข่าย MTU ที่เล็กกว่า

ค่าของ MSS ที่เผยแพร่โดยซอฟต์แวร์ TCP ในระหว่างการตั้งค่าการเชื่อมต่อ ขึ้นอยู่กับว่าปลายอีกด้านหนึ่งเป็นระบบโลคัลบนเครือข่ายฟิสิคัลเดียวกัน (นั่นคือระบบมีหมายเลขเครือข่ายเหมือนกัน) หรืออยู่บนเครือข่าย (รีโมต) ที่แตกต่างอื่น

โฮสต์บนเน็ตเวิร์กเดียวกัน:

ถ้าสิ่งอื่นใดที่จบการเชื่อมต่ออยู่บน IP network เดียวกัน MSS ที่ลงประกาศโดย TCP จะอิงตาม MTU ของโลคัลเน็ตเวิร์กอินเตอร์เฟส

```
TCP MSS = MTU - TCP header size - IP header size
```

ขนาดของ TCP คือ 20 ไบต์ ขนาดส่วนหัวของ IPv4 คือ 20 ไบต์ และขนาดส่วนหัวของ IPv6 คือ 40 ไบต์

เนื่องจากนี่คือ MSS ที่มีขนาดใหญ่ที่สุดที่เป็นไปได้ ซึ่งสามารถจัดให้มีอยู่ได้โดยไม่ต้องแตกแฟรกเมนต์ IP และค่านี้จะเป็นค่าสูงสุด ดังนั้นจึงไม่มีการปรับ MSS ที่ต้องการสำหรับเน็ตเวิร์กแบบโลคัล

โฮสต์บนเน็ตเวิร์กที่ต่างกัน:

เมื่อส่วนปลายอื่นๆ ของการเชื่อมต่ออยู่บนเน็ตเวิร์กแบบรีโมต TCP ของระบบปฏิบัติการจะดีฟอลต์ไปที่ MSS ซึ่งจะพิจารณาพร้อมกับเมธอด ด้านล่าง

เมธอดจะผันแปรหากเปิดใช้งานการค้นพบ TCP Path MTU หรือไม่ ถ้าไม่ได้เปิดใช้งานการค้นพบ Path MTU โดยที่ **tcp_pmtu_discover=0** TCP จะพิจารณาถึง MSS ที่จะใช้ในลำดับต่อไปนี้:

1. ถ้าคำสั่ง **route add** ให้ระบุขนาดของ MTU สำหรับเราต์นี้ MSS จะคำนวณจากขนาด MTU นี้
2. ถ้าพารามิเตอร์ **tcp_mssdflt** สำหรับ ISNO ได้กำหนดไว้ สำหรับให้เน็ตเวิร์กอินเตอร์เฟสใช้ ค่า **tcp_mssdflt** จะถูกใช้สำหรับ MSS
3. ถ้าไม่ได้กำหนดลำดับทั้งสองดังที่ได้กล่าวไว้ข้างต้น TCP จะใช้ค่าที่สามารถปรับแต่งได้ **no tcp_mssdflt** แบบโกลบอล ค่าดีฟอลต์สำหรับอ็อพชันนี้คือ 1460 ไบต์

การค้นหา MTU ของ TCP พาธ:

อ็อพชันโปรโตคอลการค้นหา MTU ของ TCP พาธมีการเปิดใช้งานโดยค่าดีฟอลต์ ใน AIX อ็อพชันนี้ช่วยให้โปรโตคอลสแต็คทราบขนาด MTU ต่ำสุดบนเครือข่ายใดๆ ที่อยู่ในพาธระหว่างสองโฮสต์ ในปัจจุบัน และมีการควบคุมโดยอ็อพชันเครือข่าย **tcp_pmtu_discover=1**

การใช้การค้นหา TCP Path MTU จะใช้แพ็กเก็ต TCP ของ การเชื่อมต่อเองแทนข้อความ ICMP ECHO ส่วนขยายเคอร์นัล TCP/IP เก็บรักษาตาราง ที่เรียกว่าตาราง PMTU เพื่อจัดเก็บข้อมูลการค้นหา PMTU ที่เกี่ยวข้อง รายการสำหรับปลายทางแต่ละแห่งมีการสร้างขึ้นในตาราง PMTU เมื่อสร้างการ เชื่อมต่อ TCP ไปยังปลายทางนั้น ค่า PMTU คือค่าอินเตอร์เฟส MTU ขาออก

แพ็กเก็ต TCP มีการส่งโดยมีชุดบิต Don't Fragment, หรือ DF, ในส่วนหัว IP แพ็กเก็ต TCP เข้าถึงเราเตอร์เครือข่ายที่มีค่า MTU ที่น้อยกว่า ขนาดของแพ็กเก็ต TCP เราเตอร์ส่งกลับข้อความแสดงข้อผิดพลาด ICMP ซึ่งบ่งชี้ว่าข้อความไม่สามารถส่งต่อได้เนื่องจากไม่สามารถแบ่ง เฟรกเมนต์ได้ ถ้าเราเตอร์ที่ส่งข้อความแสดงข้อผิดพลาดคอมไพล์ด้วย RFC 1191 ค่า MTU ของเครือข่ายจะมีอยู่ในข้อความแสดงข้อผิดพลาด ICMP มิฉะนั้น สำหรับแพ็กเก็ต TCP ที่จะส่งผ่านใหม่ ค่าที่น้อยกว่าของขนาด MTU ต้องถูกกำหนด จากตารางค่า MTU ที่รู้จักกันดีภายในส่วนขยายเคอร์นัล TCP/IP AIX จากนั้น อัพเดตค่า PMTU ของปลายทางในตาราง PMTU ด้วยขนาด MTU ที่เล็กกว่าและส่งผ่านแพ็กเก็ต TCP ใหม่ การเชื่อมต่อ TCP ในลำดับต่อมาใดๆ ไปยังปลายทางนั้นใช้ค่า PMTU ที่อัพเดต

คุณสามารถใช้คำสั่ง **pmtu** เพื่อดูหรือลบรายการ PMTU ข้อมูลต่อไปนี้เป็นตัวอย่างของคำสั่ง **pmtu**:

```
# pmtu display

   dst           gw            If    pmtu     refcnt   redisc_t    exp

 --------------------------------------------------------------------------


10.10.1.3    10.10.1.5        en1    1500       2          9       0

10.10.2.5    10.10.2.33       en0    1500       1          0       0
```

รายการ PMTU ที่ไม่ได้ใช้ซึ่งเป็นรายการ refcnt ที่มีค่าเป็น 0 จะถูกลบออกเพื่อป้องกันไม่ให้ตาราง PMTU ใหญ่เกินไป รายการที่ไม่ได้ใช้ จะถูกลบออกหลังผ่านไป *pmtu_expire* นาทีตั้งแต่ค่า refcnt เท่ากับ 0 อ็อพชันเครือข่าย *pmtu_expire* มีค่าดีฟอลต์เป็น 10 นาที เพื่อป้องกันไม่ให้รายการ PMTU หมดอายุ คุณสามารถตั้งค่า *pmtu_expire* เป็น 0

Route cloning ไม่จำเป็นสำหรับการใช้การค้นหา MTU ของ TCP พาธ ซึ่งหมายความว่าตารางการเราต์จะเล็กลงและจัดการได้มากขึ้น

สแตติกเราต์:

คุณสามารถแทนที่ค่า MSS ดีฟอลต์จำนวน 1460 ไบต์โดยการระบุ สแตติกเราต์ไปยังเครือข่ายรีโมตเฉพาะ

ใช้อ็อพชัน **–mtu** ของคำสั่ง **route** เพื่อระบุ MTU ให้กับเครือข่ายนั้น ในกรณีนี้ คุณอาจระบุ MTU ต่ำสุดจริงของเราต์ แทนการคำนวณค่า MSS ตัวอย่างเช่น คำสั่งต่อไปนี้ จะตั้งค่าขนาด MTU ดีฟอลต์เป็น 1500 สำหรับเราต์ไปยังเครือข่าย 192.3.3 และดีฟอลต์โฮสต์สำหรับการรับ gateway นั้นคือ en0host2:

```
# route add -net 192.1.0 jack -mtu 1500
1500 net 192.3.3: gateway en0host2
```

คำสั่ง **netstat –r** แสดงตารางเราต์ และแสดงว่าขนาด PMTU คือ 1500 ไบต์ TCP คำนวณ MSS จากขนาด MTU นั้น ข้อมูลต่อไปนี้เป็นตัวอย่างของคำสั่ง **netstat –r**:

```
# netstat -r
Routing tables
Destination      Gateway           Flags   Refs     Use  If   PMTU Exp Groups

Route tree for Protocol Family 2 (Internet):
default          res101141         UGc       0        0  en4     -   -
ausdns01.srv.ibm res101141         UGHW      8       40  en4  1500   -
10.1.14.0        server1           UHSb      0        0  en4     -   -  =>
10.1.14/24       server1           U         5     4043  en4     -   -
server1          loopback          UGHS      0      125  lo0     -   -
10.1.14.255      server1           UHSb      0        0  en4     -   -
127/8            loopback          U         2  1451769  lo0     -   -
192.1.0.0        en0host1          UHSb      0        0  en0     -   -  =>
192.1.0/24       en0host1          U         4       13  en0     -   -
en0host1         loopback          UGHS      0        2  lo0     -   -
192.1.0.255      en0host1          UHSb      0        0  en0     -   -
192.1.1/24       en0host2          UGc       0        0  en0     -   -
en1host1         en0host2          UGHW      1   143474  en0  1500   -
192.3.3/24       en0host2          UGc       0        0  en0  1500   -
192.6.0/24       en0host2          UGc       0        0  en0     -   -

Route tree for Protocol Family 24 (Internet v6):
loopbackv6       loopbackv6        UH        0        0  lo0 16896   -
```

หมายเหตุ: คำสั่ง **netstat –r** ไม่ได้แสดงค่า PMTU คุณสามารถดูค่า PMTU โดยใช้คำสั่ง **pmtu display** เมื่อคุณเพิ่มเราต์สำหรับปลายทางโดยใช้คำสั่ง **route add** และคุณระบุค่า MTU จะมีการสร้างรายการ PMTU ขึ้น ในตาราง PMTU ของปลายทางนั้น

ในสภาพแวดล้อมคงที่ขนาดเล็ก วิธีการนี้ให้การควบคุม MSS ที่แม่นยำแบบทีละเครือข่าย ข้อเสียของวิธีการนี้มีดังนี้:

- ไม่ทำงานกับการเราต์แบบไดนามิก

- ไม่สามารถใช้งานได้จริงเมื่อจำนวนเครือข่ายรีโมตเพิ่มขึ้น
- ์สแตติกเราต์ต้องมีการตั้งค่าที่ปลายทั้งสองด้าน เพื่อให้มั่นใจว่าปลายทั้งสองด้าน เห็นชอบกับ MSS ที่ใหญ่กว่าค่าดีฟอลต์

การใช้อ็อพชัน tcp_mssdflt ของคำสั่ง no:

อ็อพชัน *tcp_mssdflt* ใช้เพื่อตั้งค่าขนาดแพ็กเก็ต สูงสุดสำหรับการสื่อสารกับเครือข่ายรีโมต

อ็อพชัน *tcp_mssdflt* สากลของคำสั่ง **no** ใช้กับทุกเครือข่าย อย่างไรก็ตาม สำหรับอินเตอร์เฟสเครือข่ายที่สนับสนุนอ็อพชัน ISNO คุณสามารถตั้งค่าอ็อพชัน *tcp_mssdflt* บนแต่ละด้าน ของอินเตอร์เฟสเหล่านั้น ค่านี้แทนที่ค่าคำสั่ง **no** สากล สำหรับเราต์การใช้เครือข่าย

อ็อพชัน *tcp_mssdflt* คือขนาด TCP MSS ซึ่งแสดงถึงขนาดข้อมูล TCP เมื่อต้องการคำนวณขนาด MSS นี้ให้ใช้ขนาด MTU เครือข่ายที่ต้องการและลบออก 40 ไบต์ (20 สำหรับ IP และ 20 สำหรับส่วนหัว TCP) ไม่จำเป็นต้องปรับโปรโตคอลอ็อพชันอื่นเนื่องจาก TCP จัดการกับการปรับนี้ถ้ามีการใช้อ็อพชันอื่น เช่น อ็อพชัน *rfc1323*

ในสภาพแวดล้อมที่มี MTU ใหญ่กว่าค่าดีฟอลต์ วิธีการนี้มีประโยชน์ในแง่ที่ว่าไม่จำเป็นต้องตั้งค่า MSS สำหรับแต่ละ เครือข่าย ข้อเสียมีดังนี้:

- การเพิ่มค่าดีฟอลต์อาจทำให้เกิดการแบ่งเฟรกเมนต์ IP router ถ้าปลายทาง อยู่บนเครือข่ายที่เป็นแบบรีโมตอย่างแท้จริง และไม่ทราบ MTUs ของเครือข่าย ที่แทรกอยู่ระหว่างกลาง
- อ็อพชัน *tcp_mssdflt* ต้องมีการตั้งค่าให้เหมือนกับค่าบน โฮสต์ปลายทาง

หมายเหตุ: เริ่มต้นด้วย AIX คุณสามารถใช้อ็อพชัน *tcp_mssdflt* ได้เฉพาะถ้าอ็อพชัน *tcp_pmtu_discover* มีการตั้งค่าเป็น 0 เท่านั้น

Subnetting และอ็อพชัน subnetsarelocal ของคำสั่ง no:

คุณสามารถใช้อ็อพชัน *subnetsarelocal* ของ คำสั่ง **no** เพื่อควบคุมว่า TCP จะพิจารณา remote endpoint ว่าเป็นโลคัล (บนเครือข่ายเดียวกัน) หรือรีโมตเมื่อไร

สามารถทำให้หลายเครือข่ายฟิสิคัลแบ่งใช้หมายเลขเครือข่ายเดียวกันได้ โดยการ subnetting อ็อพชัน *subnetsarelocal* ระบุว่าสำหรับทั่วทั้งระบบ จะพิจารณาว่า subnets เป็นเครือข่ายแบบโลคัล หรือรีโมต ด้วยคำสั่ง **no -o subnetsarelocal=1** ซึ่งเป็นค่าดีฟอลต์ โฮสต์ A บน subnet 1 พิจารณาว่าโฮสต์ B บน subnet 2 อยู่บนเครือข่ายฟิสิคัลเดียวกัน

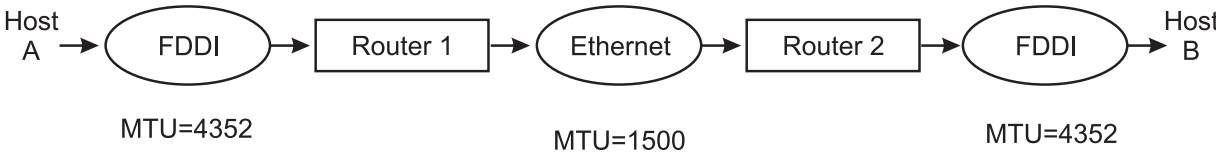
ลำดับคือเมื่อโฮสต์ A และโฮสต์ B สร้างการเชื่อมต่อ ทั้งสองโฮสต์จะตกลงเกี่ยวกับ MSS โดยสมมุติว่าอยู่บนเครือข่ายเดียวกัน แต่ละ โฮสต์เผยแพร่ MSS โดยใช้ MTU ของอินเตอร์เฟสเครือข่ายของตน โดยปกติ เลือก MSS ที่ดีที่สุด

ข้อดีของวิธีการนี้มีดังนี้:

- ไม่ต้องยึดสแตติกใดๆ เนื่องจาก MSS มีการตกลง โดยอัตโนมัติ
- ไม่ได้ปิดใช้งานหรือแทนที่การตกลงระหว่าง TCP MSS ดังนั้นความแตกต่าง เล็กน้อยใน MTU ระหว่าง subnets ที่ติดกันจึงสามารถจัดการได้ อย่างเหมาะสม

ข้อเสียของวิธีการนี้มีดังนี้:

- การแบ่งเฟรกเมนต์ IP router ที่อาจเกิดขึ้นได้เมื่อเครือข่าย MTU สูงสอง เครือข่ายมีการลิงก์ผ่านทางเครือข่าย MTU ต่ำ รูปภาพต่อไปนี้แสดง ปัญหานี้



รูปที่ 21. การแบ่งเฟรกเมนต์ระหว่าง Subnet. ภาพสาธิตนี้ แสดงพาธข้อมูลจากโฮสต์ A ผ่านทาง FDDI ที่มี MTU=4352, ผ่านทางเราเตอร์ 1 ไปยังอีเทอร์เน็ตที่มี MTU=1500 จากจุดนั้น ไปยังเราต์เตอร์ 2 และ FDDI อีกตัวหนึ่งที่มี MTU=4352 และออกไปยัง โฮสต์ B คำอธิบายเกี่ยวกับการเกิดการแบ่งเฟรกเมนต์ในตัวอย่างนี้มีอยู่ใน ข้อความต่อจากภาพสาธิต

- ในสถานการณ์จำลองนี้ โฮสต์ A และ B จะสร้างการเชื่อมต่อตามข้อมูล MTU ทั่วไปคือ 4352 แพ็กเก็ตที่เดินทางจาก A ไป B จะถูกแบ่งเฟรกเมนต์โดย เราเตอร์ 1 และยกเลิกการแบ่งเฟรกเมนต์โดยเราเตอร์ 2 การดำเนินการย้อนกลับ เมื่อเดินทางจาก B ไป A
- ทั้งต้นทางและปลายทางต้องพิจารณาว่า subnets เป็นแบบโลคัล

หมายเหตุ: ถ้าค่า *tcp_pmtu_discover* เป็น 1 ค่า MSS จะมีการคำนวณตามข้อมูลอินเตอร์เฟส MTU ขาออก ค่า *subnetsarelocal* จะ มีการพิจารณาเฉพาะถ้าค่าอ็อพชันเครือข่าย *tcp_pmtu_discover* เป็น 0 เท่านั้น

ข้อแนะนำในการปรับผลการทำงานของ IP protocol

ส่วนนี้จะจัดเตรียมข้อแนะนำสำหรับการออปติไมซ์ผลการทำงานของ IP protocol

ที่เลเยอร์ IP เฉพาะพารามิเตอร์ที่สามารถปรับได้คือ *ipqmaxlen* ซึ่งจะควบคุมความยาวของคิวอินพุต IP โดยทั่วไป อินเตอร์เฟสจะไม่ทำการสร้างคิว แพ็กเก็ตสามารถมาถึงได้อย่างรวดเร็ว และรันอยู่บนคิวอินพุต IP คุณสามารถใช้คำสั่ง **netstat -s** or **netstat -p ip** เพื่อดูตัวนับโอเวอร์โฟลว์ (`ipintrq overflows`)

ถ้าจำนวนที่ส่งคืนมีขนาดมากกว่า 0 โอเวอร์โฟลว์จะเกิดขึ้น ใช้คำสั่ง **no** เพื่อตั้งค่าความยาวสูงสุดของคิวนี้ ตัวอย่างเช่น:

```
# no -o ipqmaxlen=100
```

ตัวอย่างนี้อนุญาตให้ 100 แพ็กเก็ตอยู่ในคิว ค่าที่ต้องการใช้ที่ถูกต้อง จะถูกกำหนดโดยอัตราสูงสุดของการส่งเป็นชุดอย่างเร็วที่ได้รับ ถ้าค่านี้ไม่ได้ถูกกำหนดไว้ การใช้จำนวนของโอเวอร์โฟลว์จะสามารถช่วยพิจารณาที่ที่ควรจะเพิ่มขึ้นได้ ไม่มีหน่วยความจำเพิ่มเติมที่ถูกใช้โดยการเพิ่มความยาวของคิว อย่างไรก็ตาม การเพิ่มขึ้นอาจส่งผลทำให้ใช้เวลาเพิ่มขึ้นใน interrupt handler เนื่องจาก IP จะมีแพ็กเก็ตมากขึ้นในการประมวลผลบนคิวอินพุต ซึ่งสามารถส่งผลในทางตรงกันข้ามถึงการประมวลผล ที่ต้องการเวลา CPU การแลกเปลี่ยนจะลดการปล่อยแพ็กเก็ต เมื่อเปรียบเทียบกับสภาพพร้อมใช้งานของ CPU สำหรับการประมวลผลอื่นๆ ซึ่งเป็นสิ่งที่ดีในการเพิ่ม *ipqmaxlen* ด้วยการเพิ่มขึ้นในระดับกลาง ถ้าการแลกเปลี่ยนเกี่ยวข้องกับสภาวะแวดล้อมของคุณ

การปรับประสิทธิภาพพูล mbuf

ระบบย่อยเครือข่ายใช้ฟังก์ชันการจัดการหน่วยความจำที่เกี่ยวข้องกับ โครงสร้างข้อมูลที่เรียกว่า mbuf

Mbufs โดยส่วนใหญ่แล้ว ใช้เพื่อจัดเก็บข้อมูลในเคอร์นัลสำหรับการจราจร เครือข่ายขาเข้าและขาออก การมีพูล mbuf ที่มีขนาดถูกต้องช่วยพัฒนาประสิทธิภาพเครือข่าย ได้ ถ้าพูล mbuf มีการตั้งค่าคอนฟิกไม่ถูกต้อง ทั้งประสิทธิภาพของ เครือข่ายและระบบอาจได้รับผลกระทบ ขีดจำกัดสูงสุดของขนาดพูล mbuf ซึ่งเป็น *thewall* ที่ปรับได้ มีการกำหนดโดยอัตโนมัติโดยระบบปฏิบัติการ ขึ้นอยู่กับจำนวนของหน่วยความจำ ในระบบ ในฐานะผู้ดูแลระบบ เฉพาะคุณเท่านั้นที่สามารถปรับขีดจำกัดสูงสุด ของขนาดพูล mbuf ได้

thewall ที่ปรับได้

อ็อพชันเครือข่าย *thewall* ที่ปรับได้ตั้งค่าขีดจำกัดสูงสุด สำหรับเคอร์นัลบัฟเฟอร์เครือข่าย

ระบบตั้งค่าของ *thewall* ที่ปรับได้เป็นค่าสูงสุดโดยอัตโนมัติ และโดยทั่วไปแล้ว คุณไม่ควรเปลี่ยนค่านี้ คุณสามารถ ลดค่านี้ได้ ซึ่งจะลดจำนวนหน่วยความจำที่ระบบใช้สำหรับบัฟเฟอร์เครือข่าย แต่อาจส่งผลกระทบต่อประสิทธิภาพเครือข่าย เนื่องจาก ระบบใช้บัฟเฟอร์ ในจำนวนที่จำเป็นเท่านั้นในเวลาที่กำหนดใดๆ ถ้าระบบย่อยเครือข่ายไม่ได้มีการใช้ งานอย่างหนัก จำนวนบัฟเฟอร์ทั้งหมดควรจะต่ำกว่าค่า *thewall* อย่างมาก

หน่วยของ *thewall* ที่ปรับได้คือ 1 KB ดังนั้น 1048576 ไบต์หมายถึง 1024 MB หรือ 1 GB ของ RAM

ข้อจำกัดเกี่ยวกับรีซอร์ส mbuf

AIX 6.1 มีพื้นที่บัฟเฟอร์ mbuf สูงสุด 65 GB ซึ่งประกอบด้วยเซ็กเมนต์หน่วยความจำ 260 เซ็กเมนต์ แต่ละเซ็กเมนต์มีขนาด 256 MB

ขนาดของ **thewall** ที่สามารถปรับแต่งได้คือ 65 GB หรือครึ่งหนึ่งของจำนวนหน่วยความจำระบบ ซึ่งมีค่าที่เล็กกว่า

maxmbuf ที่สามารถปรับแต่งได้

ค่าของ *maxmbuf* ที่สามารถปรับแต่งได้จะจำกัดจำนวนหน่วยความจำที่ใช้โดยระบบย่อยการสื่อสาร

คุณยังสามารถใช้ *maxmbuf* ที่สามารถปรับแต่งได้ กับข้อจำกัด *thewall* ที่ต่ำกว่า คุณสามารถดูค่า *maxmbuf* ที่สามารถปรับแต่ง ได้ด้วยการรันคำสั่ง **lsattr -E -l sys0** ถ้าค่า *maxmbuf* มีค่ามากกว่า 0 ค่า *maxmbuf* จะถูกนำมาใช้โดยไม่พิจารณาถึงค่า *thewall* ที่สามารถปรับแต่งได้

ค่าดีฟอลต์สำหรับ *maxmbuf* ที่สามารถปรับแต่งได้คือ 0 ค่า 0 สำหรับ *maxmbuf* ที่สามารถปรับแต่งได้ จะบ่งชี้ว่า *thewall* ที่ สามารถปรับแต่งได้ถูกนำมาใช้แล้ว คุณสามารถเปลี่ยนค่า *maxmbuf* ที่สามารถปรับแต่งได้โดยใช้คำสั่ง **chdev** หรือ **smitty**

Sockthresh และ strthresh threshold ที่ปรับได้

sockthresh และ *strthresh* ที่ปรับได้ คือ thresholds สูงสุดที่จำกัดการเปิดซ็อกเก็ตใหม่หรือการเชื่อมต่อ TCP หรือการสร้างรี ซอร์สสตรีมใหม่ ซึ่งช่วยให้รีซอร์สบัฟเฟอร์มีอยู่เสมอ และทำให้มั่นใจว่าเซสชันหรือการเชื่อมต่อที่มีอยู่มีรีซอร์สในการดำเนิน งาน อย่างต่อเนื่อง

sockthresh ที่ปรับได้ระบุขีดจำกัดการใช้ หน่วยความจำ ไม่อนุญาตให้ใช้การเชื่อมต่อซ็อกเก็ตใหม่เกินกว่าค่าของ *sockthresh* ที่ปรับได้ ค่าดีฟอลต์ของ *sockthresh* ที่ปรับได้คือ 85% และหลังจากจำนวนรวมของหน่วยความจำที่จัดสรรมากถึง 85% ของ ค่า *thewall* หรือ *maxmbuf* ที่ปรับได้แล้ว คุณไม่สามารถทำการเชื่อมต่อซ็อกเก็ตใหม่ใดๆ ได้ ซึ่งหมายความว่าค่าที่ส่งคืน ของ การเรียกระบบ **socket( )** และ **socketpair( )** คือ *ENOBUFS* จนกระทั่งการใช้บัฟเฟอร์ลดลงต่ำกว่า 85%

ในลักษณะคล้ายกัน *strthresh* ที่ปรับได้จำกัดจำนวนของ หน่วยความจำ mbuf ที่ใช้สำหรับรีซอร์สสตรีม และค่าดีฟอลต์ของ *strthresh* ที่ปรับได้ คือ 85% ระบบย่อย async และ TTY รันในสภาพแวดล้อมสตรีม *strthresh* ที่ปรับได้ ระบุว่าหลังจากจำนวน รวมของหน่วยความจำที่จัดสรรมากถึง 85% ของค่า *thewall* ที่ปรับได้แล้ว จะไม่มีหน่วยความจำเพิ่มเติมไปยังรีซอร์สสตรีม ซึ่งหมายความว่าค่าที่ส่งคืนของ การเรียกสตรีมคือ *ENOSR* เพื่อเปิดสตรีม ผลักดันโมดูล หรือบันทึกใน อุปกรณ์สตรีม

คุณสามารถปรับ *sockthresh* และ *strthresh* thresholds ด้วยคำสั่ง **no**

ฟังก์ชันการจัดการ mbuf

ฟังก์ชันการจัดการ mbuf ควบคุมขนาดบัฟเฟอร์อื่นที่อยู่ในช่วง ตั้งแต่ 32 ไบต์ไปจนถึง 16384 ไบต์

พูลมีการสร้างขึ้นจากหน่วยความจำระบบโดยการร้องขอการจัดสรรไปยัง Virtual Memory Manager (VMM) พูลประกอบด้วยชิ้นส่วนที่ pinned ของ หน่วยความจำเสมือนเคอร์นัลซึ่งอยู่ในหน่วยความจำฟิสิคัลเสมอ และไม่เคย paged out ผลคือหน่วยความจำจริงที่มีอยู่สำหรับการเพจใน แอ็พพลิเคชันโปรแกรมและข้อมูลลดลงตามจำนวนที่พูล mbuf เพิ่มขึ้น

พูลหน่วยความจำเครือข่ายมีการแบ่งอย่างเท่าเทียมกันระหว่างแต่ละตัวประมวลผล จากนั้น แต่ละพูลย่อย จะถูกแบ่งออกเป็น buckets ซึ่งแต่ละ bucket มีบัฟเฟอร์ในขนาดตั้งแต่ 32 ถึง 16384 ไบต์ แต่ละ bucket สามารถยืมหน่วยความจำจาก buckets อื่นบนตัวประมวลผลเดียวกัน แต่ตัวประมวลผลไม่สามารถยืมหน่วยความจำจากพูล หน่วยความจำเครือข่ายของตัวประมวลผลอื่น เมื่อการบริการเครือข่ายต้องขนส่งข้อมูล การบริการสามารถเรียก ใช้การบริการเคอร์นัล เช่น **m_get( )** เพื่อให้ได้รับบัฟเฟอร์หน่วยความจำ ถ้าบัฟเฟอร์ มีอยู่แล้วและ pinned การบริการนั้นสามารถได้รับบัฟเฟอร์ในทันที ถ้ายัง ไม่ถึงขีดจำกัดสูงสุดและบัฟเฟอร์ไม่ได้ pinned ผลคือ บัฟเฟอร์จะถูกจัดสรรและ pinned หลังจาก pinned แล้ว หน่วยความจำยังคง pinned อยู่แต่สามารถ ปล่อยกลับไปยังพูลเครือข่ายได้ ถ้าจำนวนของบัฟเฟอร์ที่ว่างขึ้นถึงเครื่องหมาย high-water บางจำนวนจะถูก unpinned และส่งกลับไปยังระบบ สำหรับการใช้งานทั่วไป Unpinning นี้ทำโดยกระบวนการเคอร์นัล **netm( )** ผู้เรียกรูทีนย่อย **m_get( )** สามารถระบุว่าจะรอบัฟเฟอร์หน่วยความจำ เครือข่ายหรือไม่ ถ้ามีการระบุแฟล็ก **M_DONTWAIT** และไม่มีบัฟเฟอร์ที่ pinned อยู่ในเวลานั้น จำนวนความล้มเหลวจะเพิ่มขึ้น ถ้ามีการระบุแฟล็ก **M_WAIT** กระบวนการจะถูกพักไว้จนกว่า สามารถจัดสรรและ pinned บัฟเฟอร์ได้

คำสั่ง netstat -m เพื่อมอนิเตอร์ mbuf พูล

ใช้คำสั่ง **netstat -m** เพื่อตรวจสอบ การขาดแคลนหรือความล้มเหลวของการร้องขอหน่วยความจำเครือข่าย (mbufs/คลัสเตอร์)

คุณสามารถใช้คำสั่ง **netstat -Zm** เพื่อล้างข้อมูลสถิติ mbuf (หรือศูนย์) ซึ่งมีประโยชน์เมื่อกำลังรันการทดสอบให้เริ่มต้น ด้วยชุดสถิติที่ว่าง คำสั่ง **netstat -m** แสดงฟิลด์ดังต่อไปนี้:

ชื่อฟิลด์ คำนิยาม

ตามขนาด
แสดงขนาดของบัฟเฟอร์

ใช้อยู่ แสดงจำนวนบัฟเฟอร์ของขนาดเฉพาะนั้นที่ใช้อยู่

การเรียก
แสดงจำนวนการเรียก หรือการร้องขอการจัดสรรสำหรับบัฟเฟอร์แต่ละขนาด

ล้มเหลว
แสดงจำนวนของการร้องขอการจัดสรรที่ล้มเหลวเนื่องจากไม่มีบัฟเฟอร์อยู่

ล่าช้า แสดงจำนวนการเรียกที่ล่าช้า ถ้าบัฟเฟอร์ขนาดนั้นว่างและมีการตั้งค่าแฟล็ก **M_WAIT** โดยผู้เรียก

ว่าง แสดงจำนวนของบัฟเฟอร์แต่ละขนาดที่อยู่ในรายการว่าง ซึ่งพร้อมที่ จะจัดสรรได้

hiwat แสดงจำนวนบัฟเฟอร์สูงสุดที่กำหนดโดยระบบ ซึ่งสามารถคงอยู่ใน รายการว่างได้ บัฟเฟอร์ว่างใดๆ ที่เกินกว่าขีดจำกัดนี้จะถูกส่งกลับไปยัง ระบบอย่างช้าๆ

ส่งกลับ แสดงจำนวนบัฟเฟอร์ที่ถูกส่งกลับไปยังระบบ เมื่อจำนวนบัฟเฟอร์ว่าง เกินกว่าขีดจำกัด **hiwat**

คุณไม่ควรเห็นการเรียกที่ **ล้มเหลว** จำนวนมาก ค่านี้อาจมี ได้เล็กน้อย ซึ่งทริกเกอร์ระบบให้จัดสรรบัฟเฟอร์เพิ่มเติมเมื่อขนาดของบัฟเฟอร์ พูลเพิ่มขึ้น มีชุดของบัฟเฟอร์ที่กำหนดไว้แล้วของแต่ละขนาดที่ระบบใช้เริ่มต้น หลังจากการรีบูตแต่ละครั้ง และจำนวนบัฟเฟอร์จะเพิ่มขึ้น ตามความจำเป็น

ข้อมูลต่อไปนี้เป็นตัวอย่างของคำสั่ง **netstat −m** จาก เครื่องที่มีตัวประมวลผลหรือ CPU สองตัว:

```
# netstat -m

Kernel malloc statistics:

******* CPU 0 *******
By size           inuse     calls failed   delayed    free   hiwat   freed
32                   68       693      0         0      60    2320       0
64                   55       115      0         0       9    1160       0
128                  21       451      0         0      11     580       0
256                1064      5331      0         0    1384    1392      42
512                  41       136      0         0       7     145       0
1024                 10       231      0         0       6     362       0
2048               2049      4097      0         0     361     362     844
4096                  2         8      0         0     435     435     453
8192                  2         4      0         0       0      36       0
16384                 0       513      0         0      86      87     470


******* CPU 1 *******
By size           inuse     calls failed   delayed    free   hiwat   freed
32                  139       710      0         0     117    2320       0
64                   53       125      0         0      11    1160       0
128                  41       946      0         0      23     580       0
256                  62      7703      0         0    1378    1392     120
512                  37       109      0         0      11     145       0
1024                 21       217      0         0       3     362       0
2048                  2      2052      0         0     362     362     843
4096                  7        10      0         0     434     435     449
8192                  0         4      0         0       1      36       0
16384                 0      5023      0         0      87      87    2667


***** Allocations greater than 16384 Bytes *****

By size           inuse     calls failed   delayed    free   hiwat   freed
65536                 2         2      0         0       0    4096       0

Streams mblk statistic failures:
0 high priority mblk failures
0 medium priority mblk failures
0 low priority mblk failures
```

การปรับแคช ARP

Address Resolution Protocol (ARP) จะถูกใช้เพื่อแม็พแอดเดรส IPv4 ขนาด 32 บิต ลงในแอดเดรสอะแด็ปเตอร์โฮสต์ขนาด 48 บิต ที่ต้องการโดย data link protocol

ARP จะถูกจัดการอย่างโปร่งใสโดยระบบ อย่างไรก็ตาม ระบบจะยังคงรักษาแคช ARP ไว้ ซึ่งจะเป็นตารางที่เก็บ IP address ขนาด 32 บิตที่เชื่อมโยงถึง และโฮสต์แอดเดรสขนาด 48 บิต คุณอาจจำเป็นต้องเปลี่ยนขนาดของแคช ARP ในสภาวะแวดล้อมที่เชื่อมต่อกับจำนวนของเครื่อง (ไคลเอ็นต์) ที่มีขนาดใหญ่ ซึ่งสามารถทำได้โดยใช้คำสั่ง **no** และ **netstat**

คำสั่ง **no** จะปรับแต่งพารามิเตอร์การปรับเน็ตเวิร์ก และพารามิเตอร์ที่สามารถปรับแต่งได้ซึ่งเกี่ยวข้องกับ ARP คือ:

- **arpqsize = 12**
- **arpt_killc = 20**
- **arptab_bsiz = 7**
- **arptab_nb = 149**

ขนาดของตาราง ARP จะประกอบด้วยจำนวนของที่ฝากข้อมูล ซึ่งกำหนดโดยพารามิเตอร์ *arptab_nb* แต่ละที่ฝากข้อมูลจะเก็บจำนวนของรายการที่กำหนดไว้ในพารามิเตอร์ **arptab_bsiz** ค่าดีฟอลต์คือ 149 ที่ฝากข้อมูลพร้อมด้วย 7 รายการสำหรับแต่ละที่ฝากข้อมูล ดังนั้น ตารางสามารถเก็บโฮสต์แอดเดรสได้ 1043 (149 x 7) แอดเดรส ค่ากำหนดดีฟอลต์นี้จะทำงานกับระบบที่จะสื่อสารกับเครื่องอื่นๆ มากสุด 1043 เครื่องพร้อมกันบน IP network ถ้าเซิร์ฟเวอร์เชื่อมต่อกับเครื่องมากกว่า 1043 เครื่องบนเน็ตเวิร์กพร้อมกัน ตาราง ARP จะมีขนาดเล็กเกินไป ซึ่งเป็นสาเหตุทำให้ตาราง ARP พบกับปัญหาและทำให้ผลการทำงานแย่ลง ระบบปฏิบัติการต้องกำจัดรายการในแคช และแทนที่ด้วยแอดเดรสใหม่ ซึ่งต้องการ TCP หรือแพ็กเก็ต UDP เพื่อต่อคิว ขณะที่โปรโตคอล ARP จะทำการแลกเปลี่ยนข้อมูลนี้ พารามิเตอร์ *arpqsize* จะพิจารณาจำนวนของแพ็กเก็ตที่รอเหล่านี้ซึ่งสามารถต่อคิวได้โดยเลเยอร์ ARP จนกว่าการตอบกลับ ARP จะได้รับกลับมาจากคำร้องขอ ARP ถ้าคิว ARP ทำงานมากเกินไป TCP หรือแพ็กเก็ต UDP ขาออกจะหยุดทำงาน

แคช ARP ที่มีปัญหาอาจมีผลกระทบในทางลบกับผลการทำงาน ด้วยเหตุผลต่อไปนี้:

1. แพ็กเก็ตขาออกในปัจจุบันจะรอการค้นหาโปรโตคอล ARP ผ่านเน็ตเวิร์ก
2. รายการ ARP อื่นๆ ต้องถูกลบออกจากแคช ARP ถ้าแอดเดรสทั้งหมดเป็นแอดเดรสที่จำเป็นต้องมี แอดเดรสอื่นๆ อาจจำเป็นต้องมี เมื่อโฮสต์แอดเดรสที่ถูกลบ มีแพ็กเก็ตที่ส่งออกไปยังโฮสต์นั้น
3. เอาต์พุตคิว ARP อาจทำงานมากเกินไป ซึ่งอาจเป็นสาเหตุทำให้แพ็กเก็ตตกหล่น

พารามิเตอร์ *arpqsize*, *arptab_bsiz* และ *arptab_nb* คือพารามิเตอร์ที่ต้องรีบูตทั้งหมด นั่นหมายความว่า ระบบต้องถูกรีบูตหากค่ามีการเปลี่ยนแปลงไป เนื่องจากพารามิเตอร์เหล่านี้จะปรับเปลี่ยนตารางที่สร้าง ณ ตอนบูตหรือเวลาที่โหลด TCP/IP

พารามิเตอร์ *arpt_killc* คือเวลาในหน่วยนาที ก่อนที่รายการ ARP จะถูกลบทิ้ง ค่าดีฟอลต์ของพารามิเตอร์ *arpt_killc* คือ 20 นาที รายการ ARP จะถูกลบทิ้งจากตารางทุกจำนวนนาทีที่ได้กำหนดไว้ในพารามิเตอร์ **arpt_killc** เพื่อให้ครอบคลุมกรณีที่ระบบโฮสต์ต้องเปลี่ยนแอดเดรสขนาด 48 บิต ซึ่งสามารถเกิดขึ้นได้ เมื่อเปลี่ยน อะแด็ปเตอร์ของเน็ตเวิร์ก ซึ่งมั่นใจได้ว่า รายการเก่าใดๆ ในแคชได้ถูกลบทิ้งแล้ว และจะช่วยป้องกันการสื่อสารกับโฮสต์จนกระทั่งแอดเดรสเก่า ได้ถูกลบทิ้ง การเพิ่มเวลานี้จะช่วยลดการมองหา ARP โดยระบบ แต่อาจส่งผลให้มีการเก็บโฮสต์แอดเดรสเก่าที่ยาวนานขึ้น พารามิเตอร์ *arpt_killc* คือพารามิเตอร์แบบไดนามิก ดังนั้น จึงสามารถเปลี่ยนแปลงได้โดยไม่ต้องรีบูตระบบ

คำสั่ง **netstat -p arp** จะแสดงข้อมูลสถิติ ARP ซึ่งข้อมูลสถิติเหล่านี้จะแสดงจำนวนคำร้องขอ ARP ทั้งหมดที่ได้ส่งออก และจำนวนแพ็กเก็ตที่กำจัดออกจากตารางเมื่อรายการถูกลบทิ้งเพื่อสร้างห้อง สำหรับรายการใหม่ จำนวนที่กำจัดออกนี้จะมีค่าสูง ดังนั้น ขนาดของตาราง ARP ของคุณ ควรมีขนาดเพิ่มขึ้นด้วย ต่อไปนี้คือตัวอย่างของคำสั่ง **netstat -p arp**

```
# netstat -p arp

arp:
     6 packets sent
     0 packets purged
```

คุณสามารถแสดงตาราง ARP ได้ด้วยคำสั่ง **arp –a** เอาต์พุตคำสั่งจะแสดงแอดเดรสเหล่านี้ ซึ่งอยู่ในตาราง ARP และวิธีที่แอดเดรสเหล่านั้นแฮชและแฮชด้วยที่ฝากข้อมูลใด

```
 ? (10.3.6.1) at 0:6:29:dc:28:71 [ethernet] stored

bucket:    0     contains:    0 entries
bucket:    1     contains:    0 entries
bucket:    2     contains:    0 entries
bucket:    3     contains:    0 entries
bucket:    4     contains:    0 entries
bucket:    5     contains:    0 entries
bucket:    6     contains:    0 entries
bucket:    7     contains:    0 entries
bucket:    8     contains:    0 entries
bucket:    9     contains:    0 entries
bucket:   10     contains:    0 entries
bucket:   11     contains:    0 entries
bucket:   12     contains:    0 entries
bucket:   13     contains:    0 entries
bucket:   14     contains:    1 entries
bucket:   15     contains:    0 entries


...lines omitted...

There are 1 entries in the arp table.
```

การปรับการแก้ไขชื่อ

TCP/IP พยายามให้ได้รับที่อยู่ Internet Protocol (IP) จาก ชื่อโฮสต์ในกระบวนการที่เรียกว่า *การแก้ไขชื่อ*

กระบวนการแปลที่อยู่ Internet Protocol เป็นชื่อโฮสต์ เรียกกันว่า *การแก้ไขชื่อย้อนกลับ* ใช้ *รูทีนผู้แก้ไข* เพื่อแก้ไข ชื่อ รูทีนนี้จะเคียวรี DNS, NIS และไฟล์ /etc/hosts โลคัลในขั้นสุดท้าย เพื่อค้นหาข้อมูลที่ต้องการ

คุณสามารถปรับกระบวนการของการแก้ไขชื่อโดยการบันทึกทับลำดับการค้นหา ดีฟอลต์ ถ้าคุณทราบว่าจะแก้ไขชื่อได้อย่างไร ขั้นตอนนี้ทำ โดยใช้ไฟล์ /etc/netsvc.conf หรือตัวแปรสภาพแวดล้อม *NSORDER*

- ถ้าใช้ทั้งไฟล์ /etc/netsvc.conf และ *NSORDER NSORDER* จะแทนที่ไฟล์ /etc/netsvc.conf เมื่อต้องการระบุลำดับโฮสต์ด้วย /etc/netsvc.conf ให้สร้าง ไฟล์ที่มีบรรทัดดังต่อไปนี้:

  ```
  hosts=value,value,value
  ```

  โดยที่ *value* อาจเป็น (อักษรพิมพ์เล็กเท่านั้น) bind, local, nis, bind4, bind6, local4, local6, nis4, หรือ nis6 (สำหรับ /etc/hosts) ให้ระบุลำดับ บนบรรทัดเดียวกันกับค่าโดยคั่นด้วยเครื่องหมายจุลภาค อนุญาตให้มีพื้นที่ว่างสีขาวระหว่าง เครื่องหมายจุลภาคและเครื่องหมายเท่ากับได้

ค่าที่ระบุและลำดับของค่านั้น ขึ้นอยู่กับการตั้งค่าคอนฟิกเครือข่าย ตัวอย่างเช่น ถ้าเครือข่ายโลคัลมีการจัดระเบียบ เป็นเครือข่ายระนาบเดียว (flat) ผลคือต้องการไฟล์ /etc/hosts เพียง อย่างเดียวเท่านั้น ไฟล์ /etc/netsvc.conf จะมีบรรทัด ดังต่อไปนี้:

```
hosts=local
```

ตัวแปรสภาพแวดล้อม *NSORDER* จะมีการตั้งค่าเป็น:

```
NSORDER=local
```

- ถ้าเครือข่ายโลคัลเป็นเครือข่ายโดเมนที่ใช้เซิร์ฟเวอร์ชื่อสำหรับการแก้ไข ชื่อและไฟล์ /etc/hosts สำหรับแบ็คอัพ ให้ระบุการบริการ ทั้งสองอย่าง ไฟล์ /etc/netsvc.conf จะมีบรรทัด ดังต่อไปนี้:

  ```
  hosts=bind,local
  ```

  ตัวแปรสภาพแวดล้อม *NSORDER* จะมีการตั้งค่าเป็น:

  ```
  NSORDER=bind,local
  ```

ขั้นตอนวิธีจะพยายามใช้แหล่งข้อมูลแรกในรายการ จากนั้น ขั้นตอนวิธีจะพยายาม การบริการที่ระบุอื่นตามข้อมูลดังนี้:

- การบริการปัจจุบันไม่ได้รันอยู่ ดังนั้นจึงไม่มีอยู่
- การบริการปัจจุบันไม่พบชื่อและไม่ได้รับอนุญาต

การวิเคราะห์ประสิทธิภาพเครือข่าย

เมื่อเกิดปัญหาเกี่ยวกับประสิทธิภาพ ระบบของคุณอาจไม่รับรู้สิ่งใดเลย ในขณะที่มีการกระทำความผิดเกิดขึ้น

วิธีที่ง่ายในการกำหนดว่าเครือข่ายส่งผลกระทบต่อประสิทธิภาพโดยรวมหรือไม่คือ การเปรียบเทียบการดำเนินงานที่เกี่ยวข้องกับเครือข่ายกับการดำเนินงานที่ไม่เกี่ยวข้อง ถ้าคุณกำลังรันโปรแกรมที่มีการอ่านและบันทึกแบบรีโมตเป็นจำนวนมากและโปรแกรมรันได้ช้า แต่การดำเนินงานอื่นทุกอย่างรันได้เป็นปกติ แสดงว่าอาจเป็นปัญหาเครือข่าย ปัญหาคอขวดเครือข่ายที่อาจเกิดขึ้นได้ บางอย่างอาจเนื่องมาจากสาเหตุดังต่อไปนี้:

- อินเตอร์เฟสไคลเอ็นต์-เครือข่าย
- แบนด์วิธเครือข่าย
- ทอพอโลยีเครือข่าย
- อินเตอร์เฟสเซิร์ฟเวอร์เครือข่าย
- โหลด CPU ของเซิร์ฟเวอร์
- การใช้หน่วยความจำของเซิร์ฟเวอร์
- แบนด์วิธเซิร์ฟเวอร์
- การตั้งค่าคอนฟิกที่ไม่มีประสิทธิภาพ

เครื่องมือหลายอย่างประเมินสถิติเครือข่ายและให้ข้อมูลต่างๆ แต่เฉพาะส่วนของข้อมูลนี้เท่านั้นที่เกี่ยวข้องกับการปรับประสิทธิภาพ

เพื่อพัฒนาประสิทธิภาพ คุณสามารถใช้คำสั่ง **no** (อ็อพชัน เครือข่าย) และคำสั่ง **nfso** เพื่อปรับอ็อพชัน NFS คุณยังสามารถใช้คำสั่ง **chdev** และ **ifconfig** เพื่อเปลี่ยนพารามิเตอร์ของระบบและเครือข่ายได้ด้วย

คำสั่ง ping

คำสั่ง **ping** มีประโยชน์สำหรับการกำหนดสถานะ ของเครือข่ายและโฮสต์ต่างประเทศต่างๆ การติดตามและการแยกปัญหาฮาร์ดแวร์ และซอฟต์แวร์ และการทดสอบ การประเมิน และการจัดการเครือข่าย

อ็อพชันบางรายการของคำสั่ง **ping** ที่เกี่ยวข้องกับการปรับ ประสิทธิภาพมีดังนี้:

-c ระบุจำนวนแพ็กเก็ต อ็อพชันนี้มีประโยชน์เมื่อคุณได้รับบันทึกการติดตาม IP คุณสามารถตรวจจับจำนวนต่ำสุดของ **ping** แพ็กเก็ต

-s ระบุความยาวของแพ็กเก็ต คุณสามารถใช้อ็อพชันนี้เพื่อตรวจสอบการแบ่งเฟรกเมนต์ และการประกอบใหม่

-f ส่งแพ็กเก็ตในช่วงเวลา 10 ms หรือส่งในทันทีหลังจากแต่ละการตอบกลับ เฉพาะผู้ใช้รากเท่านั้นสามารถใช้อ็อพชันนี้ได้

ถ้าคุณต้องการโหลดเครือข่ายหรือระบบของคุณ อ็อพชัน **-f** ถือเป็นวิธีที่สะดวก ตัวอย่างเช่น ถ้าคุณสงสัยว่าปัญหาของคุณเกิดจากโหลดหนัก ให้โหลดสภาพแวดล้อม ของคุณโดยตั้งใจเพื่อยืนยันข้อสงสัยของคุณ เปิดหลายหน้าต่าง **aixterm** และรัน คำสั่ง **ping -f** ในแต่ละหน้าต่าง การใช้ประโยชน์ อีเทอร์เน็ตของคุณจะเพิ่มขึ้นถึงประมาณ 100 เปอร์เซ็นต์อย่างรวดเร็ว ข้อมูลต่อไปนี้เป็น ตัวอย่าง:

```
# date; ping -c 1000 -f 192.1.6.1 ; date
Thu Feb 12 10:51:00 CST 2004
PING 192.1.6.1 (192.1.6.1): 56 data bytes
.
--- 192.1.6.1 ping statistics ---
1000 packets transmitted, 1000 packets received, 0% packet loss
round-trip min/avg/max = 1/1/23 ms
Thu Feb 12 10:51:00 CST 2004
```

หมายเหตุ: คำสั่ง **ping** อาจส่งผล กระทบอย่างมากบนเครือข่าย ดังนั้นควรใช้ด้วยความระมัดระวัง Flood-pinging สามารถทำได้โดยผู้ใช้รากเท่านั้น

ในตัวอย่างเช่น 1000 แพ็กเก็ตถูกส่งภายใน 1 วินาที ระวังว่า คำสั่งนี้ใช้ IP และโปรโตคอล Internet Control Message Protocol (ICMP) ดังนั้นจึงไม่มีกิจกรรมโปรโตคอลการขนส่ง (UDP/TCP) และแอ็พพลิเคชัน เกี่ยวข้อง ข้อมูลที่ประเมินได้ เช่น เวลาเดินทางไปกลับ ไม่ได้สะท้อนถึง ลักษณะประสิทธิภาพทั้งหมด

เมื่อคุณพยายามส่ง flood ของแพ็กเก็ตไปยังปลายทาง ให้พิจารณาประเด็นต่างๆ ดังนี้:

- การส่งแพ็กเก็ตทำให้เกิดโหลดบนระบบของคุณ
- ใช้คำสั่ง **netstat -i** เพื่อมอนิเตอร์สถานะของ อินเตอร์เฟสเครือข่ายในระหว่างการทดลอง คุณอาจพบว่าระบบกำลังทิ้งแพ็กเก็ตในระหว่างการส่งโดยดูที่เอาต์พุต `Oerrs`
- คุณยังควรมอนิเตอร์รีซอร์สอื่น เช่น mbufs และคิวการส่ง/การรับ ด้วย เป็นการยากที่จะวางโหลดหนักไว้บนระบบปลายทาง ระบบของคุณอาจจะโอเวอร์โหลดก่อนระบบอื่น
- พิจารณาความสัมพันธ์ของผลลัพธ์ ถ้าคุณต้องการมอนิเตอร์หรือทดสอบ ระบบปลายทางเพียงระบบเดียว ให้ทำการทดลองเดียวกันบนระบบอื่น เพื่อเปรียบเทียบ เนื่องจากเครือข่ายหรือเราเตอร์ของคุณอาจมีปัญหา

คำสั่ง ftp

คุณสามารถใช้คำสั่ง **ftp** เพื่อส่งไฟล์ขนาดใหญ่มากโดยใช้ **/dev/zero** เป็นอินพุตและ **/dev/null** เป็นเอาต์พุต ซึ่งอนุญาตให้คุณถ่ายโอนไฟล์ขนาดใหญ่โดยไม่เกี่ยวข้องกับดิสก์ (อาจเป็นคอขวด) และไม่มีการทำแคชไฟล์ทั้งหมดในหน่วยความจำ

ให้ใช้คำสั่งย่อยต่อไปนี้ **ftp** (เปลี่ยนจำนวนให้เพิ่มขึ้น หรือลดจำนวนของบล็อกที่อ่านโดยคำสั่ง **dd**):

```
> bin
> put "|dd if=/dev/zero bs=32k count=10000" /dev/null
```

คำสั่งข้างต้นจะถ่ายโอนบล็อก 10000 บล็อกของข้อมูลและแต่ละบล็อกมีขนาด 32 KB หากต้องการเพิ่มขนาดหรือลดขนาดของไฟล์ที่ถ่ายโอน ให้เปลี่ยนจำนวนของบล็อกที่อ่านโดยคำสั่ง **dd** ซึ่งคือพารามิเตอร์ `count` หรือโดยการเปลี่ยนขนาดบล็อกซึ่งคือพารามิเตอร์ `bs` หมายเหตุ ชนิดไฟล์ดีฟอลต์สำหรับคำสั่ง **ftp** คือ `ASCII` ซึ่งจะช้ากว่าจำนวนไบต์ทั้งหมดที่ได้สแกนแล้ว โหมดไบนารี หรือ `bin` ควรจะถูกใช้สำหรับถ่ายโอนเมื่อเป็นไปได้

ตรวจสอบให้แน่ใจว่า **tcp_sendspace** และ **tcp_recvspace** ต้องมีอย่างน้อย 65535 สำหรับกิกะบิตอีเทอร์เน็ตที่มี "กรอบขนาดใหญ่มาก" และสำหรับ ATM ด้วย MTU 9180 หรือใหญ่กว่า เพื่อรับผลการทำงานที่ดีกว่า เนื่องจากขนาดของ MTU ที่ใหญ่กว่าขอแนะนำให้ใช้ขนาด 131072 ไบต์ (128 KB) สำหรับผลการทำงานที่เหมาะสม ถ้าคุณปรับแต่งอะแด็ปเตอร์กิกะบิตอีเทอร์เน็ตด้วยเครื่องมือ **SMIT** ค่าดีฟอลต์ของระบบ ISNO ควรถูกตั้งค่า อย่างถูกต้อง อ็อพชัน ISNO ไม่ได้รับการตั้งค่าอย่างถูกต้อง ถ้าคุณใช้คำสั่ง **ifconfig** เพื่อนำเสนอเน็ตเวิร์กอินเตอร์เฟส

ตัวอย่างของการตั้งค่าพารามิเตอร์เป็นดังนี้:

```
# no -o tcp_sendspace=65535
# no -o tcp_recvspace=65535
```

คำสั่งย่อย **ftp** มีดังต่อไปนี้:

```
ftp> bin
200 Type set to I.
ftp> put "|dd if=/dev/zero bs=32k count=10000" /dev/null
200 PORT command successful.
150 Opening data connection for /dev/null.
10000+0 records in
10000+0 records out
226 Transfer complete.
327680000 bytes sent in 2.789 seconds (1.147e+05 Kbytes/s)
local: |dd if=/dev/zero bs=32k count=10000 remote: /dev/null
ftp> quit
221 Goodbye.
```

การถ่ายโอนข้อมูลข้างต้นถูกเรียกใช้งานระหว่างอะแด็ปเตอร์กิกะบิตอีเทอร์เน็ต สองตัวโดยใช้ 1500 ไบต์ MTU และทรูพุตถูกรายงานดังนี้ : 114700 KB/วินาที ซึ่งเทียบเท่ากับ 112 MB/วินาที หรือ 940 Mbps

เมื่อผู้ส่งและผู้รับใช้กรอบขนาดใหญ่มาก ด้วย MTU ขนาด 9000 ทรูพุตที่รายงานคือ 120700 KB/วินาที หรือ 117.87 MB/วินาที หรือ 989 Mbps ตามที่คุณมองเห็นในตัวอย่างต่อไปนี้:

```
ftp> bin
200 Type set to I.
ftp> put "|dd if=/dev/zero bs=32k count=10000" /dev/null
200 PORT command successful.
150 Opening data connection for /dev/null.
10000+0 records in
10000+0 records out
226 Transfer complete.
327680000 bytes sent in 2.652 seconds (1.207e+05 Kbytes/s)
local: |dd if=/dev/zero bs=32k count=10000 remote: /dev/null
```

ต่อไปนี้คือตัวอย่างของการถ่ายโอนข้อมูล **ftp** ระหว่างอีเทอร์เน็ตอินเตอร์เฟสขนาด 10/100 Mbps สองตัว:

```
ftp> bin
200 Type set to I.
ftp> put "|dd if=/dev/zero bs=32k count=10000" /dev/null
200 PORT command successful.
150 Opening data connection for /dev/null.
10000+0 records in
10000+0 records out
226 Transfer complete.
327680000 bytes sent in 27.65 seconds (1.157e+04 Kbytes/s)
local: |dd if=/dev/zero bs=32k count=10000 remote: /dev/null
```

ทรูพุตของการถ่ายโอนข้อมูลคือ 11570 KB/วินาที ซึ่งเทียบเท่ากับ 11.3 MB/วินาทีหรือ 94.7 Mbps

คำสั่ง netstat

คำสั่ง **netstat** ใช้เพื่อแสดงสถานะ เครือข่าย

โดยดั้งเดิม คำสั่งนี้ใช้เพื่อกำหนดปัญหามากกว่าการประเมิน ประสิทธิภาพ อย่างไรก็ตาม สามารถใช้คำสั่ง **netstat** เพื่อกำหนดจำนวนการจราจรบนเครือข่ายเพื่อให้แน่ใจว่าปัญหาประสิทธิภาพ เกิดจากความคับคั่งของเครือข่ายหรือไม่

คำสั่ง **netstat** แสดงข้อมูลเกี่ยวกับการจราจร บนอินเตอร์เฟสเครือข่ายที่ตั้งค่าคอนฟิก เช่นข้อมูลดังต่อไปนี้:

- ที่อยู่ของบล็อกควบคุมโปรโตคอลใดๆ ที่เชื่อมโยงกับซ็อกเก็ต และสถานะของซ็อกเก็ตทั้งหมด
- จำนวนของแพ็กเก็ตที่ได้รับ ส่งผ่าน และที่ทิ้งในระบบย่อย การสื่อสาร
- สถิติรวมต่ออินเตอร์เฟส
- เราต์และสถานะของเราต์

การใช้คำสั่ง netstat:

คำสั่ง **netstat** แสดงเนื้อหาของโครงสร้างข้อมูลที่เกี่ยวข้อง กับเครือข่ายต่างๆ สำหรับการเชื่อมต่อที่ใช้งานอยู่

คำสั่ง netstat -in:

ฟังก์ชัน **netstat** นี้แสดงสถานะของอินเตอร์เฟสทั้งหมด ที่ตั้งค่าคอนฟิก

ตัวอย่างต่อไปนี้แสดงสถิติสำหรับเวิร์กสเตชันที่มี Ethernet แบบรวม (en1), PCI-X Gigabit Ethernet (en0) และ Fibre Channel Adapter ที่กำหนดค่าสำหรับ TCP/IP (fc0):

```
# netstat -in
Name  Mtu   Network     Address              Ipkts Ierrs    Opkts Oerrs  Coll
en1   1500  link#2      0.9.6b.3e.0.55       28800     0      506     0     0
en1   1500  10.3.104    10.3.104.116         28800     0      506     0     0
fc0   65280 link#3      0.0.c9.33.17.46         12     0       11     0     0
fc0   65280 192.6.0     192.6.0.1               12     0       11     0     0
en0   1500  link#4      0.2.55.6a.a5.dc         14     0       20     5     0
en0   1500  192.1.6     192.1.6.1               14     0       20     5     0
lo0   16896 link#1                          33339     0    33343     0     0
lo0   16896 127         127.0.0.1            33339     0    33343     0     0
```

ค่าจำนวนนับเป็นค่าสรุปตั้งแต่ระบบสตาร์ทอัพ

ชื่อ ชื่ออินเตอร์เฟส

Mtu หน่วยการส่งผ่านสูงสุด ขนาดสูงสุดของแพ็กเก็ตในหน่วยไบต์ที่มีการ ส่งผ่านโดยใช้อินเตอร์เฟส

Ipkts จำนวนแพ็กเก็ตทั้งหมดที่ได้รับ

Ierrs จำนวนทั้งหมดของข้อผิดพลาดอินพุต ตัวอย่างเช่น แพ็กเก็ตที่บกพร่อง ข้อผิดพลาด checksum, หรือพื้นที่ว่างบัฟเฟอร์ไม่เพียงพอในไดรเวอร์อุปกรณ์

Opkts จำนวนแพ็กเก็ตทั้งหมดที่ส่งผ่าน

Oerrs จำนวนทั้งหมดของข้อผิดพลาดเอาต์พุต ตัวอย่างเช่น ความบกพร่องในการเชื่อมต่อโลคัล โฮสต์หรือการโอเวอร์รันเอาต์พุตคิวของอะแด็ปเตอร์

Coll จำนวนของการปะทะระหว่างแพ็กเก็ตที่ตรวจพบ

หมายเหตุ: คำสั่ง **netstat –i** ไม่สนับสนุนการนับจำนวนการปะทะ สำหรับอีเทอร์เน็ตอินเตอร์เฟส (ให้ดู “ข้อมูลสถิติของอะแด็ปเตอร์” ในหน้า 358 สำหรับสถิติอีเทอร์เน็ต)

ข้อมูลต่อไปนี้เป็นคำแนะนำการปรับบางอย่าง:

- ถ้าจำนวนข้อผิดพลาดในระหว่างอินพุตแพ็กเก็ตมากกว่า 1 เปอร์เซ็นต์ ของจำนวนอินพุตแพ็กเก็ตทั้งหมด (จากคำสั่ง **netstat –i**); นั่นคือ

  ```
  Ierrs > 0.01 x Ipkts
  ```

  ให้รันคำสั่ง **netstat –m** เพื่อตรวจสอบการขาดแคลนหน่วยความจำ
- ถ้าจำนวนข้อผิดพลาดในระหว่างเอาต์พุตแพ็กเก็ตมากกว่า 1 เปอร์เซ็นต์ ของจำนวนเอาต์พุตแพ็กเก็ตทั้งหมด (จากคำสั่ง **netstat –i**); นั่นคือ

  ```
  Oerrs > 0.01 x Opkts
  ```

  ให้เพิ่ม ขนาดของคิวการส่ง (*xmt_que_size*) สำหรับอินเตอร์เฟสนั้น ขนาดของ *xmt_que_size* สามารถตรวจสอบได้โดยใช้คำสั่งต่อไปนี้:

  ```
  # lsattr -El adapter
  ```
- ถ้าอัตราการปะทะมากกว่า 10 เปอร์เซ็นต์ นั่นคือ

  ```
  Coll / Opkts > 0.1
  ```

  แสดงว่า มีการใช้ประโยชน์เครือข่ายสูง และอาจจำเป็นต้องใช้การจัดระเบียบใหม่หรือ การแบ่งพาร์ติชัน ใช้คำสั่ง **netstat –v** หรือ **entstat** เพื่อกำหนดอัตราการปะทะ

คำสั่ง netstat –i –Z:

ฟังก์ชันนี้ของคำสั่ง **netstat** ล้างข้อมูลตัวนับ สถิติทั้งหมดของคำสั่ง **netstat –i** เป็น ศูนย์

ช่วงเวลาอินเตอร์เฟส netstat –I:

ฟังก์ชัน **netstat** นี้แสดงสถิติ เกี่ยวกับอินเตอร์เฟสที่ระบุ

ฟังก์ชันนี้แสดงข้อมูลคล้ายกับคำสั่ง **netstat –i** เกี่ยวกับ อินเตอร์เฟสที่ระบุ และรายงานข้อมูลดังกล่าวในช่วงเวลาที่กำหนด ตัวอย่าง เช่น:

```
# netstat -I en0 1
    input   (en0)      output           input   (Total)    output
   packets  errs  packets  errs colls  packets  errs  packets   errs colls
       0     0       27     0     0   799655     0   390669     0     0
       0     0        0     0     0        2     0        0     0     0
       0     0        0     0     0        1     0        0     0     0
       0     0        0     0     0       78     0      254     0     0
       0     0        0     0     0      200     0       62     0     0
       0     0        1     0     0        0     0        2     0     0
```

ตัวอย่างก่อนหน้านี้แสดงเอาต์พุตคำสั่ง netstat -I สำหรับ อินเตอร์เฟส ent0 **มีการสร้างรายงานขึ้นพร้อมกันสองฉบับ รายงานหนึ่งสำหรับอินเตอร์เฟสที่ระบุและอีกรายงานหนึ่งสำหรับอินเตอร์เฟสทั้งหมดที่มีอยู่ (ทั้งหมด) ฟิลด์ต่างๆ คล้ายกับฟิลด์ใน netstat -i ตัวอย่างเช่น** อินพุต แพ็กเก็ต = Ipkts, input errs = Ierrs **และ อื่นๆ**

คำสั่ง netstat -a:

คำสั่ง netstat -a แสดงสถานะ ของซ็อกเก็ตทั้งหมด

โดยไม่มีแฟล็ก -a ซ็อกเก็ตที่ใช้โดยกระบวนการเซิร์ฟเวอร์จะไม่แสดงขึ้น ตัวอย่างเช่น:

```
# netstat -a
การเชื่อมต่ออินเทอร์เน็ตที่ใช้งานอยู่ (รวมถึงเซิร์ฟเวอร์)
Proto Recv-Q Send-Q  Local Address          Foreign Address        (สถานะ)
tcp4       0      0  *.daytime              *.*                    LISTEN
tcp        0      0  *.ftp                  *.*                    LISTEN
tcp        0      0  *.telnet               *.*                    LISTEN
tcp4       0      0  *.time                 *.*                    LISTEN
tcp4       0      0  *.sunrpc               *.*                    LISTEN
tcp        0      0  *.exec                 *.*                    LISTEN
tcp        0      0  *.login                *.*                    LISTEN
tcp        0      0  *.shell                *.*                    LISTEN
tcp4       0      0  *.klogin               *.*                    LISTEN
tcp4       0      0  *.kshell               *.*                    LISTEN
tcp        0      0  *.netop                *.*                    LISTEN
tcp        0      0  *.netop64              *.*                    LISTEN
tcp4       0   1028  brown10.telnet         remote_client.mt.1254  ESTABLISHED
tcp4       0      0  *.wsmserve             *.*                    LISTEN
udp4       0      0  *.daytime              *.*
udp4       0      0  *.time                 *.*
udp4       0      0  *.sunrpc               *.*
udp4       0      0  *.ntalk                *.*
udp4       0      0  *.32780                *.*
UNIX โดเมนซ็อกเก็ตที่ใช้งานอยู่
SADR/PCB  Type   Recv-Q Send-Q Inode    Conn     Refs     Nextref  Addr
71759200 dgram       0      0 13434d00        0        0        0 /dev/SRC
7051d580
71518a00 dgram       0      0 183c3b80        0        0        0 /dev/.SRC-unix/SRCCwfCEb
```

คุณสามารถดูข้อมูลรายละเอียดสำหรับแต่ละซ็อกเก็ตได้โดยใช้คำสั่ง netstat -ao ในตัวอย่างต่อไปนี้ ซ็อกเก็ต ftp รันบนอะแด็ปเตอร์กิกะบิตอีเทอร์เน็ต ที่ตั้งค่าคอนฟิกสำหรับ jumbo frames:

```
# netstat -ao

การเชื่อมต่ออินเทอร์เน็ตที่ใช้งานอยู่ (รวมถึงเซิร์ฟเวอร์)
```

```
Proto Recv-Q Send-Q  Local Address          Foreign Address        (สถานะ)
[...]

tcp4       0      0  server1.ftp           client1.33122         ESTABLISHED

         so_options: (REUSEADDR|OOBINLINE)
         so_state: (ISCONNECTED|PRIV)
         timeo:0 uid:0
         so_special: (LOCKBALE|MEMCOMPRESS|DISABLE)
         so_special2: (PROC)
         sndbuf:
                 hiwat:134220 lowat:33555 mbcnt:0 mbmax:536880
         rcvbuf:
                 hiwat:134220 lowat:1 mbcnt:0 mbmax:536880
                 sb_flags: (WAIT)
         TCP:
         mss:8948  flags: (NODELAY|RFC1323|SENT_WS|RCVD_WS|SENT_TS|RCVD_TS)


tcp4       0      0  server1.telnet         sig-9-49-151-26..2387  ESTABLISHED

         so_options: (REUSEADDR|KEEPALIVE|OOBINLINE)
         so_state: (ISCONNECTED|NBIO)
         timeo:0 uid:0
         so_special: (NOUAREA|LOCKBALE|EXTPRIV|MEMCOMPRESS|DISABLE)
         so_special2: (PROC)
         sndbuf:
                 hiwat:16384 lowat:4125 mbcnt:0 mbmax:65536
                 sb_flags: (SEL|NOINTR)
         rcvbuf:
                 hiwat:66000 lowat:1 mbcnt:0 mbmax:264000
                 sb_flags: (SEL|NOINTR)
         TCP:
         mss:1375

tcp4       0    925  en6host1.login         en6host2.1023          ESTABLISHED

         so_options: (REUSEADDR|KEEPALIVE|OOBINLINE)
         so_state: (ISCONNECTED|NBIO)
         timeo:0 uid:0
         so_special: (NOUAREA|LOCKBALE|EXTPRIV|MEMCOMPRESS|DISABLE)
         so_special2: (PROC)
         sndbuf:
                 hiwat:16384 lowat:16384 mbcnt:3216 mbmax:65536
                 sb_flags: (SEL|NOINTR)
         rcvbuf:
                 hiwat:130320 lowat:1 mbcnt:0 mbmax:521280
                 sb_flags: (SEL|NOINTR)
         TCP:
         mss:1448  flags: (RFC1323|SENT_WS|RCVD_WS|SENT_TS|RCVD_TS)


tcp        0      0  *.login                *.*                    LISTEN
```

```
         so_options: (ACCEPTCONN|REUSEADDR)
         q0len:0 qlen:0 qlimit:1000      so_state: (PRIV)
         timeo:0 uid:0
         so_special: (LOCKBALE|MEMCOMPRESS|DISABLE)
         so_special2: (PROC)
         sndbuf:
                 hiwat:16384 lowat:4096 mbcnt:0 mbmax:65536
         rcvbuf:
                 hiwat:16384 lowat:1 mbcnt:0 mbmax:65536
                 sb_flags: (SEL)
         TCP:
         mss:512

tcp        0      0  *.shell               *.*                   LISTEN

         so_options: (ACCEPTCONN|REUSEADDR)
         q0len:0 qlen:0 qlimit:1000      so_state: (PRIV)
         timeo:0 uid:0
         so_special: (LOCKBALE|MEMCOMPRESS|DISABLE)
         so_special2: (PROC)
         sndbuf:
                 hiwat:16384 lowat:4096 mbcnt:0 mbmax:65536
         rcvbuf:
                 hiwat:16384 lowat:1 mbcnt:0 mbmax:65536
                 sb_flags: (SEL)
         TCP:
         mss:512


tcp4       0   6394  brown10.telnet         remote_client.mt.1254  ESTABLISHED

         so_options: (REUSEADDR|KEEPALIVE|OOBINLINE)
         so_state: (ISCONNECTED|NBIO)
         timeo:0 uid:0
         so_special: (NOUAREA|LOCKBALE|EXTPRIV|MEMCOMPRESS|DISABLE)
         so_special2: (PROC)
         sndbuf:
                 hiwat:16384 lowat:4125 mbcnt:65700 mbmax:65536
                 sb_flags: (SEL|NOINTR)
         rcvbuf:
                 hiwat:16500 lowat:1 mbcnt:0 mbmax:66000
                 sb_flags: (SEL|NOINTR)
         TCP:
         mss:1375


udp4       0      0  *.time                *.*

         so_options: (REUSEADDR)
         so_state: (PRIV)
         timeo:0 uid:0
         so_special: (LOCKBALE|DISABLE)
         so_special2: (PROC)
         sndbuf:
                 hiwat:9216 lowat:4096 mbcnt:0 mbmax:36864
```

```
         rcvbuf:
                 hiwat:42080 lowat:1 mbcnt:0 mbmax:168320
                 sb_flags: (SEL)

[...]

UNIX โดเมนซ็อกเก็ตที่ใช้งานอยู่
SADR/PCB  Type   Recv-Q Send-Q Inode    Conn     Refs     Nextref  Addr
71759200 dgram       0      0 13434d00        0        0        0 /dev/SRC
7051d580
         so_state: (PRIV)
         timeo:0 uid:0
         so_special: (LOCKBALE)
         so_special2: (PROC)
         sndbuf:
                 hiwat:8192 lowat:4096 mbcnt:0 mbmax:32768
         rcvbuf:
                 hiwat:45000 lowat:1 mbcnt:0 mbmax:180000
                 sb_flags: (SEL)

71518a00 dgram       0      0 183c3b80        0        0        0 /dev/.SRC-unix/SRCCwfCEb7051d400
         so_state: (PRIV)
         timeo:0 uid:0
         so_special: (LOCKBALE)
         so_special2: (PROC)
         sndbuf:
                 hiwat:16384 lowat:4096 mbcnt:0 mbmax:65536
         rcvbuf:
                 hiwat:8192 lowat:1 mbcnt:0 mbmax:32768
                 sb_flags: (SEL)

[...]
```

ในตัวอย่างข้างบน อะแด็ปเตอร์มีการตั้งค่าคอนฟิกสำหรับ jumbo frames ซึ่งเป็นเหตุผลให้ค่า MSS มีจำนวนมากและมีการตั้งค่า **rfc1323**

คำสั่ง netstat -M:

คำสั่ง **netstat -M** แสดงสถิติคลัสเตอร์พูลของ หน่วยความจำเครือข่าย

ตัวอย่างต่อไปนี้แสดงเอาต์พุตของคำสั่ง **netstat -M**:

```
# netstat -M
Cluster pool Statistics:

  Cluster Size   Pool Size   Calls   Failed   Inuse   Max Outcount
        131072           0       0        0       0              0
         65536           0       0        0       0              0
         32768           0       0        0       0              0
         16384           0       0        0       0              0
          8192           0  191292        3       0              3
          4096           0  196021        3       0              3
          2048           0  140660        4       0              2
          1024           0       2        1       0              1
```

```
   512          0       2        1       0             1
131072          0       0        0       0             0
 65536          0       0        0       0             0
 32768          0       0        0       0             0
 16384          0       0        0       0             0
  8192          0  193948        2       0             2
  4096          0  191122        3       0             3
  2048          0  145477        4       0             2
  1024          0       0        0       0             0
   512          0       2        1       0             1
```

คำสั่ง netstat -v:

คำสั่ง netstat -v แสดงสถิติสำหรับแต่ละ Common Data Link Interface (CDLI)**-ซึ่งขึ้นอยู่กับไดรเวอร์อุปกรณ์ที่ใช้งานอยู่**

รายงานเฉพาะอินเตอร์เฟสสามารถร้องขอได้โดยใช้คำสั่ง tokstat, entstat, fddistat, หรือ atmstat

ทุกอินเตอร์เฟสมีข้อมูลเฉพาะของตนเองและข้อมูลทั่วไปบางอย่าง ตัวอย่างต่อไปนี้แสดงส่วน Token-Ring **และอีเทอร์เน็ตของคำสั่ง netstat -v โดยส่วนประกอบอินเตอร์เฟสอื่นๆ มีลักษณะคล้ายกัน เมื่อใช้อะแด็ปเตอร์ที่แตกต่างอื่น ข้อมูลสถิติจะแตกต่างไปเล็กน้อย มีการไฮไลต์ฟิลด์เอาต์พุตที่สำคัญที่สุด**

```
# netstat -v
-------------------------------------------------------------
ETHERNET STATISTICS (ent1) :
Device Type: 10/100 Mbps Ethernet PCI Adapter II (1410ff01)
Hardware Address: 00:09:6b:3e:00:55
Elapsed Time: 0 days 17 hours 38 minutes 35 seconds

Transmit Statistics:                          Receive Statistics:
--------------------                          -------------------
Packets: 519                                  Packets: 30161
Bytes: 81415                                  Bytes: 7947141
Interrupts: 2                                 Interrupts: 29873
Transmit Errors: 0                            Receive Errors: 0
Packets Dropped: 0                            Packets Dropped: 0
                                              Bad Packets: 0
Max Packets on S/W Transmit Queue: 3
S/W Transmit Queue Overflow: 0
Current S/W+H/W Transmit Queue Length: 1

Broadcast Packets: 3                          Broadcast Packets: 29544
Multicast Packets: 2                          Multicast Packets: 42
No Carrier Sense: 0                           CRC Errors: 0
DMA Underrun: 0                               DMA Overrun: 0
Lost CTS Errors: 0                            Alignment Errors: 0
Max Collision Errors: 0                       No Resource Errors: 0
Late Collision Errors: 0                      Receive Collision Errors: 0
Deferred: 0                                   Packet Too Short Errors: 0
SQE Test: 0                                   Packet Too Long Errors: 0
Timeout Errors: 0                             Packets Discarded by Adapter: 0
Single Collision Count: 0                     Receiver Start Count: 0
Multiple Collision Count: 0
Current HW Transmit Queue Length: 1
```

```
General Statistics:
-------------------
No mbuf Errors: 0
Adapter Reset Count: 0
Adapter Data Rate: 200
Driver Flags: Up Broadcast Running
        Simplex AlternateAddress 64BitSupport
        ChecksumOffload PrivateSegment DataRateSet

10/100 Mbps Ethernet PCI Adapter II (1410ff01) Specific Statistics:
--------------------------------------------------------------------
Link Status: Up
Media Speed Selected: Auto negotiation
Media Speed Running: 100 Mbps Full Duplex
Receive Pool Buffer Size: 1024
Free Receive Pool Buffers: 1024
No Receive Pool Buffer Errors: 0
Receive Buffer Too Small Errors: 0
Entries to transmit timeout routine: 0
Transmit IPsec packets: 0
Transmit IPsec packets dropped: 0
Receive IPsec packets: 0
Receive IPsec packets dropped: 0
Inbound IPsec SA offload count: 0
Transmit Large Send packets: 0
Transmit Large Send packets dropped: 0
Packets with Transmit collisions:
 1 collisions: 0           6 collisions: 0          11 collisions: 0
 2 collisions: 0           7 collisions: 0          12 collisions: 0
 3 collisions: 0           8 collisions: 0          13 collisions: 0
 4 collisions: 0           9 collisions: 0          14 collisions: 0
 5 collisions: 0          10 collisions: 0          15 collisions: 0
-------------------------------------------------------------
ETHERNET STATISTICS (ent0) :
Device Type: 10/100/1000 Base-TX PCI-X Adapter (14106902)
Hardware Address: 00:02:55:6a:a5:dc
Elapsed Time: 0 days 17 hours 0 minutes 26 seconds

Transmit Statistics:                        Receive Statistics:
--------------------                        -------------------
Packets: 15                                 Packets: 14
Bytes: 1037                                 Bytes: 958
Interrupts: 0                               Interrupts: 13
Transmit Errors: 0                          Receive Errors: 0
Packets Dropped: 0                          Packets Dropped: 0
                                            Bad Packets: 0
Max Packets on S/W Transmit Queue: 4
S/W Transmit Queue Overflow: 0
Current S/W+H/W Transmit Queue Length: 0

Broadcast Packets: 1                        Broadcast Packets: 0
Multicast Packets: 1                        Multicast Packets: 0
No Carrier Sense: 0                         CRC Errors: 0
DMA Underrun: 0                             DMA Overrun: 0
Lost CTS Errors: 0                          Alignment Errors: 0
```

```
Max Collision Errors: 0                     No Resource Errors: 0
Late Collision Errors: 0                    Receive Collision Errors: 0
Deferred: 0                                 Packet Too Short Errors: 0
SQE Test: 0                                 Packet Too Long Errors: 0
Timeout Errors: 0                           Packets Discarded by Adapter: 0
Single Collision Count: 0                   Receiver Start Count: 0
Multiple Collision Count: 0
Current HW Transmit Queue Length: 0

General Statistics:
-------------------
No mbuf Errors: 0
Adapter Reset Count: 0
Adapter Data Rate: 2000
Driver Flags: Up Broadcast Running
        Simplex 64BitSupport ChecksumOffload
        PrivateSegment LargeSend DataRateSet

10/100/1000 Base-TX PCI-X Adapter (14106902) Specific Statistics:
---------------------------------------------------------------------
Link Status: Up
Media Speed Selected: Auto negotiation
Media Speed Running: 1000 Mbps Full Duplex
PCI Mode: PCI-X (100-133)
PCI Bus Width: 64-bit
Jumbo Frames: Disabled
TCP Segmentation Offload: Enabled
        TCP Segmentation Offload Packets Transmitted: 0
        TCP Segmentation Offload Packet Errors: 0
Transmit and Receive Flow Control Status: Enabled
        XON Flow Control Packets Transmitted: 0
        XON Flow Control Packets Received: 0
        XOFF Flow Control Packets Transmitted: 0
        XOFF Flow Control Packets Received: 0
Transmit and Receive Flow Control Threshold (High): 32768
Transmit and Receive Flow Control Threshold (Low): 24576
Transmit and Receive Storage Allocation (TX/RX): 16/48
```

ฟิลด์ที่ไฮไลต์มีการอธิบายดังนี้:

- **Transmit and Receive Errors**

  จำนวนของข้อผิดพลาดเอาต์พุต/อินพุตที่พบ บนอุปกรณ์นี้ ฟิลด์นี้นับการส่งผ่านที่ไม่สำเร็จเนื่องจากข้อผิดพลาดฮาร์ดแวร์/ เครือข่าย

  การส่งผ่านที่ไม่สำเร็จเหล่านี้ยังอาจทำให้ประสิทธิภาพของระบบ ช้าลงอีกด้วย

- **Max Packets on S/W Transmit Queue**

  จำนวนสูงสุดของแพ็กเก็ตขาออก ที่เคยจัดคิวอยู่ในคิวการส่งผ่านของซอฟต์แวร์

  สิ่งที่บ่งชี้ถึงขนาดคิว ที่ผิดปกติคือ ถ้าการส่งผ่านสูงสุดที่จัดคิวเท่ากับ ขนาดคิวปัจจุบัน (*xmt_que_size*) ลักษณะนี้บ่งชี้ว่าคิวเต็มในบางเวลา

  เมื่อต้องการตรวจสอบ ขนาดปัจจุบันของคิว ให้ใช้คำสั่ง **lsattr -El** *adapter* (โดยที่ adapter คือ ตัวอย่างเช่น `ent0`) เนื่องจากคิว เชื่อมโยงกับไดรเวอร์อุปกรณ์และอะแด็ปเตอร์สำหรับอินเตอร์เฟส ให้ใช้ชื่ออะแด็ปเตอร์ ไม่ใช่ชื่ออินเตอร์เฟส ใช้ SMIT หรือคำสั่ง **chdev** เพื่อเปลี่ยนขนาดคิว

- **S/W Transmit Queue Overflow**

  จำนวนของแพ็กเก็ตขาออกที่โอเวอร์โฟลว์คิวการส่งผ่านของซอฟต์แวร์ ค่าที่ไม่ใช่ศูนย์ต้องการการดำเนินการเช่นเดียวกับถ้า `Max Packets on S/W Transmit Queue` มากถึง *xmt_que_size* ต้องเพิ่มขนาดคิว การส่งผ่าน

- **Broadcast Packets**

  จำนวนของ broadcast packets ที่ได้รับโดยไม่มี ข้อผิดพลาดใดๆ

  ถ้าค่าของ broadcast packets สูง ให้เปรียบเทียบกับ แพ็กเก็ตทั้งหมดที่ได้รับ Broadcast packets ที่ได้รับควรจะ น้อยกว่า 20 เปอร์เซ็นต์ของแพ็กเก็ตทั้งหมดที่ได้รับ ถ้าค่านี้สูง อาจบ่งชี้ว่า โหลดเครือข่ายสูง การใช้ multicasting การใช้ IP multicasting ช่วยให้สามารถส่งผ่านข้อความไปยังกลุ่มของโฮสต์ แทนที่จะต้องระบุที่อยู่ และส่งข้อความไปยังสมาชิกของกลุ่มทีละราย

- **DMA Overrun**

  สถิติ `DMA Overrun` เพิ่มขึ้น เมื่ออะแด็ปเตอร์กำลังใช้ DMA เพื่อวางแพ็กเก็ตเข้าในหน่วยความจำระบบและ การโอนย้ายยังไม่สมบูรณ์ มีบัฟเฟอร์ระบบสำหรับใช้วางแพ็กเก็ตได้ แต่การดำเนินงาน DMA ล้มเหลวในการทำเช่นนั้น กรณีนี้เกิดขึ้นเมื่อ MCA บัสยุ่งเกินไปจนอะแด็ปเตอร์ไม่สามารถใช้ DMA สำหรับแพ็กเก็ตได้ ที่ตั้งของอะแด็ปเตอร์บนบัสเป็นสิ่งที่สำคัญมากในระบบที่มีโหลดมาก โดยปกติ อะแด็ปเตอร์ในหมายเลขสล็อตที่ต่ำกว่าบนบัส ซึ่งมีระดับความสำคัญบัส สูงกว่า ใช้บัสมากจนอะแด็ปเตอร์ในหมายเลขสล็อตที่สูงกว่า ไม่ได้รับการบริการ โดยเฉพาะอย่างยิ่งถ้าอะแด็ปเตอร์ในหมายเลขสล็อตที่ต่ำกว่า เป็นอะแด็ปเตอร์ ATM

- **Max Collision Errors**

  จำนวนของการส่งผ่านที่ไม่สำเร็จเนื่องจากมี การปะทะมากเกินไป จำนวนการปะทะที่พบเกินกว่า จำนวนการลองใหม่บนอะแด็ปเตอร์

- **Late Collision Errors**

  จำนวนของการส่งผ่านที่ไม่สำเร็จเนื่องจาก ข้อผิดพลาดการปะทะในภายหลัง

- **Timeout Errors**

  จำนวนของการส่งผ่านที่ไม่สำเร็จเนื่องจาก อะแด็ปเตอร์รายงานข้อผิดพลาดไทม์เอาต์

- **Single Collision Count**

  จำนวนของแพ็กเก็ตขาออกที่พบการปะทะครั้งเดียว (ครั้งเดียวเท่านั้น) ในระหว่างการส่งผ่าน

- **Multiple Collision Count**

  จำนวนของแพ็กเก็ตขาออกที่พบการปะทะหลายครั้ง (2 – 15 ครั้ง) ในระหว่างการส่งผ่าน

- **Receive Collision Errors**

  จำนวนของแพ็กเก็ตขาเข้าที่มีข้อผิดพลาดการปะทะ ในระหว่างการรับ

- **No mbuf Errors**

  จำนวนครั้งที่ mbufs ไม่มีอยู่ ที่ไดรเวอร์อุปกรณ์โดยปกติแล้ว กรณีนี้เกิดขึ้นในระหว่างการดำเนินงานรับ เมื่อไดรเวอร์ ต้องได้รับบัฟเฟอร์หน่วยความจำเพื่อประมวลผลแพ็กเก็ตขาเข้า ถ้าพูล mbuf สำหรับขนาดที่ร้องขอว่าง แพ็กเก็ตจะถูกทิ้งไป ใช้คำสั่ง **netstat -m** เพื่อยืนยัน และเพิ่มพารามิเตอร์ *thewall*

  ค่า `No mbuf Errors` เป็นค่าเฉพาะอินเตอร์เฟสและไม่เหมือนกันกับ `requests for mbufs denied` จากเอาต์พุต **netstat -m** เปรียบเทียบค่าของตัวอย่างสำหรับคำสั่ง **netstat -m** และ **netstat -v** (ส่วนอีเทอร์เน็ต และ Token-Ring)

เมื่อต้องการทราบปัญหาประสิทธิภาพเครือข่าย ให้ตรวจสอบจำนวน `Error` ในเอาต์พุต **netstat -v**

คำแนะนำเพิ่มเติม:

- เมื่อต้องการตรวจสอบเครือข่ายอีเทอร์เน็ตที่โอเวอร์โหลด ให้คำนวณ (จากคำสั่ง **netstat -v**):

  ```
  (Max Collision Errors + Timeouts Errors) / Transmit Packets
  ```

  ถ้า ผลลัพธ์มากกว่า 5 เปอร์เซ็นต์ ให้จัดระเบียบเครือข่ายใหม่เพื่อทำให้โหลด สมดุล
- การบ่งชี้อีกอย่างหนึ่งสำหรับโหลดเครือข่ายสูงคือ (จากคำสั่ง **netstat -v**):
  ถ้าจำนวนการปะทะทั้งหมดจากเอาต์พุต **netstat -v** (สำหรับอีเทอร์เน็ต) มากกว่า 10 เปอร์เซ็นต์ของแพ็กเก็ตทั้งหมดที่ส่งผ่าน เป็นดังนี้:

  ```
  Number of collisions / Number of Transmit Packets > 0.1
  ```

โปรโตคอล netstat -p:

โปรโตคอล **netstat -p** แสดง สถิติเกี่ยวกับค่าที่ระบุสำหรับตัวแปรโปรโตคอล (`udp`, `tcp`, `sctp`,`ip`, `icmp`) ซึ่งเป็นชื่อที่รู้จักแพร่หลายสำหรับโปรโตคอล หรือนามแฝงสำหรับโปรโตคอล

ชื่อและชื่อย่อของโปรโตคอลบางส่วนมีการแสดงรายการอยู่ในไฟล์ `/etc/protocols` การตอบกลับ null บ่งชี้ว่าไม่มีตัวเลขที่จะรายงาน ถ้าไม่มีรูทีนสถิติ สำหรับการตอบกลับนั้น โปรแกรมจะรายงานค่าที่ระบุสำหรับ ตัวแปรโปรโตคอลเป็น ไม่ทราบ

ตัวอย่างต่อไปนี้แสดงเอาต์พุตสำหรับโปรโตคอล `ip` ดังนี้:

```
# netstat -p ip
ip:
        45775 total packets received
        0 bad header checksums
        0 with size smaller than minimum
        0 with data size < data length
        0 with header length < data size
        0 with data length < header length
        0 with bad options
        0 with incorrect version number
        0 fragments received
        0 fragments dropped (dup or out of space)
        0 fragments dropped after timeout
        0 packets reassembled ok
        45721 packets for this host
        51 packets for unknown/unsupported protocol
        0 packets forwarded
        4 packets not forwardable
        0 redirects sent
        33877 packets sent from this host
        0 packets sent with fabricated ip header
        0 output packets dropped due to no bufs, etc.
        0 output packets discarded due to no route
        0 output datagrams fragmented
        0 fragments created
        0 datagrams that can't be fragmented
        0 IP Multicast packets dropped due to no receiver
        0 successful path MTU discovery cycles
        1 path MTU rediscovery cycle attempted
        3 path MTU discovery no-response estimates
        3 path MTU discovery response timeouts
        1 path MTU discovery decrease detected
        8 path MTU discovery packets sent
```

```
        0 path MTU discovery memory allocation failures
        0 ipintrq overflows
        0 with illegal source
        0 packets processed by threads
        0 packets dropped by threads
        0 packets dropped due to the full socket receive buffer
        0 dead gateway detection packets sent
        0 dead gateway detection packet allocation failures
        0 dead gateway detection gateway allocation failures
```

ฟิลด์ที่ไฮไลต์มีการอธิบายดังนี้:

- **แพ็กเก็ตทั้งหมดที่ได้รับ**

  จำนวนของ IP datagrams ทั้งหมดที่ได้รับ
- **Checksum ส่วนหัวไม่ดีหรือทิ้งเฟรกเมนต์**

  ถ้าเอาต์พุตแสดง `bad header checksum` หรือ `fragments dropped` เนื่องจาก `dup or out of space` นี่บ่งชี้ว่าเครือข่ายมีแพ็กเก็ตที่เสียหายหรือ ไดรเวอร์อุปกรณ์ได้รับคิวที่ใหญ่ไม่พอ
- **เฟรกเมนต์ที่ได้รับ**

  จำนวนของเฟรกเมนต์ทั้งหมดที่ได้รับ
- **ทิ้งหลังจากไทม์เอาต์**

  ถ้า `fragments dropped after timeout` เป็นค่า อื่นที่ไม่ใช่ศูนย์ ผลคือ `time to life counter` ของเฟรกเมนต์ `ip` จะหมดอายุเนื่องจากเครือข่ายยุ่งก่อนที่จะได้รับเฟรกเมนต์ทั้งหมดของ datagram เพื่อหลีกเลี่ยงปัญหานี้ ให้ใช้คำสั่ง **no** เพื่อเพิ่มค่าของพารามิเตอร์ เครือข่าย *ipfragttl* สาเหตุอีกอย่างหนึ่งอาจเป็น การขาด mbufs; เพิ่ม *thewall*
- **แพ็กเก็ตที่ส่งจากโฮสต์นี้**

  จำนวนของ IP datagrams ที่สร้าง และส่งออกจากระบบนี้ จำนวนนับนี้ไม่รวม datagrams ที่ส่งต่อ (passthrough traffic)
- **เฟรกเมนต์ที่สร้างขึ้น**

  จำนวนเฟรกเมนต์ที่สร้างขึ้นในระบบนี้ เมื่อส่ง IP datagrams

เมื่อดูสถิติ IP ให้ดูอัตราส่วนของ `packets received` ต่อ `fragments received` ตามคำแนะนำสำหรับเครือข่าย MTU ขนาดเล็ก ถ้า 10 เปอร์เซ็นต์หรือ มากกว่าของแพ็กเก็ตมีการแบ่งเฟรกเมนต์ คุณควรสืบสวนหาสาเหตุ เพิ่มเติม เฟรกเมนต์จำนวนมากบ่งชี้ว่าโปรโตคอลบนชั้น IP บนรีโมตโฮสต์กำลังส่งผ่านข้อมูลไปยัง IP ด้วยขนาดข้อมูลที่ใหญ่กว่า MTU ของอินเตอร์เฟส Gateways/เราเตอร์ในพาธเครือข่ายยังอาจมีขนาด MTU ที่เล็กกว่าโหนดอื่นในเครือข่ายมากด้วย ล็อกอินเดียวกัน สามารถใช้กับ `packets sent` และ `fragments created`

เนื่องจากการแบ่งเฟรกเมนต์ส่งผลให้โอเวอร์เฮด CPU เพิ่มขึ้น ดังนั้นการหา สาเหตุจึงเป็นสิ่งที่สำคัญ ระวังว่าลักษณะของแอ็พพลิเคชันบางรายการอาจทำให้เกิด การแบ่งเฟรกเมนต์ได้ ตัวอย่างเช่น แอ็พพลิเคชันที่ส่งข้อมูลจำนวนเล็กน้อย อาจทำให้เกิดการแบ่งเฟรกเมนต์ได้ อย่างไรก็ตาม ถ้าคุณทราบว่าแอ็พพลิเคชัน กำลังส่งข้อมูลจำนวนมากและการแบ่งเฟรกเมนต์ยังคงเกิดขึ้นอยู่ให้หา สาเหตุ อาจเป็นไปได้ว่าขนาด MTU ที่ใช้ไม่ใช่ขนาด MTU ที่ตั้งค่าคอนฟิก บนระบบ

ตัวอย่างต่อไปนี้แสดงเอาต์พุตสำหรับโปรโตคอล `udp` ดังนี้:

```
# netstat -p udp
udp:
        11623 datagrams received
        0 incomplete headers
        0 bad data length fields
```

```
        0 bad checksums
        620 dropped due to no socket
        10989 broadcast/multicast datagrams dropped due to no socket
        0 socket buffer overflows
        14 delivered
        12 datagrams output
```

สถิติที่น่าสนใจมีดังนี้:

- **Bad Checksums**

  Bad checksums อาจเกิดขึ้นได้เนื่องจากความล้มเหลวของ ฮาร์ดแวร์การ์ดหรือสายเคเบิล

- **Dropped Due to No Socket**

  จำนวนของ UDP datagrams ที่ได้รับ ซึ่งมีพอร์ตซ็อกเก็ตปลายทางที่ไม่ได้เปิดอยู่ ส่งผลให้มีการส่งข้อความ `ICMP Destination Unreachable - Port Unreachable` แต่ถ้า UDP datagrams ที่ได้รับเป็น broadcast datagrams ข้อผิดพลาด ICMP จะไม่ถูกสร้างขึ้น ถ้าค่านี้สูง ให้สืบสวนว่าแอ็พพลิเคชันจัดการซ็อกเก็ตอย่างไร

- **ซ็อกเก็ตบัฟเฟอร์โอเวอร์โฟลว์**

  ซ็อกเก็ตบัฟเฟอร์โอเวอร์โฟลว์อาจเกิดจาก UDP ซ็อกเก็ตการส่งผ่านและการรับไม่เพียงพอ มี **nfsd** daemons น้อยเกินไป หรือค่า *nfs_socketsize*, *udp_recvspace* และ *sb_max* น้อยเกินไป

ถ้าคำสั่ง **netstat -p udp** บ่งชี้ว่าซ็อกเก็ตโอเวอร์โฟลว์ คุณอาจต้องเพิ่มจำนวนของ **nfsd** daemons บนเซิร์ฟเวอร์ อันดับแรก ตรวจสอบระบบที่ได้รับผลกระทบว่า CPU หรือ I/O ใกล้เต็มหรือไม่ และตรวจสอบค่าติดตั้งที่แนะนำสำหรับชั้นการสื่อสารอื่น โดยใช้คำสั่ง **no -a** ถ้าระบบใกล้เต็มความสามารถ คุณต้องลดโหลด หรือเพิ่มรีซอร์สของระบบนั้น อย่างใดอย่างหนึ่ง

ตัวอย่างต่อไปนี้แสดงเอาต์พุตสำหรับโปรโตคอล `tcp` ดังนี้:

```
 # netstat -p tcp
tcp:
        576 packets sent
                512 data packets (62323 bytes)
                0 data packets (0 bytes) retransmitted
                55 ack-only packets (28 delayed)
                0 URG only packets
                0 window probe packets
                0 window update packets
                9 control packets
                0 large sends
                0 bytes sent using largesend
                0 bytes is the biggest largesend
        719 packets received
                504 acks (for 62334 bytes)
                19 duplicate acks
                0 acks for unsent data
                449 packets (4291 bytes) received in-sequence
                8 completely duplicate packets (8 bytes)
                0 old duplicate packets
                0 packets with some dup. data (0 bytes duped)
                5 out-of-order packets (0 bytes)
                0 packets (0 bytes) of data after window
                0 window probes
                2 window update packets
                0 packets received after close
```

```
        0 packets with bad hardware assisted checksum
        0 discarded for bad checksums
        0 discarded for bad header offset fields
        0 discarded because packet too short
        0 discarded by listeners
        0 discarded due to listener's queue full
        71 ack packet headers correctly predicted
        172 data packet headers correctly predicted
6 connection requests
8 connection accepts
14 connections established (including accepts)
6 connections closed (including 0 drops)
0 connections with ECN capability
0 times responded to ECN
0 embryonic connections dropped
504 segments updated rtt (of 505 attempts)
0 segments with congestion window reduced bit set
0 segments with congestion experienced bit set
0 resends due to path MTU discovery
0 path MTU discovery terminations due to retransmits
0 retransmit timeouts
        0 connections dropped by rexmit timeout
0 fast retransmits
        0 when congestion window less than 4 segments
0 newreno retransmits
0 times avoided false fast retransmits
0 persist timeouts
        0 connections dropped due to persist timeout
16 keepalive timeouts
        16 keepalive probes sent
        0 connections dropped by keepalive
0 times SACK blocks array is extended
0 times SACK holes array is extended
0 packets dropped due to memory allocation failure
0 connections in timewait reused
0 delayed ACKs for SYN
0 delayed ACKs for FIN
0 send_and_disconnects
0 spliced connections
0 spliced connections closed
0 spliced connections reset
0 spliced connections timeout
0 spliced connections persist timeout
0 spliced connections keepalive timeout
```

สถิติที่น่าสนใจมีดังนี้:

- แพ็กเก็ตที่ส่ง
- แพ็กเก็ตข้อมูล
- แพ็กเก็ตข้อมูลที่ส่งผ่านใหม่
- แพ็กเก็ตที่ได้รับ
- แพ็กเก็ตซ้ำที่เสร็จสมบูรณ์แล้ว
- ไทม์เอาต์การส่งผ่านใหม่

สำหรับสถิติ TCP ให้เปรียบเทียบจำนวนของแพ็กเก็ตที่ส่งกับจำนวนของ แพ็กเก็ตข้อมูลที่ส่งผ่านใหม่ ถ้าจำนวนของแพ็กเก็ตที่ส่งผ่านใหม่มากกว่า 10-15 เปอร์เซ็นต์ของแพ็กเก็ตทั้งหมดที่ส่ง การที่ TCP เกิดไทม์เอาต์บ่งชี้ว่า การจราจรเครือข่ายอาจสูงเกินไปสำหรับการส่งคืน acknowledgments (ACKs) ก่อนที่จะไทม์เอาต์ ปัญหาคอขวดบนโหนดการรับหรือปัญหาเครือข่ายทั่วไปยังอาจทำให้เกิด การส่งผ่านใหม่ของ TCP ได้เช่นกัน ซึ่งจะทำให้การจราจรเครือข่ายมากขึ้น และ ส่งผลให้เกิดปัญหาประสิทธิภาพของเครือข่าย

นอกจากนี้ให้เปรียบเทียบจำนวนของแพ็กเก็ตที่ได้รับกับจำนวนของ แพ็กเก็ตซ้ำที่เสร็จสมบูรณ์แล้ว ถ้า TCP บนโหนดการส่งไทม์เอาต์ก่อนที่จะได้รับ ACK จาก โหนดการรับ TCP จะส่งผ่านแพ็กเก็ตใหม่ แพ็กเก็ตซ้ำเกิดขึ้น เมื่อโหนดการรับได้รับแพ็กเก็ตที่ส่งผ่านใหม่ทั้งหมดในท้ายที่สุด ถ้าจำนวนของแพ็กเก็ตซ้ำเกินกว่า 10-15 เปอร์เซ็นต์ ปัญหาอาจเป็นเช่นเดิม คือมีการจราจรเครือข่ายมากเกินไปหรือปัญหาคอขวดที่โหนดการรับ แพ็กเก็ตทำให้ การจราจรเครือข่ายเพิ่มขึ้น

ค่าสำหรับไทม์เอาต์การส่งผ่านใหม่เกิดขึ้นเมื่อ TCP ส่งแพ็กเก็ต แต่ไม่ได้รับ ACK ทันเวลา ดังนั้นจึงส่งแพ็กเก็ตอีกครั้ง ค่านี้เพิ่มขึ้น สำหรับการส่งผ่านใหม่ในเวลาต่อมาใดๆ การส่งผ่านอย่างต่อเนื่องเช่นนี้ทำให้การใช้ประโยชน์ CPU สูงขึ้น และถ้าโหนดการรับไม่ได้รับแพ็กเก็ต แพ็กเก็ตจะถูกทิ้งในท้ายที่สุด

netstat -s:

คำสั่ง **netstat -s** แสดงสถิติเกี่ยวกับแต่ละโปรโตคอล (ในขณะที่คำสั่ง **netstat -p** แสดง สถิติเกี่ยวกับโปรโตคอลที่ระบุ)

คำสั่ง **netstat -s** แสดงสถิติเกี่ยวกับ โปรโตคอลต่อไปนี้เท่านั้น:

- TCP
- UDP
- SCTP
- IP
- IPv6
- IGMP
- ICMP
- ICMPv6

netstat -s -s:

อ็อพชัน **-s -s** ที่ไม่มีในส่วนเอกสารแสดงเฉพาะบรรทัดของเอาต์พุต **netstat -s** ที่ไม่ใช่ศูนย์ ซึ่งทำให้ค้นหาจำนวนข้อผิดพลาดได้ง่ายขึ้น

netstat -s -Z:

คำสั่ง **netstat** ล้างข้อมูล ตัวนับสถิติทั้งหมดของคำสั่ง **netstat -s** เป็นศูนย์

netstat -r:

อีกอ็อพชันหนึ่งที่เกี่ยวข้องกับประสิทธิภาพคือการแสดง Path Maximum Transmission Unit (PMTU) ที่ตรวจพบ ใช้คำสั่ง **netstat -r** เพื่อแสดงค่านี้

สำหรับโฮสต์สองรายการที่สื่อสารระหว่างกันบนพาธที่ประกอบด้วยหลายเครือข่าย แพ็กเก็ตที่ส่งผ่านจะมีการแบ่งเฟรกเมนต์ถ้าขนาดของแพ็กเก็ตนั้นใหญ่กว่า MTU ที่เล็กที่สุดของเครือข่ายใดๆ ในพาธ เนื่องจากการแบ่งเฟรกเมนต์แพ็กเก็ตอาจส่งผลให้ประสิทธิภาพของ เครือข่ายลดลง จึงควรหลีกเลี่ยงการแบ่งเฟรกเมนต์โดยการส่งผ่านแพ็กเก็ต ที่มีขนาดไม่ใหญ่เกินกว่า MTU ที่เล็กที่สุดในพาธเครือข่าย ขนาด นี้เรียกว่า *พาธ MTU*

ข้อมูลต่อไปนี้เป็นตัวอย่างของคำสั่ง **netstat −r −f inet** ที่ใช้เพื่อแสดงตารางการเราต์เท่านั้น:

```
# netstat -r -f inet
Routing tables
Destination      Gateway           Flags   Refs     Use  If   PMTU Exp Groups

Route tree for Protocol Family 2 (Internet):
default           res101141        UGc       0        0  en1     -   -
ausdns01.srv.ibm  res101141        UGHW      1        4  en1  1500   -
10.1.14.0         server1          UHSb      0        0  en1     -   -  =>
10.1.14/24        server1          U         3      112  en1     -   -
brown17          loopback          UGHS      6      110  lo0     -   -
10.1.14.255       server1          UHSb      0        0  en1     -   -
magenta          res1031041        UGHW      1       42  en1     -   -
127/8            loopback          U         6    16633  lo0     -   -
192.1.6.0        en6host1          UHSb      0        0  en0     -   -  =>
192.1.6/24       en6host1          U         0       17  en0     -   -
en6host1         loopback          UGHS      0    16600  lo0     -   -
192.1.6.255      en6host1          UHSb      0        0  en0     -   -
192.6.0.0        fc0host1          UHSb      0        0  fc0     -   -  =>
192.6.0/24       fc0host1          U         0       20  fc0     -   -
fc0host1         loopback          UGHS      0        0  lo0     -   -
192.6.0.255      fc0host1          UHSb      0        0  fc0     -   -
```

netstat −D:

อ็อพชัน **−D** ช่วยให้คุณเห็นแพ็กเก็ตที่กำลังเข้ามาและ กำลังออกจากแต่ละชั้นในระบบย่อยการสื่อสาร ควบคู่ไปกับแพ็กเก็ตที่ทิ้งในแต่ละชั้น

```
# netstat -D

Source                          Ipkts               Opkts     Idrops     Odrops
-------------------------------------------------------------------------------
ent_dev1                        32556                 727          0          0
ent_dev2                            0                   1          0          0
ent_dev3                            0                   1          0          0
fcnet_dev0                         24                  22          0          0
fcnet_dev1                          0                   0          0          0
ent_dev0                           14                  15          0          0
              ---------------------------------------------------------------
Devices Total                   32594                 766          0          0
-------------------------------------------------------------------------------
ent_dd1                         32556                 727          0          0
ent_dd2                             0                   2          0          1
ent_dd3                             0                   2          0          1
fcnet_dd0                          24                  22          0          0
fcnet_dd1                           0                   0          0          0
ent_dd0                            14                  15          0          0
              ---------------------------------------------------------------
```

```
Drivers Total                 32594                768          0          2
-------------------------------------------------------------------------------
fcs_dmx0                          0                N/A          0        N/A
fcs_dmx1                          0                N/A          0        N/A
ent_dmx1                      31421                N/A       1149        N/A
ent_dmx2                          0                N/A          0        N/A
ent_dmx3                          0                N/A          0        N/A
fcnet_dmx0                        0                N/A          0        N/A
fcnet_dmx1                        0                N/A          0        N/A
ent_dmx0                         14                N/A          0        N/A
              ---------------------------------------------------------------
Demuxer Total                 31435                N/A       1149        N/A
-------------------------------------------------------------------------------
IP                            46815              34058         64          8
IPv6                              0                  0          0          0
TCP                             862                710          9          0
UDP                           12412                 13      12396          0
              ---------------------------------------------------------------
Protocols Total               60089              34781      12469          8
-------------------------------------------------------------------------------
en_if1                        31421                732          0          0
fc_if0                           24                 22          0          0
en_if0                           14                 20          0          6
lo_if0                        33341              33345          4          0
              ---------------------------------------------------------------
Net IF Total                  64800              34119          4          6
-------------------------------------------------------------------------------
(Note:  N/A -> Not Applicable)
```

ชั้นอุปกรณ์แสดงจำนวนของแพ็กเก็ตที่กำลังเข้าไปในอะแด็ปเตอร์ กำลังออกจาก อะแด็ปเตอร์ และจำนวนของแพ็กเก็ตที่ทิ้งบนอินพุตและเอาต์พุต ข้อผิดพลาด อะแด็ปเตอร์มีหลายสาเหตุ และคำสั่ง **netstat -v** สามารถตรวจสอบรายละเอียดเพิ่มเติมได้

ชั้นไดรเวอร์แสดงจำนวนแพ็กเก็ตที่จัดการโดยไดรเวอร์อุปกรณ์สำหรับ แต่ละอะแด็ปเตอร์ เอาต์พุตของคำสั่ง **netstat -v** มีประโยชน์ในการกำหนดข้อผิดพลาดที่จะนับที่นี่

ค่า Demuxer แสดงจำนวนแพ็กเก็ตที่ชั้น demux และโดยปกติแล้ว `Idrops` ที่นี่บ่งชี้ว่าการกรองเป็นสาเหตุให้แพ็กเก็ตถูกปฏิเสธ (ตัวอย่างเช่น Netware หรือ DecNet แพ็กเก็ตถูกปฏิเสธเนื่องจากไม่ได้มีการจัดการโดย ระบบที่มีการตรวจสอบ)

รายละเอียดของชั้นโปรโตคอลสามารถดูได้ในเอาต์พุตของคำสั่ง **netstat -s**

หมายเหตุ: ใน เอาต์พุตสถิติ `N/A` ที่แสดงขึ้นในค่าฟิลด์บ่งชี้ว่า ไม่สามารถนับได้ สำหรับสถิติ NFS/RPC จำนวนของแพ็กเก็ตขาเข้าที่ ส่งผ่าน RPC คือจำนวนแพ็กเก็ตเดียวกันกับที่ส่งผ่าน NFS ดังนั้นจึงไม่มีการรวมจำนวนเหล่านี้ในฟิลด์ `NFS/RPC Total` ส่งผลให้เป็น `N/A` NFS ไม่มีแพ็กเก็ตขาออกหรือตัวนับการทิ้งแพ็กเก็ตขาออกเฉพาะ NFS และ RPC ด้วยเหตุนี้ จำนวนของแต่ละรายการจึงมีค่าฟิลด์เป็น `N/A` และจำนวนรวมมีการจัดเก็บไว้ในฟิลด์ `NFS/RPC Total`

คำสั่ง netpmon

คำสั่ง **netpmon** ใช้ฟังก์ชันการติดตาม เพื่อให้ได้รูปภาพโดยละเอียดของกิจกรรมเครือข่ายในระหว่างช่วงเวลาหนึ่ง เนื่องจากใช้ฟังก์ชันการติดตาม คำสั่ง **netpmon** จึงสามารถรันได้โดยผู้ใช้รากหรือโดยสมาชิกของกลุ่มระบบเท่านั้น

คำสั่ง **netpmon** ไม่สามารถรันพร้อมกับคำสั่งประสิทธิภาพที่ขึ้นอยู่กับการติดตามอื่นๆ เช่น **tprof** และ **filemon** ในโหมดปกติ คำสั่ง **netpmon** จะรันในพื้นหลัง ในขณะที่กำลังดำเนินการและมอนิเตอร์แอ็พพลิเคชันโปรแกรมหรือคำสั่งระบบ หนึ่งรายการขึ้นไป

คำสั่ง **netpmon** เน้นกิจกรรมของระบบ ดังต่อไปนี้:

- การใช้ CPU
  - ตามกระบวนการและตัวจัดการขัดจังหวะ
  - เกี่ยวข้องกับเครือข่ายมากเพียงใด
  - อะไรเป็นสาเหตุของเวลา idle
- I/O ไดรเวอร์อุปกรณ์เครือข่าย
  - มอนิเตอร์การดำเนินการ I/O ผ่านไดร์เวอร์อุปกรณ์เครือข่าย Ethernet, Token-Ring และ Fibre-Distributed Data Interface (FDDI) ทั้งหมด
  - ในกรณีของ I/O การส่งผ่าน คำสั่งจะมอนิเตอร์การใช้ประโยชน์ ความยาว คิว และโฮสต์ปลายทาง สำหรับ ID การรับ คำสั่งยังมอนิเตอร์ เวลาในชั้น demux ด้วย
- การเรียกอินเทอร์เน็ตซ็อกเก็ต
  - มอนิเตอร์รูทีนย่อย **send()**, **recv()**, **sendto()**, **recvfrom()**, **sendmsg()**, **read()**, และ **write()** บนอินเทอร์เน็ตซ็อกเก็ต
  - รายงานสถิติสำหรับแต่ละกระบวนการเกี่ยวกับ Internet Control Message Protocol (ICMP), Transmission Control Protocol (TCP), และ User Datagram Protocol (UDP)
- NFS I/O
  - บนไคลเอ็นต์: การร้องขอ RPC, การร้องขอการอ่าน/การบันทึก NFS
  - บนเซิร์ฟเวอร์: การร้องขอการอ่าน/การบันทึกสำหรับแต่ละไคลเอ็นต์ และสำหรับแต่ละไฟล์

จะมีการคำนวณข้อมูลต่อไปนี้:

- เวลาการตอบสนองและขนาดที่เชื่อมโยงกับการดำเนินงานส่งผ่านและได้รับ ที่ระดับไดรเวอร์อุปกรณ์
- เวลาการตอบสนองและขนาดที่เชื่อมโยงกับการเรียกระบบการอ่านและการบันทึก ของอินเทอร์เน็ตซ็อกเก็ตทุกชนิด
- เวลาการตอบสนองและขนาดที่เชื่อมโยงกับการเรียกระบบการอ่านและการบันทึก ของ NFS
- เวลาการตอบสนองที่เชื่อมโยงกับการร้องขอการเรียก NFS รีโมตโพรซีเดอร์

เมื่อต้องการกำหนดว่าคำสั่ง **netpmon** มีการติดตั้งและมีอยู่ หรือไม่ ให้รันคำสั่งต่อไปนี้:

```
# lslpp -lI perfagent.tools
```

การติดตามเริ่มต้นขึ้นโดยคำสั่ง **netpmon** และสามารถเลือกให้พักไว้ ได้โดยใช้คำสั่งย่อย **trcoff** และคืนสภาพโดยใช้คำสั่งย่อย **trcon** ในทันทีที่การติดตามสิ้นสุดลง คำสั่ง **netpmon** จะบันทึก รายงานการติดตามลงในเอาต์พุตมาตรฐาน

การใช้คำสั่ง netpmon:

คำสั่ง **netpmon** จะเริ่มต้นการติดตามในทันที ยกเว้นว่ามีการใช้อ็อพชัน **-d**

ใช้คำสั่ง **trcstop** เพื่อหยุดการติดตาม ณ เวลานั้น มีการสร้างรายงานที่ระบุทั้งหมด และมีคำสั่ง **netpmon** อยู่ในสภาพแวดล้อมไคลเอ็นต์เซิร์ฟเวอร์ ใช้คำสั่ง **netpmon** เพื่อดูว่าเครือข่ายมีผลกระทบต่อประสิทธิภาพโดยรวมอย่างไร คำสั่งนี้สามารถรันได้ทั้งบน ไคลเอ็นต์และเซิร์ฟเวอร์

คำสั่ง **netpmon** สามารถอ่านข้อมูลการติดตาม I/O จากไฟล์ที่ระบุ แทนการอ่านจากกระบวนการติดตามแบบ เรียลไทม์ ในกรณีนี้ รายงาน **netpmon** สรุปกิจกรรมเครือข่าย สำหรับระบบและรอบเวลาที่แสดงโดยไฟล์การติดตาม วิธีการกระบวนการออฟไลน์นี้มีประโยชน์เมื่อจำเป็นต้องประมวลผล ไฟล์การติดตามจากเครื่องรีโมตในภายหลัง หรือรวบรวมข้อมูลการติดตาม ณ เวลา หนึ่งและประมวลผลภายหลังในอีกเวลาหนึ่ง

คำสั่ง **trcrpt -r** ต้อง ดำเนินการบนไฟล์บันทึกการติดตามและกำหนดทิศทางใหม่ไปยังอีกไฟล์หนึ่ง ดังนี้:

```
# gennames > gennames.out
# trcrpt -r trace.out > trace.rpt
```

ณ เวลานี้ ไฟล์บันทึกการติดตามที่ปรับ ถูกป้อนเข้าในคำสั่ง **netpmon** เพื่อรายงาน กิจกรรม I/O ที่ตรวจจับโดยเซสชันการติดตามที่บันทึกไว้ก่อนหน้านี้ ดังนี้:

```
# netpmon -i trace.rpt -n gennames.out | pg
```

ใน ตัวอย่างนี้ คำสั่ง **netpmon** อ่านเหตุการณ์การติดตามระบบไฟล์ จากไฟล์อินพุต `trace.rpt` เนื่องจากข้อมูลการติดตามมีการตรวจจับบนไฟล์อยู่แล้ว คำสั่ง **netpmon** จึงไม่ต้อง ไปอยู่ในพื้นหลังเพื่อให้แอ็พพลิเคชันโปรแกรมรัน ได้ หลังจากอ่านทั้งไฟล์แล้ว รายงานกิจกรรมเครือข่ายจะแสดงขึ้นบน เอาต์พุตมาตรฐาน (ซึ่งในตัวอย่างนี้ ไปป์กับคำสั่ง **pg**)

ถ้ารันคำสั่ง **trace** ที่มีแฟล็ก **-C all** ให้รันคำสั่ง **trcrpt** ที่มีแฟล็ก **-C all** ด้วย (โปรดดู “การจัดรูปแบบรายงานจากการติดตามเอาต์พุต -C” ในหน้า 439)

คำสั่ง **netpmon** ต่อไปนี้ที่กำลัง รันบนเซิร์ฟเวอร์ NFS จะดำเนินการคำสั่ง **sleep** และสร้างรายงาน หลังผ่านไป 400 วินาที ในระหว่างช่วงเวลาที่ประเมิน มีการคัดลอกไปยัง ระบบไฟล์ที่ติดตั้ง NFS `/nfs_mnt` เกิดขึ้น

```
# netpmon -o netpmon.out -O all; sleep 400; trcstop
```

ด้วยอ็อพชัน **-O** คุณสามารถระบุชนิดรายงานที่สร้าง ได้ ค่าชนิดรายงานที่ถูกต้องมีดังนี้:

`cpu` การใช้ CPU

`dd` I/O ไดรเวอร์อุปกรณ์เครือข่าย

`so` I/O การเรียกอินเทอร์เน็ตซ็อกเก็ต

`nfs` NFS I/O

`all` สร้างรายงานทุกชนิด ข้อมูลต่อไปนี้เป็นค่าดีฟอลต์

```
# cat netpmon.out

Fri Mar  5 15:41:52 2004
System: AIX crusade Node: 5 Machine: 000353534C00


========================================================================

Process CPU Usage Statistics:
-----------------------------
```

```
                                                        Network
Process (top 20)             PID  CPU Time   CPU %   CPU %
------------------------------------------------------------
netpmon                    45600    0.6995   1.023   0.000
nfsd                       50090    0.5743   0.840   0.840
UNKNOWN                    56912    0.1274   0.186   0.000
trcstop                    28716    0.0048   0.007   0.000
gil                         3870    0.0027   0.004   0.004
ksh                        42186    0.0024   0.003   0.000
IBM.ServiceRMd             14966    0.0021   0.003   0.000
IBM.ERrmd                   6610    0.0020   0.003   0.000
IBM.CSMAgentRMd            15222    0.0020   0.003   0.000
IBM.AuditRMd               12276    0.0020   0.003   0.000
syncd                       4766    0.0020   0.003   0.000
sleep                      28714    0.0017   0.002   0.000
swapper                        0    0.0012   0.002   0.000
rpc.lockd                  34942    0.0007   0.001   0.000
netpmon                    28712    0.0006   0.001   0.000
trace                      54622    0.0005   0.001   0.000
reaper                      2580    0.0003   0.000   0.000
netm                        3612    0.0002   0.000   0.000
aixmibd                     4868    0.0001   0.000   0.000
xmgc                        3354    0.0001   0.000   0.000
------------------------------------------------------------
Total (all processes)               1.4267   2.087   0.844
Idle time                          55.4400  81.108

========================================================================

First Level Interrupt Handler CPU Usage Statistics:
---------------------------------------------------
                                                        Network
FLIH                               CPU Time   CPU %   CPU %
------------------------------------------------------------
external device                      0.3821   0.559   0.559
PPC decrementer                      0.0482   0.070   0.000
data page fault                      0.0137   0.020   0.000
queued interrupt                     0.0002   0.000   0.000
------------------------------------------------------------
Total (all FLIHs)                    0.4441   0.650   0.559

========================================================================

Second Level Interrupt Handler CPU Usage Statistics:
----------------------------------------------------
                                                        Network
SLIH                               CPU Time   CPU %   CPU %
------------------------------------------------------------
phxentdd32                           2.4740   3.619   3.619
------------------------------------------------------------
Total (all SLIHs)                    2.4740   3.619   3.619

========================================================================

Network Device-Driver Statistics (by Device):
```

```
-------------------------------------------
                      ----------- Xmit -----------   -------- Recv ---------
Device                 Pkts/s  Bytes/s  Util  QLen   Pkts/s  Bytes/s   Demux
------------------------------------------------------------------------------
ethernet 4            7237.33 10957295  0.0%27.303  3862.63   282624  0.2324

================================================================================

Network Device-Driver Transmit Statistics (by Destination Host):
----------------------------------------------------------------

Host                   Pkts/s  Bytes/s
---------------------------------------
client_machine         7237.33 10957295

================================================================================

NFS Server Statistics (by Client):
----------------------------------
                       ------ Read -----   ----- Write -----     Other
Client                 Calls/s   Bytes/s   Calls/s   Bytes/s   Calls/s
--------------------------------------------------------------------------
client_machine      0.00         0      0.00         0    321.15
--------------------------------------------------------------------------
Total (all clients)         0.00         0      0.00         0    321.15

================================================================================

Detailed Second Level Interrupt Handler CPU Usage Statistics:
-------------------------------------------------------------

SLIH: phxentdd32
count:                 33256
  cpu time (msec):     avg 0.074   min 0.018   max 288.374 sdev 1.581

COMBINED (All SLIHs)
count:                 33256
  cpu time (msec):     avg 0.074   min 0.018   max 288.374 sdev 1.581

================================================================================

Detailed Network Device-Driver Statistics:
------------------------------------------

DEVICE: ethernet 4
recv packets:          33003
  recv sizes (bytes):  avg 73.2    min 60      max 618     sdev 43.8
  recv times (msec):   avg 0.000   min 0.000   max 0.005   sdev 0.000
  demux times (msec):  avg 0.060   min 0.004   max 288.360 sdev 1.587
xmit packets:          61837
  xmit sizes (bytes):  avg 1514.0  min 1349    max 1514    sdev 0.7
  xmit times (msec):   avg 3.773   min 2.026   max 293.112 sdev 8.947

================================================================================
```

```
Detailed Network Device-Driver Transmit Statistics (by Host):
------------------------------------------------------------

HOST: client_machine (10.4.104.159)
xmit packets:          61837
  xmit sizes (bytes):  avg 1514.0  min 1349    max 1514    sdev 0.7
  xmit times (msec):   avg 3.773   min 2.026   max 293.112 sdev 8.947

======================================================================

Detailed NFS Server Statistics (by Client):
-------------------------------------------

CLIENT: client_machine
other calls:           2744
  other times (msec):  avg 0.192   min 0.075   max 0.311   sdev 0.025

COMBINED (All Clients)
other calls:           2744
  other times (msec):  avg 0.192   min 0.075   max 0.311   sdev 0.025
```

เอาต์พุตของคำสั่ง **netpmon** ประกอบขึ้นจากรายงานที่แตกต่างกันสองชนิดคือ: *สากล* และ *รายละเอียด* รายงานสากลแสดงรายการสถิติดังนี้:

- ตัวประมวลผลที่ใช้งานบ่อยที่สุด
- ตัวจัดการขัดจังหวะระดับแรก
- ตัวจัดการขัดจังหวะระดับที่สอง
- ไดรเวอร์อุปกรณ์เครือข่าย
- การส่งผ่านของไดรเวอร์อุปกรณ์เครือข่าย
- การเรียกซ็อกเก็ต TCP
- สถิติเซิร์ฟเวอร์หรือไคลเอ็นต์ NFS

รายงานสากลแสดงขึ้นที่ตอนต้นของเอาต์พุต **netpmon** และจัดทำขึ้นในระหว่างช่วงเวลาที่ประเมิน รายงานรายละเอียด นำเสนอข้อมูลเพิ่มเติมสำหรับรายงานสากล โดยค่าดีฟอลต์ รายงานมีข้อจำกัดใน การแสดงสถิติที่ประเมินซึ่งใช้งานมากที่สุด 20 รายการเท่านั้น ข้อมูลทั้งหมดในรายงาน มีการแสดงรายการจากด้านบนสุดไปด้านล่างสุดตามการใช้งานบ่อยที่สุดไปถึงน้อยที่สุด

รายงานโกลบอลของคำสั่ง netpmon:

รายงานที่สร้างขึ้นด้วยคำสั่ง **netpmon** จะขึ้นต้นด้วยส่วนหัว ซึ่งระบุวันที่ ID เครื่อง และระยะเวลาของการมอนิเตอร์ในหน่วยวินาที

ส่วนหัวจะตามด้วยชุดของโกลบอล และรายงานโดยละเอียด สำหรับชนิดรายงานที่ระบุไว้ทั้งหมด

สถิติการใช้ microprocessor:

แต่ละแถวอธิบายการใช้ microprocessor ที่เชื่อมโยงกับกระบวนการ

ยกเว้นว่ามีการระบุอ็อพชัน verbose (**-v**) รายการแสดง 20 กระบวนการ ที่ใช้งานมากที่สุดเท่านั้น ที่ด้านล่างของรายงาน มีการแสดงผลรวมของการใช้ microprocessor สำหรับกระบวนการทั้งหมด และรายงานเวลา idle ของ microprocessor จำนวนเปอร์เซ็นต์เวลา idle คำนวณจากเวลา idle หารด้วย ช่วงเวลาที่ประเมิน ความแตกต่างระหว่างผลรวมเวลา microprocessor และช่วงเวลาที่ประเมินเป็นผลมาจากตัวจัดการขัดจังหวะ

`Network CPU %` คือเปอร์เซ็นต์ของเวลาทั้งหมดที่กระบวนการนี้ใช้ในการดำเนินการโค้ดที่เกี่ยวข้องกับเครือข่าย

ถ้ามีการใช้แฟล็ก **-t** จะมีการแสดงสถิติการใช้เธรด microprocessor ด้วย แถวกระบวนการแต่ละแถวที่อธิบายข้างบนมีแถวตามหลังติดกันมา ซึ่งอธิบายการใช้ microprocessor ของแต่ละเธรดที่เป็นของกระบวนการนั้น ฟิลด์ต่างๆ ในแถวเหล่านี้เหมือนกับฟิลด์ของกระบวนการ ยกเว้น ฟิลด์ชื่อ เธรดไม่มีชื่อ

ในรายงานตัวอย่าง จำนวนเปอร์เซ็นต์ `Idle time` (81.104 เปอร์เซ็นต์) ที่แสดงในรายงานการใช้ microprocessor สากลคำนวณจาก `Idle time` (55.4400) หารด้วย `measured interval` คูณ 8 (8.54 วินาทีคูณ 8) เนื่องจากมี microprocessors แปดตัวใน เซิร์ฟเวอร์นี้ ถ้าคุณต้องการดูกิจกรรมของ microprocessor แต่ละตัว คุณสามารถใช้คำสั่ง **sar**, **ps**, หรือคำสั่งเฉพาะ SMP อื่น การคำนวณคล้ายกันนี้ใช้กับ `CPU %` ทั้งหมดที่ใช้โดยกระบวนการทั้งหมด `Idle time` เกิดขึ้นจาก I/O เครือข่าย ผลต่างระหว่าง `CPU Time` ทั้งหมด (55.4400 + 1.4267) และ `measured interval` เป็นผลมาจากตัวจัดการขัดจังหวะ และหลาย microprocessors ในรายงานตัวอย่างแสดงว่า การใช้ microprocessor ส่วนใหญ่เกี่ยวข้องกับเครือข่าย: (0.844 / 2.087) = 40.44 เปอร์เซ็นต์

หมายเหตุ: ถ้าผลลัพธ์ของ `CPU %` เครือข่ายทั้งหมดหารด้วย `CPU %` ทั้งหมดมากกว่า 0.5 จาก `Process CPU Usage Statistics` ของเซิร์ฟเวอร์ NFS แสดงว่าการใช้ microprocessor ส่วนใหญ่เกี่ยวข้องกับเครือข่าย

วิธีการนี้ยังเป็นวิธีที่ดีในการดูการใช้ microprocessor โดยกระบวนการ โดยไม่ต้องใช้เอาต์พุตของกระบวนการเฉพาะ

ข้อมูลสถิติการใช้ไมโครโพรเซสเซอร์ Interrupt Handler ระดับแรก:

แต่ละแถวอธิบายถึงการใช้ไมโครโพรเซสเซอร์ที่เชื่อมโยงกับ interrupt handler (FLIH) ในระดับแรก

ที่ส่วนท้ายของรายงาน การใช้ไมโครโพรเซสเซอร์สำหรับ FLIH ทั้งหมดจะถูกนำมารวมกัน

เวลา CPU
จำนวนเวลาทั้งหมดของไมโครโพรเซสเซอร์ที่ใช้โดย FLIH นี้

CPU %
การใช้ไมโครโพรเซสเซอร์สำหรับ interrupt handler นี้เป็นเปอร์เซ็นต์ของเวลาทั้งหมด

เน็ตเวิร์ก CPU %
เปอร์เซ็นต์เวลาทั้งหมดที่ interrupt handler นี้เรียกใช้งานในฐานะของเหตุการณ์ ที่เกี่ยวข้องกับเน็ตเวิร์ก

สถิติการใช้ Second Level Interrupt Handler microprocessor:

แต่ละแถวอธิบายการใช้ microprocessor ที่เชื่อมโยงกับ second-level interrupt handler (SLIH) ที่ด้านล่างของรายงาน มีการแสดงผลรวมของการใช้ microprocessor ของ SLIHs ทั้งหมด

สถิติไดรเวอร์อุปกรณ์เครือข่ายโดยเรียงตามอุปกรณ์:

คำสั่ง **netpmon** สามารถใช้เพื่อสร้างรายงานที่ แสดงรายการสถิติไดรเวอร์อุปกรณ์เครือข่ายโดยเรียงตามอุปกรณ์ได้

แต่ละแถวอธิบายสถิติที่เชื่อมโยงกับอุปกรณ์เครือข่าย

อุปกรณ์
ชื่อของไฟล์พิเศษที่เชื่อมโยงกับอุปกรณ์

Xmit Pkts/s
แพ็กเก็ตต่อวินาทีที่ส่งผ่านทางอุปกรณ์นี้

Xmit Bytes/s
ไบต์ต่อวินาทีที่ส่งผ่านทางอุปกรณ์นี้

Xmit Util
เวลาที่ยุ่งของอุปกรณ์นี้ ซึ่งแสดงเป็นเปอร์เซ็นต์ของเวลาทั้งหมด

Xmit Qlen
จำนวนการร้องขอที่กำลังรอจะส่งผ่านทางอุปกรณ์นี้ โดยเฉลี่ยในช่วงเวลาหนึ่ง รวมถึงธุรกรรมใดๆ ที่กำลังส่งผ่านอยู่ในปัจจุบัน

Recv Pkts/s
แพ็กเก็ตต่อวินาทีที่ได้รับผ่านทางอุปกรณ์นี้

Recv Bytes/s
ไบต์ต่อวินาทีที่ได้รับผ่านทางอุปกรณ์นี้

Recv Demux
เวลาที่ใช้ในชั้น demux เป็นเศษส่วนของเวลาทั้งหมด

ในตัวอย่างนี้ `Xmit QLen` คือ 27.303 `Recv Bytes/s` คือ 10957295 (10.5 MB/วินาที) ซึ่งปิดตามขีดจำกัด wire ของ 100 Mbps Ethernet ดังนั้น ในกรณีนี้ เครือข่ายจึงเกือบเต็มแล้ว

สถิติการส่งผ่านไดรเวอร์อุปกรณ์เครือข่ายโดยเรียงตามโฮสต์ปลายทาง:

คำสั่ง **netpmon** สามารถใช้เพื่อสร้าง รายงานที่แสดงรายการสถิติการส่งผ่านไดรเวอร์อุปกรณ์เครือข่ายโดยเรียงตามโฮสต์ปลายทาง

แต่ละแถวอธิบายจำนวนของการจราจรส่งผ่านที่เชื่อมโยงกับ โฮสต์ปลายทางเฉพาะที่ระดับไดรเวอร์อุปกรณ์

โฮสต์ ชื่อโฮสต์ปลายทาง เครื่องหมายดอกจัน (*) ใช้สำหรับการส่งผ่านซึ่ง ไม่สามารถกำหนดชื่อโฮสต์ได้

Pkts/s แพ็กเก็ตต่อวินาทีที่ส่งผ่านไปยังโฮสต์นี้

Bytes/s ไบต์ต่อวินาทีที่ส่งผ่านไปยังโฮสต์นี้

สถิติการเรียกซ็อกเก็ต TCP สำหรับแต่ละ IP โดยเรียงตามกระบวนการ:

สถิติเหล่านี้แสดงขึ้นสำหรับอินเทอร์เน็ตโปรโตคอลแต่ละรายการที่ใช้

แต่ละแถวอธิบายจำนวนของกิจกรรมรูทีนย่อย **read( )** และ **write( )** บนซ็อกเก็ตของโปรโตคอลชนิดนี้ซึ่งเชื่อมโยงกับกระบวนการเฉพาะ ที่ด้านล่างของรายงาน มีการแสดงผลรวมของการเรียกซ็อกเก็ตทั้งหมดของโปรโตคอลนี้

สถิติเซิร์ฟเวอร์ NFS โดยเรียงตามไคลเอ็นต์:

แต่ละแถวอธิบายจำนวนของกิจกรรม NFS ที่จัดการโดยเซิร์ฟเวอร์นี้ในนามของไคลเอ็นต์เฉพาะ ที่ด้านล่างของรายงาน มีการแสดงผลรวมของ การเรียกของไคลเอ็นต์ทั้งหมด

บนเครื่องไคลเอ็นต์ สถิติเซิร์ฟเวอร์ NFS ถูกแทนที่โดยสถิติ ไคลเอ็นต์ NFS (สถิติไคลเอ็นต์ NFS สำหรับแต่ละเซิร์ฟเวอร์ (โดยเรียงตามไฟล์), สถิติ NFS Client RPC (โดยเรียงตามเซิร์ฟเวอร์), สถิติไคลเอ็นต์ NFS (โดยเรียงตามกระบวนการ))

รายงานที่แสดงรายละเอียดของ netpmon:

รายงานที่แสดงรายละเอียดจะถูกสร้างขึ้นสำหรับชนิดของรายงานที่ร้องขอทั้งหมด (**-O**) สำหรับชนิดของรายงานเหล่านี้ รายงานที่แสดงรายละเอียดจะถูกสร้างขึ้น นอกเหนือจากรายงานแบบโกลบอล รายงานที่แสดงรายละเอียดจะมีรายการ สำหรับแต่ละรายการที่อยู่ในรายงานแบบโกลบอล พร้อมกับข้อมูลสถิติสำหรับแต่ละชนิดของการทำรายการที่เชื่อมโยงกับรายการ

ข้อมูลสถิติการทำรายการประกอบด้วยการนับจำนวนของการทำรายการสำหรับชนิดนั้น ตามด้วยเวลาตอบสนองและข้อมูลการแจกแจงขนาด (ซึ่งสามารถนำไปปรับใช้ได้) ข้อมูลการแจกแจงประกอบด้วยค่าเฉลี่ย ค่าต่ำสุด และค่าสูงสุดพร้อมกับค่าความเบี่ยงเบนมาตรฐาน โดยทั่วไป สองในสามของค่า จะอยู่ระหว่างค่าเฉลี่ยลบค่าความเบี่ยงเบนมาตรฐานกับค่าเฉลี่ยบวกค่าความเบี่ยงเบนมาตรฐาน ขนาดจะถูกรายงานในหน่วยไบต์ เวลาตอบสนองจะถูกรายงานในหน่วยมิลลิวินาที

ข้อมูลสถิติการใช้ไมโครโพรเซสเซอร์ Interrupt Handler ระดับที่สองโดยละเอียด:

คำสั่ง **netpmon** สามารถสร้างรายงานที่แสดงข้อมูลสถิติการใช้ไมโครโพรเซสเซอร์ Interrupt Handler ระดับที่สองโดยละเอียด

ฟิลด์เอาต์พุตจะถูกกล่าวถึงดังนี้:

SLIH ชื่อของ interrupt handler ระดับที่สอง

count จำนวนของอินเตอร์รัปต์สำหรับชนิดนี้

cpu time (msec)
ข้อมูลสถิติการใช้ไมโครโพรเซสเซอร์สำหรับการจัดการกับอินเตอร์รัปต์ ของชนิดนี้

ข้อมูลสถิติเกี่ยวกับไดร์เวอร์อุปกรณ์เน็ตเวิร์กโดยละเอียดเรียงลำดับตามอุปกรณ์:

คำสั่ง **netpmon** สามารถสร้างรายงานที่แสดง ข้อมูลสถิติไดร์เวอร์อุปกรณ์เน็ตเวิร์กโดยละเอียดสำหรับอุปกรณ์แต่ละตัวในเน็ตเวิร์ก

ฟิลด์เอาต์พุตจะถูกกล่าวถึงดังนี้:

DEVICE
ชื่อพาธของไฟล์พิเศษที่เชื่อมโยงกับอุปกรณ์

แพ็กเก็ตที่รับ
จำนวนของแพ็กเก็ตที่ได้รับผ่านอุปกรณ์นี้

ขนาดที่รับ (ไบต์)
ข้อมูลสถิติเกี่ยวกับขนาดสำหรับแพ็กเก็ตที่ได้รับ

เวลาที่ได้รับ (มิลลิวินาที)
ข้อมูลสถิติเกี่ยวกับเวลาตอบสนองสำหรับการประมวลผลแพ็กเก็ตที่ได้รับ

เวลา demux (มิลลิวินาที)
ข้อมูลสถิติเกี่ยวกับเวลาสำหรับการประมวลผลแพ็กเก็ตที่ได้รับในเลเยอร์ demux

แพ็กเก็ต xmit
จำนวนของแพ็กเก็ตที่ส่งผ่านอุปกรณ์นี้

ขนาดของ xmit (ไบต์)
ข้อมูลสถิติเกี่ยวกับขนาดสำหรับแพ็กเก็ตที่ส่งผ่าน

เวลาของ xmit (มิลลิวินาที)
ข้อมูลสถิติเกี่ยวกับเวลาตอบสนองสำหรับการประมวลผลแพ็กเก็ตที่ส่งผ่าน

ซึ่งมีรายการโดยละเอียดอื่ๆน เช่น ข้อมูลสถิติการส่งผ่านไดร์เวอร์อุปกรณ์เน็ตเวิร์กโดยละเอียด (เรียงลำดับตามโฮสต์) และ ข้อมูลสถิติเกี่ยวกับการเรียก TCP ช็อกเก็ตสำหรับอินเตอร์เน็ตโปรโตคอลแต่ละตัว (เรียงลำดับตามการประมวลผล) สำหรับไคลเอ็นต์ NFS มีรายงาน ข้อมูลสถิติเกี่ยวกับไคลเอ็นต์ NFS สำหรับแต่ละเซิร์ฟเวอร์โดยละเอียด (เรียงลำดับตามไฟล์), ข้อมูลสถิติเกี่ยวกับ NFS Client RPC โดยละเอียด (เรียงลำดับตามเซิร์ฟเวอร์) และ ข้อมูลสถิติเกี่ยวกับไคลเอ็นต์ NFS โดยละเอียด (เรียงลำดับตามการประมวลผล) สำหรับเซิร์ฟเวอร์ NFS มีรายงาน ข้อมูลสถิติเกี่ยวกับเซิร์ฟเวอร์ NFS โดยละเอียด (เรียงลำดับตามไคลเอ็นต์) ซึ่งจะคล้ายกับฟิลด์เอาต์พุตที่อธิบายข้างต้น

ในตัวอย่าง ผลลัพธ์จาก ข้อมูลสถิติเกี่ยวกับไดร์เวอร์อุปกรณ์เน็ตเวิร์กโดยละเอียด จะเป็นดังต่อไปนี้:

- recv bytes = 33003 packets * 73.2 bytes/packet = 2,415,819.6 bytes
- xmit bytes = 61837 packets * 1514 bytes/packet = 93,621,218 bytes
- total bytes exchanged = 2,415,819.6 + 93,621,218 = 96,037,037.6 bytes
- total bits exchanged = 96,037,037.6 * 8 bits/byte = 768,296,300.8 bits
- network speed = 768,296,300.8 / 8.54 = 89,964,438 bits/sec (ประมาณ 90 Mbps) – สมมุติว่าสำเนาของ NFS ใช้จำนวนของการติดตามทั้งหมด

เนื่องจากในรายงานไดร์เวอร์อุปกรณ์แบบโกลบอล คุณสามารถสรุปได้ว่า ในกรณีนี้จะเต็มไปด้วยเน็ตเวิร์ก ขนาดที่ได้รับโดยเฉลี่ยคือ 73.2 ไบต์ และสะท้อนถึงความเป็นจริงที่ว่า เซิร์ฟเวอร์ NFS ที่ถูกติดตาม จะได้รับการตอบรับ สำหรับข้อมูลที่เซิร์ฟเวอร์ส่งไป ขนาดการส่งโดยเฉลี่ยคือ 1514 ไบต์ ซึ่งเป็น MTU (maximum transmission unit) ดีฟอลต์สำหรับอุปกรณ์อีเทอร์เน็ต *อินเตอร์เฟส* การแทนที่ *อินเตอร์เฟส* พร้อมกับชื่ออินเตอร์เฟส เช่น en0 หรือ tr0 คุณสามารถเปลี่ยน MTU หรือค่าความยาวของคิวการส่งผ่านอะแด็ปเตอร์เพื่อขอรับผลการทำงานที่ดีกว่า ด้วยคำสั่งต่อไปนี้:

```
# ifconfig tr0 mtu 8500
```

หรือ

```
# chdev -l 'tok0' -a xmt_que_size='150'
```

ถ้าเน็ตเวิร์กแออัดอยู่แล้ว การเปลี่ยนแปลงค่า MTU หรือคิวจะไม่ได้ช่วยให้เกิดประโยชน์

หมายเหตุ:

1. ถ้าขนาดของแพ็กเก็ตการส่งผ่านและการรับมีขนาดเล็กบนรายงานข้อมูลสถิติเกี่ยวกับไดร์เวอร์อุปกรณ์ การเพิ่มขนาดของ MTU ปัจจุบันจะส่งผลทำให้ผลการทำงานของเน็ตเวิร์ก ดีขึ้น
2. ถ้าช่วงเวลารอของระบบเนื่องจากการเรียกเน็ตเวิร์กมีปริมาณสูงจากข้อมูลสถิติช่วงเวลาของเน็ตเวิร์ก สำหรับรายงานไคลเอ็นต์ NFS ผลการทำงานที่ไม่ดีอาจมีสาเหตุมาจาก เน็ตเวิร์ก

ข้อจำกัดของคำสั่ง netpmon:

คำสั่ง **netpmon** จะใช้ตัวช่วยการติดตาม เพื่อเก็บรวบรวมข้อมูลสถิติ ดังนั้น จึงมีผลกระทบกับเวิร์กโหลดของระบบ ดังต่อไปนี้

- ในระดับกลางซึ่งคือเวิร์กโหลดเชิงเน็ตเวิร์ก คำสั่ง **netpmon** จะเพิ่มการใช้ประโยชน์จาก CPU ทั้งหมด 3–5 เปอร์เซ็นต์
- ในสภาวะแวดล้อมที่ใช้ CPU อย่างสมบูรณ์แบบด้วย I/O ชนิดใดๆ เพียงเล็กน้อย คำสั่ง **netpmon** ทำให้การคอมไพล์ขนาดใหญ่ช้าลงประมาณ 3.5 เปอร์เซ็นต์

หากต้องการทำให้เกิดสถานการณ์เหล่านี้ลดน้อยลง ให้ใช้การประมวลผลแบบออฟไลน์ และบนระบบที่มี CPU จำนวนมากจะใช้แฟล็ก **-C all** พร้อมกับคำสั่ง **trace**

คำสั่ง traceroute

คำสั่ง **traceroute** มีไว้สำหรับใช้ใน การทดสอบ การประเมิน และการจัดการเครือข่าย

ในขณะที่คำสั่ง **ping** ยืนยันการเข้าถึงได้ของเครือข่าย IP แต่คุณไม่สามารถระบุและแก้ไขปัญหาแยกเฉพาะบางอย่าง ลองพิจารณาสถานการณ์ ต่อไปนี้:

- เมื่อมีหลาย hops (ตัวอย่างเช่น gateways หรือเราต์) อยู่ระหว่างระบบ ของคุณและปลายทาง อาจมีปัญหาบางที่ใน พาธ ระบบปลายทางอาจมีปัญหา แต่คุณต้องทราบว่าแท้จริงแล้ว แพ็กเก็ต หายไปที่ใด
- คำสั่ง **ping** จะหยุดทำงานและไม่ได้บอกให้คุณทราบถึงเหตุผลที่ แพ็กเก็ตหายไป

คำสั่ง **traceroute** สามารถบอกคุณได้ว่าแพ็กเก็ตตั้งอยู่ที่ใด และทำไมเราต์จึงหายไป ถ้าแพ็กเก็ตของคุณต้องเดินทางผ่านเราเตอร์และ ลิงก์ ซึ่งเป็นของและจัดการโดยองค์กรหรือบริษัทอื่น เป็นเรื่องยากที่จะตรวจสอบเราเตอร์ที่เกี่ยวข้องโดยใช้คำสั่ง **telnet** คำสั่ง **traceroute** ให้ข้อมูลเพิ่มเติมสำหรับคำสั่ง **ping**

หมายเหตุ: คำสั่ง **traceroute** ควรจะใช้สำหรับการแยกข้อบกพร่องด้วยตนเองเป็นหลัก เนื่องจากการเพิ่มโหลด บนเครือข่าย จึงไม่ควรใช้คำสั่ง **traceroute** ในระหว่างการดำเนินงานปกติหรือจากสคริปต์อัตโนมัติ

ตัวอย่าง traceroute ที่สำเร็จ

คำสั่ง **traceroute** ใช้ UDP แพ็กเก็ตและใช้ฟังก์ชันการรายงานข้อผิดพลาด ICMP คำสั่งส่ง UDP แพ็กเก็ตสามครั้งไปยังแต่ละ gateway หรือเราเตอร์ระหว่างทาง เริ่มต้นด้วย gateway ที่ใกล้ที่สุดและขยายการค้นหาทีละหนึ่ง hop สุดท้าย การค้นหาไปถึงระบบปลายทาง ในเอาต์พุต คุณเห็นชื่อ gateway, IP address ของ gateway, และเวลาเดินทางไปกลับสามรอบของ gateway ดูตัวอย่าง ต่อไปนี้:

```
# traceroute aix1
trying to get source for aix1
source should be 10.53.155.187
traceroute to aix1.austin.ibm.com (10.53.153.120) from 10.53.155.187 (10.53.155.187), 30 hops max
outgoing MTU = 1500
 1  10.111.154.1 (10.111.154.1)  5 ms  3 ms  2 ms
 2  aix1 (10.53.153.120)  5 ms  5 ms  5 ms
```

ข้อมูลต่อไปนี้เป็นอีกตัวอย่างหนึ่ง:

```
# traceroute aix1
trying to get source for aix1
source should be 10.53.155.187
traceroute to aix1.austin.ibm.com (10.53.153.120) from 10.53.155.187 (10.53.155.187), 30 hops max
outgoing MTU = 1500
 1  10.111.154.1 (10.111.154.1)  10 ms  2 ms  3 ms
 2  aix1 (10.53.153.120)  8 ms  7 ms  5 ms
```

หลังจากรายการ address resolution protocol (ARP) หมดอายุ คำสั่งเดิมจะถูกทำซ้ำ หมายเหตุว่าแพ็กเก็ตแรกไปยังแต่ละ gateway หรือปลายทางใช้เวลาเดินทางไปกลับนานกว่า ที่เป็นเช่นนี้เนื่องจากโอเวอร์เฮด ที่เกิดจาก ARP ถ้ามีการใช้เครือข่ายพับลิกสวิตช์ (WAN) ในเราต์ แพ็กเก็ตแรกจะใช้หน่วยความจำมากเนื่องจากการสร้างการเชื่อมต่อ และอาจก่อให้เกิดไทม์เอาต์ ดีฟอลต์ไทม์เอาต์สำหรับแต่ละแพ็กเก็ตคือ 3 วินาที คุณสามารถเปลี่ยนค่าดีฟอลต์ด้วยอ็อพชัน **-w**

10 ms แรกเกิดขึ้นจาก ARP ระหว่างระบบต้นทาง (9.53.155.187) และ gateway 9.111.154.1 8 ms ที่สองเกิดขึ้นจาก ARP ระหว่าง gateway และปลายทางสุดท้าย (wave) ในกรณีนี้ คุณกำลังใช้ DNS และทุกครั้งก่อนคำสั่ง **traceroute** จะส่งแพ็กเก็ตจะมีการค้นหาเซิร์ฟเวอร์ DNS

ตัวอย่าง traceroute ที่ล้มเหลว

สำหรับพาธยาว ไปยังปลายทางของคุณหรือเราต์เครือข่ายที่ซับซ้อน คุณอาจเห็นปัญหาจำนวนมาก เกี่ยวกับคำสั่ง **traceroute** เนื่องจากหลายสิ่ง มีการพึ่งพาระหว่างกัน การค้นหาปัญหาอาจทำให้คุณเสียเวลา เปล่า ถ้าเราเตอร์หรือระบบทั้งหมดที่เกี่ยวข้องอยู่ภายใต้การควบคุมของคุณ คุณอาจ สามารถสืบสวนปัญหาได้โดยสมบูรณ์

ปัญหา Gateway (เราเตอร์)

ใน ตัวอย่างต่อไปนี้ แพ็กเก็ตถูกส่งจากระบบ 9.53.155.187 มีระบบเราเตอร์ สองระบบระหว่างทางไปยังบริดจ์ ความสามารถในการเราต์ถูกเอาออก จากระบบเราเตอร์ที่สองด้วยความตั้งใจโดยการตั้งค่าอ็อพชัน **ipforwarding** ของคำสั่ง **no** เป็น 0 โปรดดูตัวอย่างต่อไปนี้:

```
# traceroute lamar
trying to get source for lamar
source should be 9.53.155.187
traceroute to lamar.austin.ibm.com (9.3.200.141) from 9.53.155.187 (9.53.155.187), 30 hops max
outgoing MTU = 1500
 1  9.111.154.1 (9.111.154.1)  12 ms  3 ms  2 ms
 2  9.111.154.1 (9.111.154.1)  3 ms !H *  6 ms !H
```

ถ้าได้รับข้อความแสดงข้อผิดพลาด ICMP ไม่รวม `Time Exceeded` และ `Port Unreachable` มีการแสดงดังนี้:

!H โฮสต์ไม่สามารถเข้าถึงได้

!N เครือข่ายไม่สามารถเข้าถึงได้

!P โปรโตคอลไม่สามารถเข้าถึงได้

!S เราต์ต้นทางล้มเหลว

!F ต้องการการแบ่งเฟรกเมนต์

ปัญหาระบบปลายทาง

เมื่อระบบปลายทางไม่ตอบกลับภายในช่วงเวลาไทม์เอาต์ 3 วินาที เคียวรีทั้งหมดจะไทม์เอาต์ และผลลัพธ์แสดงขึ้นพร้อมกับเครื่องหมายดอกจัน (*)

```
# traceroute chuys
trying to get source for chuys
source should be 9.53.155.187
traceroute to chuys.austin.ibm.com (9.53.155.188) from 9.53.155.187 (9.53.155.187), 30 hops max
outgoing MTU = 1500
 1  * * *
 2  * * *
 3  * * *
^C#
```

ถ้าคุณคิดว่าปัญหาเกิดขึ้นจากลิงก์การสื่อสาร ให้ใช้ช่วงเวลาไทม์เอาต์ที่นานขึ้นพร้อมด้วยแฟล็ก **-w** แม้ว่าเกิดขึ้นได้น้อย พอร์ตทั้งหมดที่เคียวรีอาจถูกใช้แล้ว คุณสามารถเปลี่ยนพอร์ต แล้วลอง อีกครั้ง

จำนวนของ "hops" ไปยังปลายทาง

ตัวอย่างเอาต์พุต อีกรายการหนึ่งอาจเป็นดังนี้:

```
# traceroute mysystem.university.edu (129.2.130.22)
traceroute to mysystem.university.edu (129.2.130.22), 30 hops max
1 helios.ee.lbl.gov (129.3.112.1) 0 ms 0 ms 0 ms
2 lilac-dmc.university.edu (129.2.216.1) 39 ms 19 ms 39 ms
3 lilac-dmc.university.edu (129.2.215.1) 19 ms 39 ms 19 ms
4 ccngw-ner-cc.university.edu (129.2.135.23) 39 ms 40 ms 19 ms
5 ccn-nerif35.university.edu (129.2.167.35) 39 ms 39 ms 39 ms
6 csgw/university.edu (129.2.132.254) 39 ms 59 ms 39 ms
7 * * *
8 * * *
9 * * *
10 * * *
11 * * *
12 * * *
13 rip.university.EDU (129.2.130.22) 59 ms! 39 ms! 39 ms!
```

iptrace daemon และคำสั่ง ipreport และ ipfilter

คุณสามารถใช้เครื่องมือจำนวนมากสำหรับการเฝ้าดูกิจกรรมของเน็ตเวิร์ก เครื่องมือบางส่วนจะรันภายใต้ระบบปฏิบัติการ และเครื่องมือบางส่วนจะรันอยู่บนฮาร์ดแวร์เฉพาะงาน เครื่องมือหนึ่งที่สามารถนำมาใช้ เพื่อขอรับคำอธิบายโดยละเอียดแบบแพ็กเก็ตต่อแพ็กเก็ตของกิจกรรม LAN ซึ่งสร้างโดยเวิร์กโหลดคือ ชุดของ **iptrace** daemon และคำสั่ง **ipreport**

หากต้องการใช้ **iptrace** daemon ด้วยระบบปฏิบัติการเวอร์ชัน 4 คุณจำเป็นต้องมีชุดไฟล์ `bos.net.tcp.server` **iptrace** daemon จะสอดแทรกอยู่ในชุดไฟล์นี้พร้อมกับคำสั่งอื่นๆ บางส่วน เช่น คำสั่ง **trpt** และ **tcdump iptrace** daemon สามารถเริ่มทำงานได้โดยผู้ใช้ราก

ตามค่าดีฟอลต์แล้ว **iptrace** daemon จะติดตามแพ็กเก็ตทั้งหมด อ็อพชัน **-a** อนุญาตให้แยกแพ็กเก็ต address resolution protocol (ARP) อ็อพชันอื่นๆ สามารถทำให้มีขอบเขตการติดตามที่แคบลงในโฮสต์ต้นทางโดยเฉพาะ (**-s**) โฮสต์ปลายทาง (**-d**) หรือโปรโตคอล (**-p**) เนื่องจาก **iptrace** สามารถใช้จำนวนเวลาของตัวประมวลผลที่สำคัญ ซึ่งได้ระบุไว้เมื่อคุณอธิบายถึงแพ็กเก็ตที่คุณติดตาม

เนื่องจาก **iptrace** คือ daemon ให้เริ่มต้น **iptrace** daemon ด้วยคำสั่ง **startsrc** แทนที่จะเริ่มต้นจากบรรทัดรับคำสั่งโดยตรง เมธอดนี้ช่วยทำให้ควบคุมและปิดได้อย่างง่ายดาย ตัวอย่างจะเป็นดังต่อไปนี้:

```
# startsrc -s iptrace -a "-i en0 /home/user/iptrace/log1"
```

คำสั่งนี้จะเริ่มต้น **iptrace** daemon ด้วยคำสั่ง เพื่อติดตามกิจกรรมทั้งหมดบนอินเตอร์เฟสกิกะบิตอีเทอร์เน็ต en0 และวางข้อมูลการติดตามลงใน /home/user/iptrace/log1 หากต้องการหยุด daemon ให้ใช้คำสั่งต่อไปนี้:

```
# stopsrc -s iptrace
```

ถ้าคุณไม่ได้เริ่มต้น **iptrace** daemon ด้วยคำสั่ง **startsrc** คุณต้องใช้คำสั่ง **ps** เพื่อค้นหา ID การประมวลผล และยกเลิกด้วยคำสั่ง **kill**

คำสั่ง **ipreport** คือตัวจัดรูปแบบสำหรับไฟล์บันทึกการทำงาน เอาต์พุตจะถูกเขียนลงในเอาต์พุตมาตรฐาน อ็อพชันอนุญาตให้จดจำและจัดรูปแบบแพ็กเก็ต RPC (**-r**) ระบุแต่ละแพ็กเก็ตด้วยจำนวน (**-n**) และใช้คำนำหน้าแต่ละบรรทัดด้วยสตริงอักขระ 3 ตัวอักษรซึ่งบ่งชี้ถึงโปรโตคอล (**-s**) ตัวอย่างคำสั่ง **ipreport** เพื่อจัดรูปแบบไฟล์ log1 ที่เพิ่งสร้างขึ้น (ซึ่งผู้ใช้รากเป็นเจ้าของ) มีดังต่อไปนี้:

```
# ipreport -ns log1 >log1_formatted
```

ผลลัพธ์จะเป็นลำดับของแพ็กเก็ตที่รายงานซึ่งคล้ายกับตัวอย่าง ต่อไปนี้ แพ็กเก็ตแรกคือ ส่วนของครึ่งแรกของแพ็กเก็ต **ping** ฟิลด์ที่น่าสนใจมีดังต่อไปนี้:

- โฮสต์แอดเดรสต้นทาง (SRC) และปลายทาง (DST) ซึ่งจะอยู่ในรูปของจุดทศนิยมและใน ASCII
- ความยาวของ IP แพ็กเก็ต (ip_len)
- การบ่งชี้ของโปรโตคอลในระดับที่สูงกว่าที่ใช้งานอยู่ (ip_p)

```
Packet Number 7
ETH: ====( 98 bytes transmitted on interface en0 )==== 10:28:16.516070112
ETH:    [ 00:02:55:6a:a5:dc -> 00:02:55:af:20:2b ]  type 800  (IP)
IP:     < SRC =       192.1.6.1 >  (en6host1)
IP:     < DST =       192.1.6.2 >  (en6host2)
IP:     ip_v=4, ip_hl=20, ip_tos=0, ip_len=84, ip_id=1789, ip_off=0
IP:     ip_ttl=255, ip_sum=28a6, ip_p = 1 (ICMP)
ICMP:   icmp_type=8 (ECHO_REQUEST)  icmp_id=18058  icmp_seq=3

Packet Number 8
ETH: ====( 98 bytes received on interface en0 )==== 10:28:16.516251667
ETH:    [ 00:02:55:af:20:2b -> 00:02:55:6a:a5:dc ]  type 800  (IP)
IP:     < SRC =       192.1.6.2 >  (en6host2)
IP:     < DST =       192.1.6.1 >  (en6host1)
IP:     ip_v=4, ip_hl=20, ip_tos=0, ip_len=84, ip_id=11325, ip_off=0
IP:     ip_ttl=255, ip_sum=366, ip_p = 1 (ICMP)
ICMP:   icmp_type=0 (ECHO_REPLY)  icmp_id=18058  icmp_seq=3
```

ตัวอย่างถัดไปคือกรอบจากการดำเนินการ **ftp** โปรดสังเกตว่า IP แพ็กเก็ตคือขนาดของ MTU สำหรับ LAN นี้ (1492 ไบต์)

```
Packet Number 20
ETH: ====( 1177 bytes transmitted on interface en0 )==== 10:35:45.432353167
ETH:    [ 00:02:55:6a:a5:dc -> 00:02:55:af:20:2b ]  type 800  (IP)
IP:     < SRC =       192.1.6.1 >  (en6host1)
IP:     < DST =       192.1.6.2 >  (en6host2)
IP:     ip_v=4, ip_hl=20, ip_tos=8, ip_len=1163, ip_id=1983, ip_off=0
```

```
IP:     ip_ttl=60, ip_sum=e6a0, ip_p = 6 (TCP)
TCP:    <source port=32873, destination port=20(ftp-data) >
TCP:    th_seq=623eabdc, th_ack=973dcd95
TCP:    th_off=5, flags<PUSH | ACK>
TCP:    th_win=17520, th_sum=0, th_urp=0
TCP: 00000000     69707472 61636520 322e3000 00008240     |iptrace 2.0....@|
TCP: 00000010     2e4c9d00 00000065 6e000065 74000053     |.L.....en..et..S|
TCP: 00000020     59535841 49584906 01000040 2e4c9d1e     |YSXAIXI....@.L..|
TCP: 00000030     c0523400 0255af20 2b000255 6aa5dc08     |.R4..U. +..Uj...|
TCP: 00000040     00450000 5406f700 00ff0128 acc00106     |.E..T......(....|
TCP: 00000050     01c00106 0208005a 78468a00 00402e4c     |.......ZxF...@.L|
TCP: 00000060     9d0007df 2708090d 0a0b0c0d 0e0f1011     |....'...........|
TCP: 00000070     12131415 16171819 1a1b1c1d 1e1f2021     |.............. !|
TCP: 00000080     22232425 26272829 2a2b2c2d 2e2f3031     |"#$%&'()*+,-./01|
TCP: 00000090     32333435 36370000 0082402e 4c9d0000     |234567....@.L...|
             --------- Lots of uninteresting data omitted -----------
TCP: 00000440     15161718 191a1b1c 1d1e1f20 21222324     |........... !"#$|
TCP: 00000450     25262728 292a2b2c 2d2e2f30 31323334     |%&'()*+,-./01234|
TCP: 00000460     353637                                  |567             |
```

คำสั่ง **ipfilter** จะแตกส่วนหัวของการดำเนินการอื่นๆ จากไฟล์เอาต์พุต **ipreport** และแสดงส่วนหัวเหล่านั้นในตาราง ข้อมูล NFS ที่กำหนดเองบางส่วนโดยพิจารณาถึงคำร้องขอและการตอบกลับจะถูกจัดเตรียมไว้

หากต้องการพิจารณาว่า คำสั่ง **ipfilter** ได้ถูกติดตั้งและพร้อมใช้งาน ให้รันคำสั่งต่อไปนี้:

```
# lslpp -lI perfagent.tools
```

ตัวอย่างคำสั่งจะเป็นดังนี้:

```
# ipfilter log1_formatted
```

ส่วนหัวของการดำเนินการที่จดจำไว้ในปัจจุบันคือ: udp, nfs, tcp, ipx, icmp คำสั่ง **ipfilter** มีชนิดของรายงานที่แตกต่างกันอยู่สามแบบ ดังนี้:

- ไฟล์เดี่ยว (`ipfilter.all`) ซึ่งแสดงรายการของการดำเนินการ ที่เลือกไว้ทั้งหมด ตารางที่แสดงหมายเลขแพ็กเก็ต เวลาต้นทาง & ปลายทาง ความยาว ลำดับที่ # Ack # พอร์ตต้นทาง พอร์ตปลายทาง เน็ตเวิร์กอินเตอร์เฟส และชนิดของการดำเนินการ
- ไฟล์แต่ละไฟล์สำหรับส่วนหัวที่เลือกไว้แต่ละส่วน (`ipfilter.udp`, `ipfilter.nfs`, `ipfilter.tcp`, `ipfilter.ipx`, `ipfilter.icmp`) ข้อมูลที่มีอยู่จะเหมือนกับ `ipfilter.all`
- ไฟล์ `nfs.rpt` ที่รายงานอยู่บนคำร้องขอและการตอบกลับ NFS ตารางจะประกอบด้วย: ID การทำรายการ# ชนิดของคำร้องขอ สถานะของคำร้องขอ หมายเลขแพ็กเก็ตการเรียก เวลาของการเรียก ขนาดของการเรียก หมายเลขแพ็กเก็ตการตอบกลับ เวลาที่ตอบกลับ ขนาดของการตอบกลับ และเวลาที่ใช้ไปในหน่วยมิลลิวินาทีระหว่างการเรียและการตอบกลับ

ข้อมูลสถิติของอะแด็ปเตอร์

คำสั่งที่อยู่ในส่วนนี้จะแสดงเอาต์พุตที่เปรียบเทียบกับคำสั่ง **netstat -v** ซึ่งอนุญาตให้คุณรีเซ็ตข้อมูลสถิติของอะแด็ปเตอร์ (**-r**) และเพื่อขอรับเอาต์พุตโดยละเอียดเพิ่มเติม (**-d**) จากเอาต์พุตของคำสั่ง **netstat -v** ที่จัดเตรียมไว้

คำสั่ง entstat

คำสั่ง **entstat** จะแสดงข้อมูลสถิติที่รวบรวมไว้โดยไดร์เวอร์อุปกรณ์อีเทอร์เน็ตที่ระบุเฉพาะ ผู้ใช้สามารถเลือกที่จะระบุว่า ข้อมูลสถิติที่ระบุเฉพาะอุปกรณ์จะแสดงเพิ่มเติมจากข้อมูลสถิติ ทั่วไปของอุปกรณ์ การใช้อ็อพชัน **-d** จะแสดงข้อมูลสถิติที่ขยายเพิ่มสำหรับอะแด็ปเตอร์นี้ และควรนำมาใช้เพื่อทำให้มั่นใจว่า ข้อมูลสถิติทั้งหมดได้ถูกแสดงแล้ว ถ้าไม่ได้ระบุแฟล็กใดๆ ข้อมูลสถิติทั่วไปของอุปกรณ์ จะแสดงขึ้น

คำสั่ง **entstat** จะเรียกใช้งานเมื่อคำสั่ง **netstat** รันพร้อมกับแฟล็ก **-v** คำสั่ง **netstat** จะไม่ออกคำสั่ง **entstat** ด้วยแฟล็กใดๆ

```
# entstat ent0
-------------------------------------------------------------
ETHERNET STATISTICS (ent0) :
Device Type: 10/100/1000 Base-TX PCI-X Adapter (14106902)
Hardware Address: 00:02:55:6a:a5:dc
Elapsed Time: 1 days 18 hours 47 minutes 34 seconds

Transmit Statistics:                          Receive Statistics:
--------------------                          -------------------
Packets: 1108055                              Packets: 750811
Bytes: 4909388501                             Bytes: 57705832
Interrupts: 0                                 Interrupts: 681137
Transmit Errors: 0                            Receive Errors: 0
Packets Dropped: 0                            Packets Dropped: 0
                                              Bad Packets: 0
Max Packets on S/W Transmit Queue: 101
S/W Transmit Queue Overflow: 0
Current S/W+H/W Transmit Queue Length: 0

Broadcast Packets: 3                          Broadcast Packets: 3
Multicast Packets: 3                          Multicast Packets: 5
No Carrier Sense: 0                           CRC Errors: 0
DMA Underrun: 0                               DMA Overrun: 0
Lost CTS Errors: 0                            Alignment Errors: 0
Max Collision Errors: 0                       No Resource Errors: 0
Late Collision Errors: 0                      Receive Collision Errors: 0
Deferred: 0                                   Packet Too Short Errors: 0
SQE Test: 0                                   Packet Too Long Errors: 0
Timeout Errors: 0                             Packets Discarded by Adapter: 0
Single Collision Count: 0                     Receiver Start Count: 0
Multiple Collision Count: 0
Current HW Transmit Queue Length: 0

General Statistics:
-------------------
No mbuf Errors: 0
Adapter Reset Count: 0
Adapter Data Rate: 2000
Driver Flags: Up Broadcast Running
        Simplex 64BitSupport ChecksumOffload
        PrivateSegment LargeSend DataRateSet
```

ในรายงานข้างต้น คุณอาจต้องการให้เน้นความสนใจในเรื่อง:

ข้อผิดพลาดในการส่งผ่าน

จำนวนของข้อผิดพลาดเอาต์พุตที่พบในอุปกรณ์นี้ นี่คือตัวนับการส่งผ่านที่ไม่เป็นผลสำเร็จ เนื่องจากข้อผิดพลาดด้านฮาร์ดแวร์/เน็ตเวิร์ก

ข้อผิดพลาดในการรับ

จำนวนของข้อผิดพลาดเอาต์พุตที่พบในอุปกรณ์นี้ นี่คือตัวนับการรับที่ไม่เป็นผลสำเร็จ เนื่องจากข้อผิดพลาดด้านฮาร์ดแวร์/เน็ตเวิร์ก

แพ็กเก็ตที่ปล่อย

จำนวนของแพ็กเก็ตที่ยอมรับโดยไดร์เวอร์อุปกรณ์สำหรับการส่งที่ไม่ได้กำหนดไว้ในอุปกรณ์ (ด้วยเหตุผลบางประการ)

แพ็กเก็ตสูงสุดสำหรับคิวการส่งผ่านซอฟต์แวร์

จำนวนสูงสุดของแพ็กเก็ตขาออกที่อยู่ในคิวสำหรับการส่งผ่านซอฟต์แวร์

คิวโอเวอร์โฟลว์การส่งผ่านซอฟต์แวร์

จำนวนของแพ็กเก็ตขาออกที่โอเวอร์โฟลว์คิวการส่งผ่าน

ข้อผิดพลาดที่ไม่มีรีซอร์ส

จำนวนของแพ็กเก็ตขาเข้าที่ตกหล่นโดยฮาร์ดแวร์เนื่องจากการขาดรีซอร์ส โดยปกติแล้ว ข้อผิดพลาดนี้จะเกิดขึ้นเนื่องจาก บัฟเฟอร์การรับบนอะแด็ปเตอร์มีปริมาณลดลง อะแด็ปเตอร์บางตัวอาจมีขนาดของบัฟเฟอร์การรับเหมือนกับ พารามิเตอร์ที่สามารถปรับแต่งได้ ให้ตรวจสอบแอ็ตทริบิวต์คอนฟิกูเรชันอุปกรณ์ (หรือวิธีใช้ SMIT) สำหรับข้อมูลการปรับที่เป็นไปได้

จำนวนความขัดแย้งเดี่ยว/จำนวนความขัดแย้งจำนวนมาก

จำนวนของความขัดแย้งบนเน็ตเวิร์กแบบอีเทอร์เน็ต ความขัดแย้งเหล่านี้จะถูกอธิบายไว้ที่นี่ แทนที่จะอยู่ในคอลัมน์ความขัดแย้งของเอาต์พุตของคำสั่ง **netstat -i**

ให้สังเกตในตัวอย่างนี้ อีเทอร์เน็ตอะแด็ปเตอร์ จะมีลักษณะการทำงานที่ดีเนื่องจากไม่มี ข้อผิดพลาดในการรับ ข้อผิดพลาดเหล่านี้อาจเกิดขึ้นได้ในบางครั้ง เมื่อเน็ตเวิร์กที่ปริมาณมากจะส่งผ่านแพ็กเก็ตบางส่วน แพ็กเก็ตบางส่วนจะถูกส่งผ่านอีกครั้งได้เป็นผลสำเร็จ แต่จะถูกบันทึกเป็นข้อผิดพลาด ในการรับ

ถ้าคุณได้รับข้อผิดพลาด คิวโอเวอร์โฟลว์การส่งผ่านซอฟต์แวร์ ค่าของ แพ็กเก็ตสูงสุดบนคิวการส่งผ่านซอฟต์แวร์ จะสอดคล้องกับข้อจำกัดของคิวการส่งผ่าน สำหรับอะแด็ปเตอร์นี้ (**xmt_que_size**)

หมายเหตุ: ค่าเหล่านี้สามารถแสดงถึง *คิวฮาร์ดแวร์* ถ้าอะแด็ปเตอร์ไม่สนับสนุนคิวการส่งผ่านซอฟต์แวร์ ถ้ามีโอเวอร์โฟลว์คิวการส่งผ่านแล้ว ให้เพิ่มข้อจำกัดของคิวฮาร์ดแวร์หรือซอฟต์แวร์สำหรับไดร์เวอร์

ถ้ามีรีซอร์สการรับที่ไม่เพียงพอ เหตุการณ์นี้จะบ่งชี้โดย แพ็กเก็ตที่ตกหล่น : และขึ้นอยู่กับชนิดของอะแด็ปเตอร์ ซึ่งจะบ่งชี้โดย บัฟเฟอร์การรับที่ไม่เพียงพอ หรือ ข้อผิดพลาดที่ไม่มีรีซอร์ส: หรือตัวนับที่คล้างคลึงกันบางส่วน

เวลาที่ใช้ไป จะแสดงระยะเวลาแบบเรียวไทม์ซึ่งจะผ่านไปตั้งแต่เวลาครั้งล่าสุดที่รีเซ็ต ข้อมูลสถิติ หากต้องการรีเซ็ตข้อมูลสถิติ ให้ใช้คำสั่ง **entstat -r** *adapter_name*

เช่นเดียวกับเอาต์พุต ที่สามารถแสดงโทเค็นริง FDDI และอินเตอร์เฟส ATM โดยใช้คำสั่ง **tokstat, fddistat** และ **atmstat**

คำสั่ง tokstat

คำสั่ง **tokstat** จะแสดงข้อมูลสถิติที่รวบรวมโดยไดร์เวอร์อุปกรณ์ของโทเค็นริงที่ระบุเฉพาะ ผู้ใช้สามารถเลือกที่จะระบุว่า ข้อมูลสถิติที่ระบุเฉพาะอุปกรณ์จะแสดงเพิ่มเติมจากข้อมูลสถิติ ของไดร์เวอร์อุปกรณ์ ถ้าไม่ได้ระบุแฟล็กไว้ ข้อมูลไดร์เวอร์อุปกรณ์จะถูกแสดงเท่านั้น

คำสั่งนี้ยังเรียกใช้งาน เมื่อคำสั่ง **netstat** รันพร้อมกับแฟล็ก **–v** คำสั่ง **netstat** จะไม่ออกคำสั่ง **tokstat** ด้วยแฟล็กใดๆ

เอาต์พุตจะสร้างขึ้นโดยคำสั่ง **tokstat tok0** และการกำหนดปัญหาจะคล้ายกับที่ได้กล่าวไว้ใน “คำสั่ง entstat” ในหน้า 359

คำสั่ง fddistat

คำสั่ง **fddistat** จะแสดงข้อมูลสถิติที่รวบรวมโดยไดร์เวอร์อุปกรณ์ FDDI ที่ระบุเฉพาะ ผู้ใช้ สามารถเลือกที่จะระบุว่า ข้อมูลสถิติที่ระบุเฉพาะอุปกรณ์จะแสดงเพิ่มเติมจากข้อมูลสถิติ ของไดร์เวอร์อุปกรณ์ ถ้าไม่ได้ระบุแฟล็กไว้ ข้อมูลไดร์เวอร์อุปกรณ์จะถูกแสดงเท่านั้น

คำสั่งนี้ยังเรียกใช้งาน เมื่อคำสั่ง **netstat** รันพร้อมกับแฟล็ก **–v** คำสั่ง **netstat** จะไม่ออกคำสั่ง **fddistat** ด้วยแฟล็กใดๆ

เอาต์พุตจะสร้างขึ้นโดยคำสั่ง **fddistat fddi0** และการกำหนดปัญหาจะคล้ายกับที่ได้กล่าวไว้ใน “คำสั่ง entstat” ในหน้า 359

คำสั่ง atmstat

คำสั่ง **atmstat** จะแสดงข้อมูลสถิติที่รวบรวมโดยไดร์เวอร์อุปกรณ์ ATM ที่ระบุเฉพาะ ผู้ใช้ สามารถเลือกที่จะระบุว่า ข้อมูลสถิติที่ระบุเฉพาะอุปกรณ์จะแสดงเพิ่มเติมจากข้อมูลสถิติ ของไดร์เวอร์อุปกรณ์ ถ้าไม่ได้ระบุแฟล็กไว้ ข้อมูลไดร์เวอร์อุปกรณ์จะถูกแสดงเท่านั้น

เอาต์พุตจะสร้างขึ้นโดยคำสั่ง **atmstat atm0** และการกำหนดปัญหาจะคล้ายกับที่ได้กล่าวไว้ใน “คำสั่ง entstat” ในหน้า 359

คำสั่ง no

ใช้คำสั่ง **no** และแฟล็กของคำสั่งเพื่อแสดง ค่าเครือข่ายปัจจุบัน และเพื่อเปลี่ยนอ็อพชัน

–a พิมพ์อ็อพชันและค่าปัจจุบันทั้งหมด

–d ตั้งค่าอ็อพชันกลับไปเป็นค่าดีฟอลต์

–o *option=NewValue*

หากต้องการรายการของแอ็ตทริบิวต์ทั้งหมดสำหรับคำสั่ง **no** ให้ดู “พารามิเตอร์ที่ปรับได้ของอ็อพชันเครือข่าย” ในหน้า 500

หมายเหตุ: คำสั่ง **no** ทำการตรวจสอบการไม่มีช่วง ถ้าใช้อย่างไม่ถูกต้อง คำสั่ง **no** อาจทำให้ระบบของคุณไม่ทำงานได้

แอ็ตทริบิวต์เครือข่ายบางอย่างเป็นแอ็ตทริบิวต์แบบรันไทม์ซึ่งสามารถเปลี่ยนแปลง ได้ตลอดเวลา แอ็ตทริบิวต์อื่นเป็นแอ็ตทริบิวต์แบบเวลาโหลดซึ่งต้องตั้งค่าก่อนที่จะโหลดส่วนขยายเคอร์นัล **netinet**

หมายเหตุ: เมื่อใช้คำสั่ง **no** เพื่อเปลี่ยนพารามิเตอร์ การเปลี่ยนแปลงมีผลเฉพาะเมื่อบูตระบบครั้งถัดไปเท่านั้น ณ จุดนั้น พารามิเตอร์ทั้งหมดมีการรีเซ็ตเป็นค่าดีฟอลต์แรกเริ่ม

หมายเหตุ: เมื่อต้องการ เปิดใช้งานและปิดใช้งานอ็อพชันคำสั่ง **no** เฉพาะ ในการรีบูตในอนาคต ข้อมูลนี้ต้องมีอยู่ในไฟล์ `/etc/tunables/nextboot` ป้อน **no -r -o** <*no_optionname*>=<*value*> บนบรรทัดรับคำสั่ง เช่น **no -r -o arptab_bsiz=10** บนการรีบูตลำดับถัดไป **arptab_bsiz=10** ยังคงมีผลบังคับใช้ และนำไปใช้กับไฟล์บูตถัดไป

ถ้าระบบของคุณใช้การตั้งค่าคอนฟิกเครือข่าย Berkeley-style ให้ตั้งค่า แอ็ตทริบิวต์ใกล้ตำแหน่งบนสุดของไฟล์ `/etc/rc.bsdnet` ถ้าคุณใช้ระบบ SP ให้แก้ไขไฟล์ `tuning.cust`

ประสิทธิภาพการทำงาน NFS

AIX มี เครื่องมือและเมธอดสำหรับการมอนิเตอร์ Network File System (NFS) และการปรับแต่งทั้ง เซิร์ฟเวอร์ และไคลเอ็นต์

งานที่เกี่ยวข้อง:

"การปรับปรุงผลการทำงานสำหรับการเขียนไคลเอ็นต์ไฟล์ NFS ที่มีขนาดใหญ่" ในหน้า 505
การเขียนขนาดใหญ่ ไฟล์ลำดับผ่านระบบไฟล์ที่ประกอบเข้ากับ NFS สามารถทำให้ลดความรุนแรงในอัตราการถ่ายโอนไฟล์ไปยังเซิร์ฟเวอร์ NFS ในสถานการณ์นี้ คุณระบุสถานการณ์ที่มีอยู่ และใช้ขั้นตอนต่างๆ เพื่อแก้ไขปัญหา

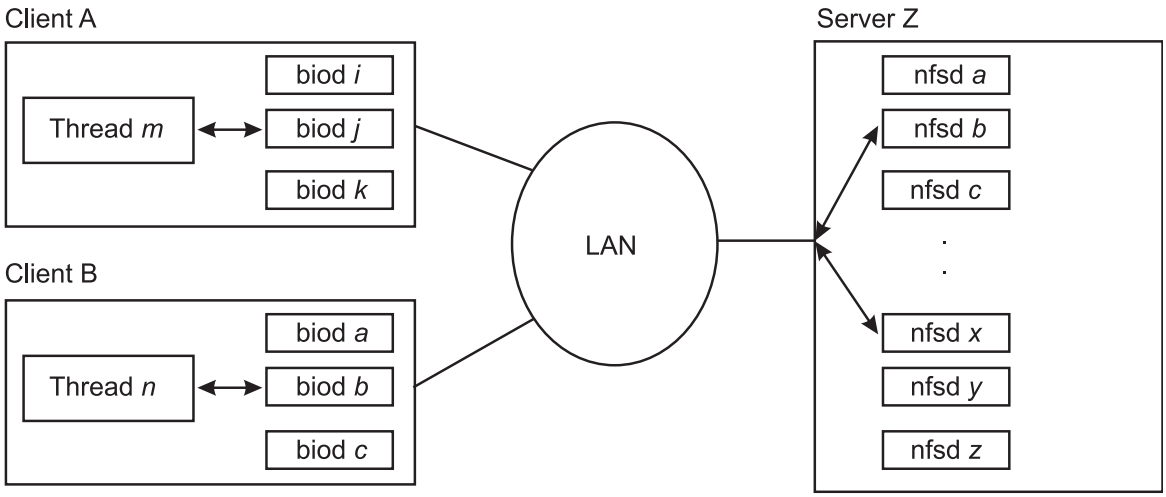
ระบบไฟล์เครือข่าย

NFS ช่วยให้โปรแกรมบนระบบหนึ่งสามารถเข้าถึงไฟล์บนระบบอื่นได้ อย่างโปร่งใสโดยการติดตั้งรีโมตไดเร็กทอรี

โดยปกติ เมื่อบูตเซิร์ฟเวอร์ สามารถทำให้ไดเร็กทอรีมีอยู่ได้โดยใช้คำสั่ง **exportfs** และ daemons ที่จัดการ เข้าถึงแบบรีโมต (**nfsd** daemons) เริ่มต้นขึ้น ในลักษณะคล้ายกัน จุดติดตั้ง ของรีโมตไดเร็กทอรีและการเริ่มต้นของจำนวน NFS block I/O daemons (**biod** daemon) ที่เหมาะสมซึ่งจะจัดการเข้าถึงแบบรีโมต มีการดำเนินการในระหว่างบูตระบบไคลเอ็นต์

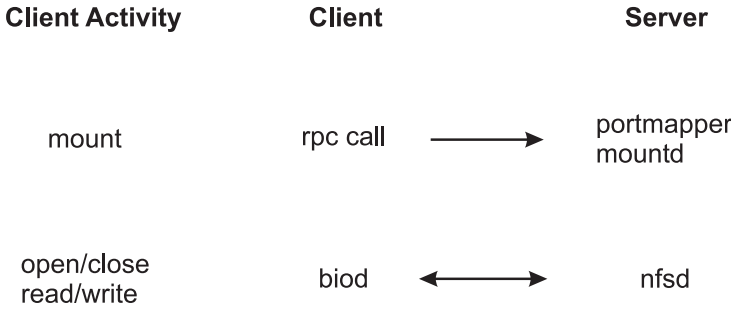
nfsd และ **biod** daemons เป็นแบบมัลติเธรดทั้งคู่ ซึ่งหมายความว่ามีหลายเคอร์นัลเธรดภายในกระบวนการหนึ่ง นอกจากนี้ daemons ยังมีการปรับด้วยตัวเองซึ่งจะสร้างหรือลบเธรด ตามความจำเป็น โดยขึ้นอยู่กับจำนวนของกิจกรรม NFS

รูปภาพต่อไปนี้แสดงโครงสร้างของไดอะล็อกระหว่าง ไคลเอ็นต์และเซิร์ฟเวอร์ NFS เมื่อเธรดในระบบไคลเอ็นต์พยายาม อ่านหรือบันทึกไฟล์ในไดเร็กทอรีที่ติดตั้ง NFS การร้องขอจะถูกกำหนดทิศทางใหม่ จากกลไก I/O ปกติไปยังเธรด **biod** อย่างใดอย่างหนึ่งของไคลเอ็นต์ เธรด **biod** ส่งการร้องขอไปยังเซิร์ฟเวอร์ที่เหมาะสม ซึ่งมีการกำหนดให้กับเธรด NFS อย่างใดอย่างหนึ่งของเซิร์ฟเวอร์ (เธรด **nfsd**) ในขณะที่ประมวลผลการร้องขอ ทั้งเธรด **biod** และ **nfsd** ที่เกี่ยวข้องไม่ได้ทำงานอื่นใด



รูปที่ 22. การโต้ตอบระหว่างไคลเอ็นต์และเซิร์ฟเวอร์ NFS. ภาพสาธิตนี้ แสดงไคลเอ็นต์สองรายการและเซิร์ฟเวอร์หนึ่งรายการบนเครือข่ายใน star topology ปกติ ไคลเอ็นต์ A กำลังรันเธรดแอ็พพลิเคชัน m ซึ่งข้อมูลมีการกำหนดทิศทางไปยังเธรด biod อย่างใดอย่างหนึ่ง ในลักษณะคล้ายกัน ไคลเอ็นต์ B กำลังรันเธรดแอ็พพลิเคชัน n และกำหนดทิศทางข้อมูลไปยังเธรด biod อย่างใดอย่างหนึ่ง เธรดต่างๆ ส่งข้อมูลบนเครือข่ายไปยัง เซิร์ฟเวอร์ Z ซึ่งมีการกำหนดให้กับเธรด NFS (nfsd) อย่างใดอย่างหนึ่งของเซิร์ฟเวอร์

NFS ใช้ Remote Procedure Calls (RPC) เพื่อสื่อสาร RPCs สร้างขึ้น บนส่วนบนสุดของโปรโตคอล External Data Representation (XDR) ซึ่ง แปลงข้อมูลเป็นรูปแบบทั่วไปก่อนที่จะส่งผ่านและอนุญาตให้เครื่อง ที่มีสถาปัตยกรรมแตกต่างอื่นแลกเปลี่ยนข้อมูลได้ ไลบรารี RPC เป็นไลบรารีของโพรซีเดอร์ที่ช่วยให้กระบวนการโลคัล (ไคลเอ็นต์) เข้าถึงกระบวนการรีโมต (เซิร์ฟเวอร์) เพื่อดำเนินการเรียกโพรซีเดอร์ได้ ราวกับว่ากระบวนการโลคัล (ไคลเอ็นต์) ดำเนินการเรียกโพรซีเดอร์ ในพื้นที่ว่างที่อยู่ของตนเอง เนื่องจากไคลเอ็นต์และเซิร์ฟเวอร์เป็นสองกระบวนการที่ แยกกัน จึงไม่จำเป็นต้องมีอยู่บนระบบฟิสิคัลเดียวกันอีกต่อไป



รูปที่ 23. กระบวนการติดตั้งและ NFS. ภาพสาธิตนี้ คือตารางสามคอลัมน์ที่มี กิจกรรมไคลเอ็นต์ ไคลเอ็นต์ และ เซิร์ฟเวอร์ เป็น ส่วนหัวของคอลัมน์ทั้งสาม กิจกรรมไคลเอ็นต์แรกคือการติดตั้ง การเรียก RPC เดินทางจากไคลเอ็นต์ไปยัง portmapper mountd ของเซิร์ฟเวอร์ กิจกรรม ไคลเอ็นต์ที่สองคือการเปิด/การปิดการอ่าน/การบันทึก การโต้ตอบระหว่าง เธรด biod ของไคลเอ็นต์และเธรด nfsd ของเซิร์ฟเวอร์ มีอยู่สองวิธี

portmap daemon, **portmapper** คือ daemon การบริการเครือข่าย ที่จัดเตรียมวิธีมาตรฐานในการค้นหาหมายเลขพอร์ตที่เชื่อมโยงกับโปรแกรม เฉพาะให้แก่ไคลเอ็นต์ เมื่อร้องขอการบริการบนเซิร์ฟเวอร์ เซิร์ฟเวอร์นั้นจะมีการลงทะเบียนด้วย **portmap** daemon เป็นเซิร์ฟเวอร์ ที่มีอยู่ จากนั้น **portmap** daemon เก็บรักษาตารางของคู่ข้อมูล โปรแกรมและพอร์ตไว้

เมื่อไคลเอ็นต์เริ่มต้นการร้องขอไปยังเซิร์ฟเวอร์ไคลเอ็นต์จะติดต่อกับ **portmap** daemon ก่อนเพื่อดูว่าการบริการตั้งอยู่ที่ใด **portmap** daemon รับข้อมูลอยู่บนพอร์ตที่รู้จักกันดี ดังนั้นไคลเอ็นต์จึงไม่ต้องค้นหา **portmap** daemon ตอบกลับไคลเอ็นต์ด้วยพอร์ตของ การบริการที่ไคลเอ็นต์กำลังร้องขอ หลังจากไคลเอ็นต์ได้รับ หมายเลขพอร์ตแล้ว ไคลเอ็นต์สามารถทำการร้องขอในอนาคตทั้งหมดของตนไป ยังแอ็พพลิเคชันได้โดยตรง

mountd daemon คือเซิร์ฟเวอร์ daemon ที่ตอบการร้องขอของไคลเอ็นต์ในการติดตั้งระบบไฟล์หรือไดเร็กทอรีที่ส่งออกของเซิร์ฟเวอร์ **mountd** daemon กำหนดว่าระบบไฟล์ใดมีอยู่โดยการอ่านไฟล์ `/etc/xtab` กระบวนการติดตั้งเกิดขึ้นดังนี้:

1. การติดตั้งของไคลเอ็นต์ทำการเรียกไปยัง **portmap** daemon ของเซิร์ฟเวอร์ เพื่อค้นหาหมายเลขพอร์ตที่กำหนดให้กับ **mountd** daemon
2. **portmap** daemon ส่งผ่านหมายเลขพอร์ตไปยังไคลเอ็นต์
3. จากนั้น คำสั่ง **mount** ของไคลเอ็นต์จะติดต่อเซิร์ฟเวอร์ **mountd** daemon โดยตรง และส่งผ่านชื่อของไดเร็กทอรี ที่ต้องการ
4. เซิร์ฟเวอร์ **mountd** daemon ตรวจสอบ `/etc/xtab` (สร้าง โดยคำสั่ง **exportfs -a** ซึ่งอ่าน `/etc/exports`) เพื่อตรวจสอบการมีอยู่และสิทธิได้รับอนุญาตบนไดเร็กทอรีที่ร้องขอ
5. ถ้าตรวจสอบเกณฑ์ทั้งหมดแล้ว เซิร์ฟเวอร์ **mountd** daemon จะได้รับ การจัดการไฟล์ (ตัวชี้ไปยังไดเร็กทอรีระบบไฟล์) สำหรับไดเร็กทอรีที่ส่งออก และส่งผ่านกลับไปยังเคอร์นัลของไคลเอ็นต์

ไคลเอ็นต์ติดต่อ **portmap** daemon เฉพาะในการร้องขอ การติดตั้งครั้งแรกสุดเท่านั้นนับตั้งแต่ระบบรีสตาร์ท หลังจากไคลเอ็นต์ทราบ หมายเลขพอร์ตของ **mountd** daemon แล้ว ไคลเอ็นต์จะไปยังหมายเลข พอร์ตนั้นโดยตรงเพื่อการร้องขอการติดตั้งในลำดับต่อมา

biod daemon คือบล็อกอินพุต/เอาต์พุต daemon และเป็นสิ่งจำเป็น ในการร้องขอ read-ahead และ write-behind ตลอดจนการอ่านไดเร็กทอรีด้วย เธรด **biod** daemon ช่วยพัฒนาประสิทธิภาพ NFS โดยการเติมหรือทำให้บัฟเฟอร์แคชว่างในนามของแอ็พพลิเคชันไคลเอ็นต์ NFS เมื่อผู้ใช้บนระบบไคลเอ็นต์ต้องการ อ่านจากหรือบันทึกลงในไฟล์บนเซิร์ฟเวอร์ เธรด **biod** จะส่งการร้องขอไปยังเซิร์ฟเวอร์ การดำเนินงาน NFS ต่อไปนี้มีการส่ง ไปยังเซิร์ฟเวอร์โดยตรงจากส่วนขยายเคอร์นัลไคลเอ็นต์ NFS ของ ระบบปฏิบัติการ และไม่ต้องใช้ **biod** daemon:

- **getattr()**
- **setattr()**
- **lookup()**
- **readlink()**
- **create()**
- **remove()**
- **rename()**
- **link()**
- **symlink()**
- **mkdir()**
- **rmdir()**
- **readdir()**
- **readdirplus()**
- **fsstat()**

nfsd daemon คือตัวแทนที่ใช้งานอยู่ซึ่งนำเสนอการบริการ NFS จาก เซิร์ฟเวอร์ NFS การรับการร้องขอโปรโตคอล NFS หนึ่งรายการใดๆ จาก ไคลเอ็นต์จำเป็นต้องใส่ใจเธรด **nfsd** daemon เป็นพิเศษ จนกว่าจะมีการดำเนินการตรงตามการร้องขอและมีการส่งผลของการประมวลผล การร้องขอกลับไปยังไคลเอ็นต์

การขนส่งของเครือข่าย NFS

TCP คือโปรโตคอลการขนส่งดีฟอลต์สำหรับ NFS แต่คุณสามารถใช้ UDP ได้เช่นกัน

คุณสามารถเลือกโปรโตคอลการขนส่งสำหรับแต่ละการติดตั้ง UDP ทำงานได้อย่างมีประสิทธิภาพบนเครือข่ายที่สะอาดหรือมีประสิทธิภาพและเซิร์ฟเวอร์ที่มีการตอบสนองไว สำหรับเครือข่ายที่มีพื้นที่กว้าง หรือเครือข่ายที่ยุ่ง หรือเครือข่ายที่มีเซิร์ฟเวอร์ช้า TCP อาจให้ประสิทธิภาพที่ดีกว่าเนื่องจากการควบคุมโฟลว์ในตัวสามารถลด เวลาแฝงของการส่งผ่านใหม่บนเครือข่ายให้เหลือน้อยที่สุด

เวอร์ชันต่างๆ ของ NFS

AIX สนับสนุน ทั้ง NFS เวอร์ชัน 2 และเวอร์ชัน 3 บนเครื่องเดียวกัน ระบบปฏิบัติการ ยังสนับสนุน NFS เวอร์ชัน 4 ด้วย

NFS เวอร์ชัน 3 ยังคงเป็นค่าดีฟอลต์ต่อไป ถ้าไม่ได้ระบุเวอร์ชัน เป็นอ็อพชันการติดตั้งบนไคลเอ็นต์ AIX เมื่อใช้กับการขนส่งเครือข่าย คุณสามารถเลือกเวอร์ชันของโปรโตคอล NFS สำหรับแต่ละการติดตั้งได้

NFS เวอร์ชัน 4:

NFS เวอร์ชัน 4 เป็นข้อกำหนดคุณลักษณะโปรโตคอลล่าสุดสำหรับ NFS และมีการกำหนดไว้ใน RFC 3530

แม้ว่ามีลักษณะคล้ายกับ NFS เวอร์ชันก่อนหน้านี้ ส่วนใหญ่แล้วคล้ายกับเวอร์ชัน 3 แต่โปรโตคอลใหม่มีการพัฒนาฟังก์ชันใหม่หลายอย่างในพื้นที่ต่างๆ เช่น การรักษาความปลอดภัย การปรับสเกลได้ และการจัดการข้อมูล back-end ลักษณะเหล่านี้ทำให้ NFS เวอร์ชัน 4 เป็นตัวเลือกที่ดีกว่าสำหรับสภาพแวดล้อมการแบ่งใช้ไฟล์ที่แจกจ่าย จำนวนมาก

คุณลักษณะบางอย่างของโปรโตคอล NFS เวอร์ชัน 4 รวมถึงดังต่อไปนี้:

- “การนำไปปฏิบัติของการเปลี่ยนแปลงการดำเนินการ NFS” ในหน้า 366
- “ความต้องการ TCP” ในหน้า 366
- “โปรโตคอลการล็อกที่รวมเข้าด้วยกัน” ในหน้า 366
- “ส่วนสนับสนุนการประกอบเข้าด้วยกัน” ในหน้า 366
- “กลไกการรักษาความปลอดภัยที่ปรับปรุงแล้ว” ในหน้า 366
- “ส่วนสนับสนุนความเป็นสากล” ในหน้า 366
- “แบบจำลองแอ็ตทริบิวต์ที่สามารถขยายเพิ่มได้” ในหน้า 366
- “ส่วนสนับสนุนรายการควบคุมสิทธิเข้าถึง” ในหน้า 367

หมายเหตุว่าฟังก์ชันเพิ่มเติมและความซับซ้อนของโปรโตคอลใหม่ ส่งผลให้โอเวอร์เฮดการประมวลผลเพิ่มขึ้น ด้วยเหตุนี้ประสิทธิภาพของ NFS เวอร์ชัน 4 จึงอาจจะ ช้ากว่า NFS เวอร์ชัน 3 สำหรับหลายแอ็พพลิเคชัน ผลกระทบต่อประสิทธิภาพ แตกต่างกันไปอย่างมาก ขึ้นอยู่กับว่าคุณใช้ฟังก์ชันใหม่ใด ตัวอย่างเช่น ถ้าคุณใช้กลไกการรักษาความปลอดภัยเดียวกันบน NFS เวอร์ชัน 4 และ เวอร์ชัน 3 ระบบของคุณ อาจทำงานช้ากว่าเล็กน้อยเมื่อใช้กับ NFS เวอร์ชัน 4 อย่างไรก็ตาม คุณอาจสังเกต ว่าประสิทธิภาพด้อยลงอย่างมากเมื่อเปรียบเทียบกับประสิทธิภาพของ เวอร์ชัน 3 ที่ใช้การพิสูจน์ตัวตน UNIX แบบดั้งเดิม (AUTH_SYS) เนื่องจาก เวอร์ชัน 4 ใช้ Kerberos 5 ที่มีความเป็นส่วนตัวซึ่งหมายถึง การเข้ารหัสข้อมูลผู้ใช้ทั้งหมด

โปรดทราบด้วยว่า โดยปกติแล้ว คำแนะนำการปรับใดๆ ที่กำหนดสำหรับ NFS เวอร์ชัน 3 ใช้กับ NFS เวอร์ชัน 4 ด้วย

การนำไปปฏิบัติของการเปลี่ยนแปลงการดำเนินการ NFS:

ไม่เหมือนกับ NFS เวอร์ชัน 2 และ 3 ในเวอร์ชัน 4 ประกอบด้วยโพรซีเดอร์ RPC สองโพรซีเดอร์คือ: NULL และ COMPOUND

โพรซีเดอร์ COMPOUND ประกอบด้วยการดำเนินการ NFS ตั้งแต่หนึ่งแบบขึ้นไป ซึ่งได้ถูกนิยามแยกออกจากโพรซีเดอร์ RPC ในเวอร์ชัน NFS ก่อนหน้านี้ การเปลี่ยนแปลงนี้อาจส่งผลทำให้จำเป็นต้องมี RPC เพียงเล็กน้อยในการดำเนินการกับ การดำเนินการระบบไฟล์แบบโลจิคัลผ่านเน็ตเวิร์ก

ความต้องการ TCP:

โปรโตคอล NFS เวอร์ชัน 4 บังคับการใช้โปรโตคอลการขนส่ง ที่มีการควบคุมความคับคั่ง เพื่อให้ประสิทธิภาพการทำงานดีขึ้นในสภาพแวดล้อม WAN

AIX ไม่ สนับสนุนการใช้ UDP ที่มี NFS เวอร์ชัน 4

โปรโตคอลการล็อกที่รวมเข้าด้วยกัน:

NFS เวอร์ชัน 4 ประกอบด้วยส่วนสนับสนุนสำหรับคำแนะนำในการล็อกไฟล์ในช่วงของไบต์

โปรโตคอล Network Lock Manager (NLM) และ `rpc.lockd` และ `rpc.statd` daemons ที่เชื่อมโยงจะไม่ถูกนำมาใช้งาน สำหรับความสามารถในการทำงานร่วมกันที่ดีกว่าด้วยระบบปฏิบัติการที่ไม่ใช่ UNIX NFS เวอร์ชัน 4 ยังสนับสนุนการจองที่แบ่งใช้แบบเปิด และสอดแทรกคุณลักษณะเพื่อให้เหมาะสมกับแพล็ตฟอร์มของเซิร์ฟเวอร์พร้อมกับการล็อกที่บังคับให้ทำ

ส่วนสนับสนุนการประกอบเข้าด้วยกัน:

NFS ในเวอร์ชัน 4 สนับสนุนระบบไฟล์ที่ประกอบเข้าผ่านการดำเนินการของโปรโตคอล

ไคลเอ็นต์ไม่ได้ใช้โปรโตคอลที่ประกอบแยกต่างหากหรือสื่อสารกับ `rpc.mountd` daemon

กลไกการรักษาความปลอดภัยที่ปรับปรุงแล้ว:

NFS เวอร์ชัน 4 ประกอบด้วยส่วนสนับสนุนสำหรับโปรโตคอลการรักษาความปลอดภัย RPCSEC-GSS

โปรโตคอลการรักษาความปลอดภัย RPCSEC-GSS อนุญาตให้ใช้การนำไปปฏิบัติของกลไกการรักษาความปลอดภัยจำนวนมาก โดยไม่จำเป็นต้องเพิ่มนิยามการพิสูจน์ตัวตน RPC ใหม่ สำหรับแต่ละกลไก NFS บน AIX เท่านั้นที่สนับสนุนกลไกการรักษาความปลอดภัย Kerberos 5

ส่วนสนับสนุนความเป็นสากล:

NFS ในเวอร์ชัน 4 ข้อมูลแบบอิงสตริงจะถูกเข้ารหัสด้วย UTF-8 แทนการส่งผ่านข้อมูลดิบในหน่วยไบต์

แบบจำลองแอ็ตทริบิวต์ที่สามารถขยายเพิ่มได้:

แบบจำลองแอ็ตทริบิวต์ใน NFS เวอร์ชัน 4 อนุญาตให้ใช้ความสามารถในการทำงานร่วมกันได้ดีกว่าพร้อมกับการนำไปปฏิบัติแบบไม่ใช้ UNIX และง่ายต่อผู้ใช้ในการเพิ่มนิยามของแอ็ตทริบิวต์

ส่วนสนับสนุนรายการควบคุมสิทธิเข้าถึง:

NFS เวอร์ชัน 4 สอดแทรกนิยามสำหรับแอ็ตทริบิวต์ ACL

แบบจำลอง ACL จะคล้ายกับแบบจำลอง Windows NT นั่นคือ แบบจำลองจะนำเสนอชุดของสิทธิ์ และชนิดของรายการเพื่อให้สิทธิ์หรือปฏิเสธสิทธิ์ สำหรับผู้ใช้หรือกลุ่มพื้นฐาน

NFS เวอร์ชัน 3:

ขอแนะนำอย่างยิ่งให้ใช้ NFS เวอร์ชัน 3 แทน NFS เวอร์ชัน 2 เนื่องจาก คุณลักษณะโปรโตคอลในตัวที่สามารถเพิ่มประสิทธิภาพได้

ผลผลิตการบันทึก:

แอ็พพลิเคชันที่กำลังรันอยู่บนระบบไคลเอ็นต์อาจบันทึกข้อมูลลงในไฟล์ เป็นระยะๆ ซึ่งเป็นการเปลี่ยนเนื้อหาของไฟล์

จำนวนข้อมูลที่แอ็พพลิเคชันสามารถบันทึกลงในหน่วยเก็บคงที่ บนเซิร์ฟเวอร์ในรอบเวลาหนึ่งคือการประเมิน *ผลผลิตการบันทึก* ของ ระบบไฟล์ที่แจกจ่าย ดังนั้น ผลผลิตการผลิตจึงเป็นลักษณะที่สำคัญของ ประสิทธิภาพ ทุกระบบไฟล์ที่แจกจ่าย รวมถึง NFS ต้องทำให้แน่ใจว่าข้อมูล มีการบันทึกลงในไฟล์ปลายทางอย่างปลอดภัย ในขณะที่ลดผลกระทบของ เวลาแฝงเซิร์ฟเวอร์จากผลผลิตการบันทึกให้เหลือน้อยที่สุดในเวลาเดียวกัน

โปรโตคอล NFS เวอร์ชัน 3 นำเสนอวิธีการที่ดีขึ้นในการเพิ่มผลผลิตการบันทึก โดยการตัดความต้องการการบันทึกซิงโครนัสของ NFS เวอร์ชัน 2 ในขณะที่รักษาประโยชน์ของความหมาย close-to-open ไคลเอ็นต์ NFS เวอร์ชัน 3 ลดเวลาแฝงของการดำเนินงานบันทึกในเซิร์ฟเวอร์ลงอย่างมาก โดยการบันทึกข้อมูลลงในข้อมูลไฟล์แคชของเซิร์ฟเวอร์ (หน่วยความจำหลัก) แต่ไม่จำเป็นต้องบันทึกลงในดิสก์ ในเวลาต่อมา ไคลเอ็นต์ NFS ออกใช้การร้องขอการดำเนินงาน commit ไปยังเซิร์ฟเวอร์ ซึ่งช่วยให้มั่นใจว่าเซิร์ฟเวอร์ได้บันทึกข้อมูลทั้งหมด ลงในหน่วยเก็บที่มั่นคง คุณลักษณะนี้เรียกว่า *safe asynchronous writes* ซึ่งลดจำนวนของการร้องขอดิสก์ I/O บนเซิร์ฟเวอร์ได้เป็นอย่างมาก ส่งผลให้ ผลผลิตการบันทึกดีขึ้นอย่างเห็นได้ชัด

การบันทึกถือว่า "ปลอดภัย" เนื่องจากข้อมูลสถานะของข้อมูล มีการเก็บรักษาไว้ เพื่อบ่งชี้ว่ามีการจัดเก็บเสร็จเรียบร้อยแล้วหรือไม่ ด้วยเหตุนี้ ถ้าเซิร์ฟเวอร์เสียหายก่อนการดำเนินงาน commit ไคลเอ็นต์จะทราบโดยการดู ที่การบ่งชี้สถานะ ว่าจะส่งการร้องขอการบันทึกใหม่เมื่อเซิร์ฟเวอร์กลับมาทำงาน ได้หรือไม่

การร้องขอที่ลดลงสำหรับแอ็ตทริบิวต์ไฟล์:

เนื่องจากบางครั้งข้อมูลการอ่านตั้งอยู่ในแคชในช่วงเวลาที่นานขึ้นตาม การคาดการณ์ความต้องการ ไคลเอ็นต์ต้องตรวจสอบเพื่อให้มั่นใจว่าข้อมูลที่แคช ของพวกเขายังคงใช้ได้ถ้าแอ็พพลิเคชันอื่นทำการเปลี่ยนแปลงในไฟล์ ด้วยเหตุนี้ ไคลเอ็นต์ NFS จึงได้รับแอ็ตทริบิวต์ของไฟล์ซึ่งมีเวลาที่แก้ไขไฟล์ ครั้งล่าสุดเป็นครั้งคราว โดยใช้เวลาการแก้ไข ไคลเอ็นต์สามารถทราบได้ว่าข้อมูลที่แคชของพวกเขายังคงใช้ได้หรือไม่

การพยายามทำให้การร้องขอแอ็ตทริบิวต์มีจำนวนต่ำที่สุดช่วยให้ไคลเอ็นต์ มีประสิทธิภาพมากขึ้นและลดเซิร์ฟเวอร์โหลดให้เหลือน้อยที่สุด ส่งผลให้การปรับสเกลได้และประสิทธิภาพสูงขึ้น ดังนั้น NFS เวอร์ชัน 3 จึงได้รับการออกแบบมาเพื่อส่งคืนแอ็ตทริบิวต์สำหรับการดำเนินงานทั้งหมด ลักษณะนี้เพิ่มโอกาส ที่แอ็ตทริบิวต์ในแคชจะทันสมัย และลดจำนวนของการร้องขอแอ็ตทริบิวต์ แยกต่างหาก

การใช้เน็ตเวิร์กเทคโนโลยีที่มีแบนด์วิธสูงอย่างมีประสิทธิภาพ:

การผ่อนผันข้อจำกัดเกี่ยวกับขนาดของ RPC อนุญาตให้ NFS ใช้เทคโนโลยีเน็ตเวิร์กแบนด์วิธสูงได้อย่างมีประสิทธิภาพมากขึ้น เช่น FDDI, 100baseT (100 Mbps) และ 1000baseT (Gigabit) Ethernet และ SP Switch และมีส่วนช่วยทำให้ผลการทำงานของ NFS เพิ่มขึ้นในการอ่านตามลำดับและเพิ่มผลการทำงานเกี่ยวกับการเขียน

NFS เวอร์ชัน 2 จะมีข้อจำกัดเกี่ยวกับขนาด RPC สูงสุด 8 KB ซึ่งจำกัดจำนวนของข้อมูล NFS ที่สามารถถ่ายโอนผ่านเน็ตเวิร์กได้ในหนึ่งครั้ง ใน NFS เวอร์ชัน 3 ข้อจำกัดนี้จะถูกผ่อนผัน ดีฟอลต์ขนาดการอ่าน/เขียนมีขนาด 32 KB สำหรับ NFS บน AIX และขนาดสูงสุด 64 KB ซึ่งเปิดใช้งาน NFS เพื่อสนร้างและถ่ายโอนชิ้นของข้อมูลที่มีขนาดใหญ่กว่า ในหนึ่งแพ็กเก็ต RPC

การร้องขอการค้นหาไดเร็กทอรีที่ลดลง:

รายการไดเร็กทอรีทั้งหมด ตัวอย่างเช่น โดยใช้คำสั่ง **ls -l** มีข้อกำหนดว่าข้อมูลชื่อและแอ็ตทริบิวต์จะต้องได้มาจาก เซิร์ฟเวอร์สำหรับรายการทั้งหมดในรายการไดเร็กทอรี

ไคลเอ็นต์ NFS เวอร์ชัน 2 เคียวรีเซิร์ฟเวอร์แยกต่างหากกันสำหรับรายการของ ไฟล์และข้อมูลชื่อไดเร็กทอรีและแอ็ตทริบิวต์สำหรับรายการไดเร็กทอรีทั้งหมดโดย ใช้การร้องขอการค้นหา ด้วย NFS เวอร์ชัน 3 ข้อมูลรายชื่อและแอ็ตทริบิวต์ มีการส่งคืนพร้อมกันผ่านทางการดำเนินงาน **READDIRPLUS** ซึ่งช่วยให้ ทั้งไคลเอ็นต์และเซิร์ฟเวอร์ไม่ต้องทำภารกิจหลายอย่าง

มีการเพิ่มการสนับสนุนการแคชชื่อไฟล์ที่ยาวขึ้น (มากกว่า 31 อักขระ) ในไดเร็กทอรีไคลเอ็นต์ NFS สำหรับแคชการค้นหาชื่อหรือ **dnlc** การดำเนินการของคุณลักษณะนี้ มีประโยชน์สำหรับโหลดงานของไคลเอ็นต์ NFS ที่ใช้ชื่อไฟล์ยาวมาก ซึ่งก่อนหน้านี้ทำให้เกิดการดำเนินงาน **LOOKUP** ที่ต่อเนื่องบนเซิร์ฟเวอร์ เนื่องจาก **dnlc** misses ตัวอย่างของเวิร์กโหลดชนิดนี้คือคำสั่ง **ls -l** ซึ่งระบุไปแล้วก่อนหน้านี้

การมอนิเตอร์และการปรับประสิทธิภาพ NFS

มีหลายคำสั่งที่คุณสามารถใช้เพื่อมอนิเตอร์สถิติ NFS และเพื่อปรับแอ็ตทริบิวต์ NFS

เพื่อให้ได้ประสิทธิภาพ NFS ที่ดี จำเป็นต้องปรับและแก้ไขปัญหาคอขวด ไม่เฉพาะแต่ภายในตัว NFS เท่านั้น แต่ยังต้องทำภายในระบบปฏิบัติการและ ฮาร์ดแวร์ที่ใช้งานด้วย เวิร์กโหลดที่ประกอบด้วยกิจกรรมการอ่าน/การบันทึกจำนวนมากเป็นสิ่งที่ได้รับผลกระทบได้ง่าย และจำเป็นต้องปรับและตั้งค่าคอนฟิกทั้งระบบ ส่วนนี้ ยังมีข้อมูลเกี่ยวกับเวิร์กโหลดที่อาจจะไม่เหมาะสมอย่างยิ่งสำหรับการใช้ NFS

ตามกฎทั่วไป ก่อนคุณเริ่มต้นปรับค่าของตัวแปรการปรับใดๆ ต้องแน่ใจว่าคุณเข้าใจว่าคุณกำลังพยายามให้เกิดผลสำเร็จอะไรโดยการแก้ไขค่าเหล่านี้ และผลกระทบเชิงลบที่อาจเกิดขึ้นได้จาก การเปลี่ยนแปลงเหล่านี้คืออะไร

สถิติ NFS และพารามิเตอร์การปรับ

NFS รวบรวมสถิติเกี่ยวกับชนิดของการดำเนินงาน NFS ที่ทำ พร้อมกับ ข้อมูลข้อผิดพลาดและตัวบ่งชี้ประสิทธิภาพ

คุณสามารถใช้คำสั่งต่อไปนี้เพื่อระบุปัญหาคอขวดที่อาจเกิดขึ้นได้ สังเกต ชนิดของการดำเนินงาน NFS ที่เกิดขึ้นบนระบบของคุณ และปรับ พารามิเตอร์เฉพาะ NFS

คำสั่ง nfsstat:

คำสั่ง **nfsstat** แสดงข้อมูลสถิติเกี่ยวกับ NFS และอินเตอร์เฟส RPC ไปยังเคอร์นัลของไคลเอ็นต์และเซิร์ฟเวอร์

คำสั่งนี้ยังใช้เพื่อเริ่มต้นตัวนับของสถิติเหล่านี้ใหม่ (**nfsstat -z**) ด้วย สำหรับประเด็นประสิทธิภาพ สถิติ RPC (อ็อพชัน **-r**) นับเป็นข้อมูลแรกที่ควรจะดู สถิติ NFS แสดงให้คุณทราบว่าแอ็พพลิเคชันใช้ NFS อย่างไร

สถิติ RPC:

คำสั่ง **nfsstat** แสดงข้อมูลสถิติเกี่ยวกับ การเรียก RPC

ประเภทสถิติที่แสดงขึ้นมีดังนี้:

- จำนวนทั้งหมดของการเรียก RPC ที่ได้รับหรือที่ปฏิเสธ
- จำนวนทั้งหมดของการเรียก RPC ที่ส่งหรือที่ปฏิเสธโดยเซิร์ฟเวอร์
- จำนวนครั้งที่ไม่มีแพ็กเก็ต RPC เมื่อพยายามจะรับ
- จำนวนแพ็กเก็ตที่สั้นเกินไปหรือมีส่วนหัวที่ผิดรูป
- จำนวนครั้งที่ต้องส่งผ่านการเรียกอีกครั้ง
- จำนวนครั้งที่การตอบไม่ตรงกับการเรียก
- จำนวนครั้งที่การเรียกไทม์เอาต์
- จำนวนครั้งที่การเรียกต้องรอการจัดการของไคลเอ็นต์ที่ยุ่ง
- จำนวนครั้งที่ต้องรีเฟรชข้อมูลการพิสูจน์ตัวตน

ส่วน NFS ของเอาต์พุตคำสั่ง **nfsstat** แบ่งออกเป็นสถิติ ของ NFS เวอร์ชัน 2 และเวอร์ชัน 3 ส่วน RPC แบ่งออกเป็น สถิติ `Connection oriented` (TCP) และ `Connectionless` (UDP)

โปรดอ้างอิง “การปรับประสิทธิภาพ NFS บนเซิร์ฟเวอร์” ในหน้า 378 และ “การปรับ NFS บนไคลเอ็นต์” ในหน้า 382 สำหรับเอาต์พุตเฉพาะของแต่ละหัวข้อ

คำสั่ง nfso:

คุณสามารถใช้คำสั่ง **nfso** เพื่อตั้งค่าคอนฟิกแอ็ตทริบิวต์ NFS

คำสั่งนี้ตั้งค่าหรือแสดงอ็อพชันที่เกี่ยวข้องกับ NFS ซึ่งเชื่อมโยงกับเคอร์นัล ที่กำลังรันอยู่ในปัจจุบันและส่วนขยายเคอร์นัล NFS โปรดดู คำสั่ง nfso ใน *Commands Reference, Volume 4* สำหรับ คำอธิบายโดยละเอียดเกี่ยวกับคำสั่งและเอาต์พุตของคำสั่ง

หมายเหตุ: คำสั่ง **nfso** ไม่ทำ การตรวจสอบช่วง ถ้าใช้อย่างไม่ถูกต้อง คำสั่ง **nfso** อาจทำให้ระบบของคุณไม่ทำงานได้

พารามิเตอร์ **nfso** และค่าของพารามิเตอร์สามารถแสดงขึ้นได้โดยใช้คำสั่ง **nfso -a** ดังนี้:

```
# nfso -a
                   portcheck = 0
                 udpchecksum = 1
              nfs_socketsize = 60000
          nfs_tcp_socketsize = 60000
           nfs_setattr_error = 0
         nfs_gather_threshold = 4096
          nfs_repeat_messages = 0
nfs_udp_duplicate_cache_size = 5000
nfs_tcp_duplicate_cache_size = 5000
```

```
    nfs_server_base_priority = 0
         nfs_dynamic_retrans = 1
            nfs_iopace_pages = 0
         nfs_max_connections = 0
             nfs_max_threads = 3891
      nfs_use_reserved_ports = 0
    nfs_device_specific_bufs = 1
           nfs_server_clread = 1
                nfs_rfc1323 = 1
          nfs_max_write_size = 65536
           nfs_max_read_size = 65536
       nfs_allow_all_signals = 0
                  nfs_v2_pdts = 1
                  nfs_v3_pdts = 1
              nfs_v2_vm_bufs = 1000
              nfs_v3_vm_bufs = 1000
   nfs_securenfs_authtimeout = 0
   nfs_v3_server_readdirplus = 1
           lockd_debug_level = 0
           statd_debug_level = 0
           statd_max_threads = 50
             utf8_validation = 1
                  nfs_v4_pdts = 1
              nfs_v4_vm_bufs = 1000
```

แอ็ตทริบิวต์ NFS ส่วนใหญ่เป็นแอ็ตทริบิวต์แบบรันไทม์ซึ่งสามารถเปลี่ยนแปลง ได้ตลอดเวลา แอ็ตทริบิวต์แบบเวลาโหลด เช่น *nfs_socketsize* ต้องหยุด NFS ก่อน แล้วรีสตาร์ทในภายหลัง คำสั่ง **nfso -L** ให้ข้อมูลรายละเอียดเพิ่มเติมเกี่ยวกับแต่ละแอ็ตทริบิวต์เหล่านี้ รวมถึง ค่าปัจจุบัน ค่าดีฟอลต์ และข้อจำกัดเกี่ยวกับเวลาที่การเปลี่ยนแปลงค่า มีผลบังคับใช้จริงดังนี้:

```
# nfso -L

NAME                      CUR    DEF    BOOT   MIN    MAX    UNIT           TYPE     DEPENDENCIES
--------------------------------------------------------------------------------
portcheck                 0      0      0      0      1      On/Off           D
--------------------------------------------------------------------------------
udpchecksum               1      1      1      0      1      On/Off           D
--------------------------------------------------------------------------------
nfs_socketsize            600000 600000 600000 40000  1M     Bytes            D
--------------------------------------------------------------------------------
nfs_tcp_socketsize        600000 600000 600000 40000  1M     Bytes            D
--------------------------------------------------------------------------------
nfs_setattr_error         0      0      0      0      1      On/Off           D
--------------------------------------------------------------------------------
nfs_gather_threshold      4K     4K     4K     512    8193   Bytes            D
--------------------------------------------------------------------------------
nfs_repeat_messages       0      0      0      0      1      On/Off           D
--------------------------------------------------------------------------------
nfs_udp_duplicate_cache_size
                          5000   5000   5000   5000   100000 Req              I
--------------------------------------------------------------------------------
nfs_tcp_duplicate_cache_size
                          5000   5000   5000   5000   100000 Req              I
--------------------------------------------------------------------------------
nfs_server_base_priority  0      0      0      31     125    Pri              D
--------------------------------------------------------------------------------
```

```
nfs_dynamic_retrans       1      1      1      0      1      On/Off           D
--------------------------------------------------------------------------------
nfs_iopace_pages          0      0      0      0      65535  Pages            D
--------------------------------------------------------------------------------
nfs_max_connections       0      0      0      0      10000  Number           D
--------------------------------------------------------------------------------
nfs_max_threads           3891   3891   3891   5      3891   Threads          D
--------------------------------------------------------------------------------
nfs_use_reserved_ports    0      0      0      0      1      On/Off           D
--------------------------------------------------------------------------------
nfs_device_specific_bufs  1      1      1      0      1      On/Off           D
--------------------------------------------------------------------------------
nfs_server_clread         1      1      1      0      1      On/Off           D
--------------------------------------------------------------------------------
nfs_rfc1323               1      1      0      0      1      On/Off           D
--------------------------------------------------------------------------------
nfs_max_write_size        64K    32K    32K    512    64K    Bytes            D
--------------------------------------------------------------------------------
nfs_max_read_size         64K    32K    32K    512    64K    Bytes            D
--------------------------------------------------------------------------------
nfs_allow_all_signals     0      0      0      0      1      On/Off           D
--------------------------------------------------------------------------------
nfs_v2_pdts               1      1      1      1      8      PDTs             M
--------------------------------------------------------------------------------
nfs_v3_pdts               1      1      1      1      8      PDTs             M
--------------------------------------------------------------------------------
nfs_v2_vm_bufs            1000   1000   1000   512    5000   Bufs             I
--------------------------------------------------------------------------------
nfs_v3_vm_bufs            1000   1000   1000   512    5000   Bufs             I
--------------------------------------------------------------------------------
nfs_securenfs_authtimeout 0      0      0      0      60     Seconds          D
--------------------------------------------------------------------------------
nfs_v3_server_readdirplus 1      1      1      0      1      On/Off           D
--------------------------------------------------------------------------------
lockd_debug_level         0      0      0      0      10     Level            D
--------------------------------------------------------------------------------
statd_debug_level         0      0      0      0      10     Level            D
--------------------------------------------------------------------------------
statd_max_threads         50     50     50     1      1000   Threads          D
--------------------------------------------------------------------------------
utf8_validation           1      1      1      0      1      On/Off           D
--------------------------------------------------------------------------------
nfs_v4_pdts               1      1      1      1      8      PDTs             M
--------------------------------------------------------------------------------
nfs_v4_vm_bufs            1000   1000   1000   512    5000   Bufs             I
--------------------------------------------------------------------------------


n/a means parameter not supported by the current platform or kernel

Parameter types:
    S = Static: cannot be changed
    D = Dynamic: can be freely changed
    B = Bosboot: can only be changed using bosboot and reboot
```

```
    R = Reboot: can only be changed during reboot
    C = Connect: changes are only effective for future socket connections
    M = Mount: changes are only effective for future mountings
    I = Incremental: can only be incremented

Value conventions:
    K = Kilo: 2^10       G = Giga: 2^30       P = Peta: 2^50
    M = Mega: 2^20       T = Tera: 2^40       E = Exa: 2^60
```

เมื่อต้องการแสดงหรือเปลี่ยนพารามิเตอร์เฉพาะ ให้ใช้คำสั่ง **nfso -o** ตัวอย่างเช่น:

```
# nfso -o portcheck
portcheck= 0
# nfso -o portcheck=1
```

พารามิเตอร์สามารถมีการรีเซ็ตเป็นค่าดีฟอลต์ได้โดยใช้อ็อพชัน **-d** ตัวอย่างเช่น:

```
# nfso -d portcheck
# nfso -o portcheck
portcheck= 0
```

คำแนะนำการปรับ TCP/IP สำหรับประสิทธิภาพ NFS

NFS ใช้ UDP หรือ TCP เพื่อดำเนินการ I/O เครือข่าย

ตรวจสอบให้แน่ใจว่าคุณได้ใช้เทคนิคการปรับตามที่อธิบายไว้ใน “การปรับประสิทธิภาพ TCP และ UDP” ในหน้า 282 และ “การปรับประสิทธิภาพพูล mbuf” ในหน้า 319 โดยเฉพาะอย่างยิ่ง คุณควรทำ ดังต่อไปนี้:

- ตรวจสอบรายการบันทึกข้อผิดพลาดระบบโดยการรันคำสั่ง **errpt** และค้นหารายงานปัญหาเกี่ยวกับอุปกรณ์เครือข่ายหรือสื่อเครือข่าย
- ตรวจสอบให้แน่ใจว่าคิวการส่งผ่านและการรับของอะแด็ปเตอร์ LAN มีการตั้งค่า เป็นค่าสูงสุด โปรดดู “การปรับรีซอร์สอะแด็ปเตอร์” ในหน้า 310 สำหรับ ข้อมูลเพิ่มเติม
- ตรวจสอบ `0errs` ด้วยคำสั่ง **netstat -i** ข้อผิดพลาดเหล่านี้จำนวนมากอาจบ่งชี้ว่าขนาดคิวการส่งผ่านของอุปกรณ์เครือข่ายที่เกี่ยวข้องไม่ใหญ่เพียงพอ
- ตรวจสอบให้แน่ใจว่าขนาดซ็อกเก็ตบัฟเฟอร์ TCP และ UDP มีการตั้งค่าคอนฟิกอย่างเหมาะสม *nfs_tcp_socketsize* ที่ปรับได้ของคำสั่ง **nfso** ควบคุมขนาดซ็อกเก็ตบัฟเฟอร์ TCP, *tcp_sendspace* และ *tcp_recvspace* ที่ใช้โดย NFS ในลักษณะคล้ายกัน *nfs_udp_socketsize* ที่ปรับได้ ควบคุมขนาดซ็อกเก็ตบัฟเฟอร์ UDP, *udp_sendspace* และ *udp_recvspace* ที่ใช้โดย NFS ปฏิบัติตามคำแนะนำที่อธิบายในการปรับประสิทธิภาพ TCP และ UDP สำหรับการตั้งค่าขนาดซ็อกเก็ตบัฟเฟอร์ที่ปรับได้ ตามการปรับ TCP และ UDP ปกติ ค่าของ **sb_max** ที่ปรับได้ของคำสั่ง **no** ต้องมากกว่าค่า *nfs_tcp_socketsize* และ *nfs_udp_socketsize* โดยทั่วไป คุณควรพบว่าค่าดีฟอลต์ที่ใช้ใน AIX ควรจะ สมบูรณ์ แต่ไม่เสียเวลาที่จะตรวจสอบค่านี้ เมื่อต้องการตรวจสอบ UDP ซ็อกเก็ต บัฟเฟอร์โอเวอร์รัน ให้รันคำสั่ง **netstat -s -p udp** และมองหาแพ็กเก็ตจำนวนมากที่ทิ้งซึ่งมีการรายงานในฟิลด์ ซ็อกเก็ต บัฟเฟอร์โอเวอร์โฟลว์
- ตรวจสอบให้แน่ใจว่ามีการตั้งค่าคอนฟิกหน่วยความจำเครือข่ายที่เพียงพอในระบบ รันคำสั่ง **netstat -m** และดูว่ามีการร้องขอสำหรับ mbufs ที่ปฏิเสธหรือล่าช้าหรือไม่ ถ้ามี ให้เพิ่มจำนวน ของ mbufs ที่มีอยู่บนเครือข่าย หากต้องการข้อมูลเพิ่มเติมเกี่ยวกับการปรับ ระบบเพื่อตัดปัญหา mbuf ให้ดู “การปรับประสิทธิภาพพูล mbuf” ในหน้า 319
- ตรวจสอบปัญหาการเราต์ทั่วไป ใช้คำสั่ง **traceroute** เพื่อค้นหา hops การเราต์ที่ไม่ได้คาดไว้หรือความล่าช้า

- ถ้าเป็นไปได้ให้เพิ่มขนาด MTU บน LAN ตัวอย่างเช่น บนเครือข่ายกิกะบิตอีเทอร์เน็ต 16 Mb การเพิ่มขนาด MTU จากค่าดีฟอลต์ 1500 ไบต์เป็น 9000 ไบต์ (jumbo frames) ช่วยให้สามารถส่งผ่านการร้องขอการอ่านหรือการ บันทึก NFS ขนาด 8 KB ได้โดยไม่มีการแบ่งเฟรกเมนต์ และยังให้ประสิทธิภาพดีกว่าการใช้ พื้นที่ว่าง mbuf เพื่อลดโอกาสของโอเวอร์รันเป็นอย่างมาก
- ตรวจสอบขนาด MTU ที่ไม่ตรงกัน รันคำสั่ง **netstat –i** และ ตรวจสอบ MTU บนไคลเอ็นต์และเซิร์ฟเวอร์ ถ้าขนาด MTU แตกต่างกัน ให้ลองทำให้เหมือนกัน และดูว่าแก้ไขปัญหาได้หรือไม่ นอกจากนี้ ควรตรวจสอบด้วยว่า อุปกรณ์เครือข่ายที่ช้าหรือครอบคลุมพื้นที่กว้าง เช่น เราเตอร์หรือบริดจ์ ระหว่าง เครื่องอาจแบ่งเฟรกเมนต์แพ็กเก็ตเพิ่มเติมเพื่อ traverse เซกเมนต์เครือข่ายหรือไม่ วิธีแก้ไขที่เป็นไปได้อย่างหนึ่ง คือ พยายามกำหนด MTU ที่เล็กที่สุดระหว่างต้นทางและปลายทาง และเปลี่ยนค่าติดตั้ง **rsize** และ **wsize** ในการติดตั้ง NFS เป็นจำนวน ที่ต่ำกว่า lowest–common–denominator MTU
- เมื่อรัน NFS เวอร์ชัน 3 ที่มี TCP และใช้ค่าดีฟอลต์ของ 32 KB หรือขนาด RPC ที่ใหญ่ขึ้น คุณควรจะตั้งค่าอ็อพชัน *nfs_rfc1323* ของคำสั่ง **nfso** ซึ่งช่วยให้ขนาดหน้าต่าง TCP ใหญ่กว่า 64 KB ส่งผลให้ลดเวลาการรอการตอบรับ TCP ต้องตั้งค่าอ็อพชัน บนแต่ละด้านของการเชื่อมต่อ TCP ตัวอย่างเช่น บนทั้งไคลเอ็นต์และเซิร์ฟเวอร์ NFS
- ตรวจสอบความล่าช้าระหว่างแพ็กเก็ตที่น้อยมาก นานๆ ครั้งที่ ความล่าช้านี้อาจทำให้เกิดปัญหาได้ ถ้ามีเราเตอร์หรือฮาร์ดแวร์อื่นอยู่ระหว่าง เซิร์ฟเวอร์และไคลเอ็นต์ คุณสามารถตรวจสอบส่วนเอกสารฮาร์ดแวร์เพื่อดูว่า สามารถตั้งค่าคอนฟิกความล่าช้าระหว่างแพ็กเก็ตได้หรือไม่ ถ้าทำได้ ให้ลองเพิ่มความล่าช้า
- ตรวจสอบความเร็วของสื่อขนาดใหญ่ที่ไม่ตรงกัน เมื่อแพ็กเก็ต traversing สองสื่อ ที่มีความเร็วแตกต่างกันมาก เราเตอร์อาจทิ้งแพ็กเก็ตเมื่อนำมาใช้บนเครือข่าย ความเร็วสูงและพยายามไล่แพ็กเก็ตไปบนเครือข่ายที่ช้ากว่า ตัวอย่างเช่น กรณีนี้อาจเกิดขึ้นเมื่อเราเตอร์พยายามนำแพ็กเก็ตออกจาก เซิร์ฟเวอร์บนกิกะบิตอีเทอร์เน็ตและส่งไปยังไคลเอ็นต์บนอีเทอร์เน็ต 100 Mbps อาจไม่สามารถส่งแพ็กเก็ตบนอีเทอร์เน็ต 100 Mbps ได้เร็วเพียงพอให้ทัน กับกิกะบิตอีเทอร์เน็ต นอกเหนือจากการเปลี่ยนเราเตอร์แล้ว วิธีแก้ไขที่เป็นไปได้ อีกอย่างหนึ่งคือ พยายามลดอัตราของการร้องขอไคลเอ็นต์ และ/หรือการใช้ขนาดการอ่าน/การบันทึกที่เล็กลง
- จำนวนสูงสุดของการเชื่อมต่อ TCP ที่ใช้ได้สำหรับเซิร์ฟเวอร์สามารถควบคุมโดย อ็อพชัน *nfs_max_connections* ใหม่ ค่าดีฟอลต์เป็น 0 บ่งชี้ว่าไม่มีขีดจำกัด ไคลเอ็นต์จะปิดการเชื่อมต่อ TCP ที่ idle นานประมาณ 5 นาที และสร้างการเชื่อมต่อใหม่เมื่อแน่ใจว่า มีการใช้งาน เซิร์ฟเวอร์จะปิดการเชื่อมต่อที่ idle นานประมาณ 6 นาที
- ระบบปฏิบัติการมีอ็อพชันในการปิด UDP checksum สำหรับ NFS เท่านั้น คุณสามารถใช้อ็อพชันคำสั่ง **nfso** ที่เรียกว่า *udpchecksum* ค่าดีฟอลต์คือ 1 ซึ่งหมายความว่า checksum มีการเปิดใช้งาน คุณสามารถได้รับประสิทธิภาพที่ดีขึ้น เล็กน้อยโดยการปิดค่านี้ โดยต้องแลกกับโอกาสการเสียหายของข้อมูลที่ เพิ่มมากขึ้น

แพ็กเก็ตที่ปล่อย

แม้ว่าแพ็กเก็ตที่ปล่อยจะถูกตรวจพบในครั้งแรกบนไคลเอ็นต์ NFS แต่ก็มีความท้าทายที่แท้จริงในการค้นหาแพ็กเก็ตที่หายไป แพ็กเก็ตอาจถูกปล่อยได้ที่ไคลเอ็นต์ เซิร์ฟเวอร์ หรือที่ใดๆ บนเน็ตเวิร์ก

แพ็กเก็ตที่ทิ้งโดยไคลเอ็นต์:

ไม่บ่อยนักที่แพ็กเก็ตจะถูกทิ้งโดยไคลเอ็นต์

เนื่องจากแต่ละเธรด **biod** สามารถทำงานบนการดำเนินงาน NFS ได้ทีละหนึ่งรายการเท่านั้น เธรดจึงต้องคอยการตอบการเรียก RPC จากการดำเนินงานนั้น ก่อนที่จะออกใช้การเรียก RPC อื่น กลไก self–pacing นี้หมายความว่ามีโอกาสเพียงเล็กน้อยเท่านั้น ที่รีซอร์สระบบจะเกิดการโอเวอร์รัน การดำเนินงานที่สร้างภาระมากที่สุด อาจจะเป็นการอ่าน เนื่องจากมีโอกาสที่อัตราข้อมูลจำนวนมากจะผ่านเข้าไป ในเครื่อง ในขณะที่ข้อมูลอาจมีปริมาณมาก แต่จำนวนที่แท้จริง ของการเรียก RPC พร้อมกันค่อนข้างต่ำ และแต่ละเธรด **biod** มีพื้นที่ว่างของตนเองที่จัดสรรสำหรับการตอบ ดังนั้น จึงเป็นเรื่องผิดปกติอย่างมากที่ ไคลเอ็นต์จะทิ้งแพ็กเก็ต

โดยทั่วไป แพ็กเก็ตมีโอกาสถูกทิ้งโดยเครือข่ายหรือโดยเซิร์ฟเวอร์มากกว่า

แพ็กเก็ตที่ทิ้งโดยเซิร์ฟเวอร์:

มีหลายสถานการณ์ที่เซิร์ฟเวอร์ทิ้งแพ็กเก็ตที่มีโหลด มาก

1. ไดรเวอร์อะแด็ปเตอร์เครือข่าย

   เมื่อเซิร์ฟเวอร์ NFS ตอบกลับการร้องขอจำนวนมาก บางครั้งเซิร์ฟเวอร์โอเวอร์รันคิวเอาต์พุตของอินเตอร์เฟส ไดรเวอร์ คุณสามารถสังเกตกรณีนี้ได้โดยดูที่สถิติซึ่งรายงานโดย คำสั่ง **netstat –i** ตรวจสอบคอลัมน์ที่ชื่อว่า `Oerrs` และ ดูจำนวนนับใดๆ แต่ละค่า `Oerrs` คือแพ็กเก็ตที่ทิ้ง กรณีนี้สามารถปรับ ได้ง่ายโดยการเพิ่มขนาดของคิวการส่งผ่านของไดรเวอร์ อุปกรณ์ที่มีปัญหา แนวคิดเบื้องหลังคิวที่ตั้งค่าคอนฟิกได้คือ คุณไม่ต้องให้คิวการส่งผ่านยาว เกินไป เนื่องจากเวลาแฝงที่เกิดขึ้นในการประมวลผลคิว แต่ เนื่องจาก NFS รักษาพอร์ตและ XID เดียวกันสำหรับการเรียก การเรียกที่สองจึง พอใจกับการตอบกลับไปยังการตอบของการเรียกที่หนึ่ง นอกจากนี้ เวลาแฝงของการจัดการคิว ยังน้อยกว่าเวลาแฝงของการส่งผ่าน UDP ใหม่ที่เกิดขึ้นจาก NFS ในกรณีที่ ทิ้งแพ็กเก็ตเป็นอย่างมาก
2. ซ็อกเก็ตบัฟเฟอร์

   UDP ซ็อกเก็ตบัฟเฟอร์เป็นอีกที่หนึ่งซึ่งเซิร์ฟเวอร์ ทิ้งแพ็กเก็ต แพ็กเก็ตที่ทิ้งเหล่านี้จะถูกนับโดยชั้น UDP และคุณสามารถ ดูสถิติได้โดยใช้คำสั่ง **netstat –p udp** ตรวจสอบสถิติ ซ็อกเก็ตบัฟเฟอร์โอเวอร์โฟลว์

   โดยปกติแล้ว แพ็กเก็ต NFS ถูกทิ้งที่ซ็อกเก็ตบัฟเฟอร์เฉพาะถ้าเซิร์ฟเวอร์มีการจราจรการบันทึก NFS จำนวนมาก เซิร์ฟเวอร์ NFS ใช้ซ็อกเก็ต UDP ที่แนบกับพอร์ต NFS 2049 และข้อมูลขาเข้าทั้งหมดจะมีการบัฟเฟอร์บนพอร์ต UDP นั้น ขนาดดีฟอลต์ของบัฟเฟอร์นี้ คือ 60,000 ไบต์ คุณสามารถหารจำนวนนั้นด้วยขนาดของแพ็กเก็ตการบันทึก NFS เวอร์ชัน 3 ดีฟอลต์ (32786) เพื่อให้ทราบว่าแพ็กเก็ตการบันทึกพร้อมกันจำนวน 19 รายการจะทำให้บัฟเฟอร์นั้นโอเวอร์โฟลว์ได้ คุณอาจเห็นกรณีที่ปรับเซิร์ฟเวอร์แล้ว และไม่มีแพ็กเก็ตที่ทิ้งสำหรับซ็อกเก็ตบัฟเฟอร์หรือ `Oerrs` ไดรเวอร์ แต่ไคลเอ็นต์ยังคงเจอกับไทม์เอาต์ และการส่งผ่านใหม่ นี่เป็นสถานการณ์จำลองที่ตีความได้สองอย่าง ถ้าเซิร์ฟเวอร์มีการ โหลดมาก เซิร์ฟเวอร์อาจเพียงแต่โอเวอร์โหลดและ backlog งานของ **nfsd** daemons บนเซิร์ฟเวอร์อาจส่งผลให้เวลาการตอบกลับมากเกินกว่าดีฟอลต์ไทม์เอาต์ที่ตั้งค่าบนไคลเอ็นต์ อีกกรณีหนึ่งที่อาจเป็นไปได้ และอาจเป็นปัญหาถ้าทราบว่าเซิร์ฟเวอร์จะ idle คือแพ็กเก็ตกำลังถูกทิ้งบนเครือข่าย

แพ็กเก็ตที่ทิ้งบนเครือข่าย:

ถ้าไม่มีการโอเวอร์โฟลว์ของซ็อกเก็ตบัฟเฟอร์หรือ `Oerrs` บน เซิร์ฟเวอร์ ไคลเอ็นต์จะได้รับไทม์เอาต์และการส่งผ่านใหม่จำนวนมาก และ เซิร์ฟเวอร์จะ idle ส่งผลให้แพ็กเก็ตมีโอกาสมากที่จะถูกทิ้งบนเครือข่าย

ในกรณีนี้ *เครือข่าย* อ้างอิงถึงอุปกรณ์หลากหลายประเภท ซึ่งรวมถึง สื่อบันทึกและอุปกรณ์เครือข่าย เช่น เราเตอร์ บริดจ์, concentrators, และชุด ทั้งหมดของอุปกรณ์ที่สามารถดำเนินการขนส่งแพ็กเก็ตระหว่าง ไคลเอ็นต์และเซิร์ฟเวอร์

ในทุกครั้งที่เซิร์ฟเวอร์ไม่โอเวอร์โหลดและไม่ได้ทิ้งแพ็กเก็ต แต่ประสิทธิภาพ NFS ไม่ดี ให้เข้าใจว่าแพ็กเก็ตกำลังจะถูกทิ้งบนเครือข่าย ต้องใช้ความพยายามอย่างมากเพื่อพิสูจน์เรื่องนี้และค้นหารายละเอียดว่าเครือข่ายจะ ทิ้งแพ็กเก็ตอย่างไร วิธีที่ง่ายที่สุดในการกำหนดปัญหาขึ้นอยู่กับ ความใกล้ชิดทางฟิสิคัลที่เกี่ยวข้องและรีซอร์สที่มีอยู่เป็นส่วนใหญ่

บางครั้ง เซิร์ฟเวอร์และไคลเอ็นต์อยู่ใกล้กันมากเพียงพอที่จะเชื่อมต่อกันโดยตรง โดยการส่งผ่านเซกเมนต์เครือข่ายใหญ่ขึ้นที่อาจทำให้เกิดปัญหาได้ สิ่งที่เห็นได้ชัดคือ ถ้าทำเช่นนี้และแก้ไขปัญหาได้ ตัวเครื่องเองก็จะ ไม่มีปัญหา อย่างไรก็ตาม บ่อยครั้งที่ไม่สามารถเดินสาย การเชื่อมต่อโดยตรงได้ และต้องติดตามแก้ไขปัญหา คุณสามารถใช้ sniffers เครือข่ายและเครื่องมืออื่นๆ เพื่อดีบักปัญหาดังกล่าว

คอนฟิกูเรชันระบบย่อยของดิสก์สำหรับผลการทำงาน NFS

หนึ่งในต้นทางของคอขวดทั่วไปในเวิร์กโหลดของการอ่าน/เขียน คือระบบย่อยของดิสก์ที่ปรับแต่งได้ไม่ดี

ขณะที่คุณอาจพิจารณาถึงการปรับระบบย่อยของดิสก์บนเซิร์ฟเวอร์ NFS เท่านั้น หมายเหตุ การตั้งค่าดิสก์ที่ปรับแต่งได้ไม่ดีบนไคลเอ็นต์ NFS อาจเป็นปัญหาที่แท้จริงใน สถานการณ์จำลองบางสถานการณ์ ตัวอย่างของเวิร์กโหลดนี้ซึ่งไฟล์ถูกคัดลอกโดย แอ็พพลิเคชันบนไคลเอ็นต์ NFS จากระบบไฟล์ที่ประกอบเข้ากับ NFS กับระบบไฟล์บนโลคัลบนไคลเอ็นต์ ในกรณีนี้ สิ่งสำคัญคือ ระบบย่อยของดิสก์บนไคลเอ็นต์ต้องถูกปรับไว้อย่างถูกต้อง ดังนั้น ผลการทำงานของการเขียนลงในระบบไฟล์บนโลคัลจะไม่กลายเป็นคอขวด โปรดดูเทคนิคการปรับ ดังที่ได้กล่าวไว้แล้วใน “ผลการทำงานของโลจิคัลวอลุ่มและดิสก์ I/O” ในหน้า 195 โดยเฉพาะอย่างยิ่ง ให้พิจารณาสิ่งต่อไปนี้:

- สำหรับเวิร์กโหลดการอ่านและการเขียนแบบปกติผ่าน NFS ให้ประเมินผลความสามารถของดิสก์ ซึ่งมีระบบไฟล์ที่ต้องการใช้ คุณสามารถทำสิ่งนี้ได้โดยการเขียนลง หรืออ่านจากไฟล์แบบโลคัลบนระบบไฟล์ คุณควรใช้คำสั่ง **iostat** เพื่อวัดค่าความสามารถของทรูพุต ของดิสก์ เนื่องจากการทดสอบแอ็พพลิเคชันอาจเสร็จสิ้นโดยไม่มีการเขียน ข้อมูลลงดิสก์ ตัวอย่างเช่น ข้อมูลบางส่วนยังคงอยู่ในหน่วยความจำ จากนั้น คุณสามารถพิจารณาทรูพุตที่ถูกวัดค่านี้ บนการเขียน/อ่านแบบโลคัลตามการโยงไว้ที่ส่วนบนของผลการทำงานที่คุณจะสามารถบรรลุเป้าหมายผ่าน NFS ได้ เนื่องจากคุณไม่ได้สร้างการประมวลผลเพิ่มเติม และเวลาแฝงที่ใช้เชื่อมโยงกับ NFS
- บ่อยครั้งที่มีความจำเป็นในการทำให้บรรลุเป้าหมายด้วยการเข้าถึงข้อมูลแบบขนาน การเข้าถึงแบบพร้อมกัน ในระบบไฟล์เดียวบนเซิร์ฟเวอร์โดยไคลเอ็นต์จำนวนมากหรือการประมวลผลไคลเอ็นต์จำนวนมาก อาจส่งผลทำให้ทรูพุตเกิดคอขวดบนดิสก์ I/O สำหรับอุปกรณ์ที่ระบุเฉพาะ คุณสามารถใช้คำสั่ง **iostat** เพื่อประเมินผลการโหลดดิสก์โดยเฉพาะอย่างยิ่ง พารามิเตอร์ *%tm_act* จะบ่งชี้เปอร์เซ็นต์ของเวลาที่ดิสก์เฉพาะแอ็คทีฟอยู่ แต่ค่าสูงสุดยังสามารถบ่งชี้ถึงดิสก์อะแด็ปเตอร์ที่เชื่อมโยงมีการโหลดมากเกินไป
- ขณะที่ไม่เกี่ยวข้องกับการปรับระบบย่อยของดิสก์โดยตรง จึงมีค่าที่จะกล่าวว่า การเขียนอย่างพร้อมเพียงกันลงในไฟล์เดี่ยวอาจส่งผลทำให้เกิด contention บน inode สำหรับการล็อกไฟล์ ระบบไฟล์ส่วนใหญ่จะใช้การล็อก inode เพื่อเข้าถึงไฟล์แบบอนุกรม และมั่นใจว่า ความสอดคล้องกันของข้อมูล จะถูกเขียนลงด้วยเช่นกัน น่าเสียดาย สิ่งนี้เป็นการหยุดผลการทำงานของการเขียนในกรณีที่ เธรดจำนวนมากกำลังพยายามเขียนลงไฟล์เดียวกันอย่างพร้อมเพียงกัน เนื่องจากเธรดที่พักการล็อก inode ไว้เท่านั้นที่จะถูกอนุญาตให้เขียนลงในไฟล์ ที่จุดเดี่ยวใดๆ ในเวลานั้น
- สำหรับเซิร์ฟเวอร์ NFS ขนาดใหญ่ กลยุทธ์ทั่วไปที่ควรแบ่งดิสก์ I/O ตามความต้องการระหว่างดิสก์จำนวนมากและอุปกรณ์ดิสก์อะแด็ปเตอร์เท่าที่จะทำได้ บนระบบที่มีดิสก์ I/O ซึ่งได้กระจายได้อย่างดีแล้ว สิ่งที่เป็นไปได้คือ การเข้าถึงจุดที่ CPU โหลดบนเซิร์ฟเวอร์ ซึ่งกลายเป็นปัจจัยที่จำกัดบนผลการทำงานของเวิร์กโหลด

การใช้ NFS ในทางที่ผิดซึ่งกระทบกับผลการทำงาน

การใช้ที่ผิดวิธีของ NFS จะเกิดขึ้นเนื่องจากผู้ใช้ไม่ได้จำแนกว่า ไฟล์ที่ผู้ใช้กำลังเข้าถึงอยู่ทีส่วนท้ายของพาธการสื่อสารที่มีราคาแพง

ตัวอย่างนี้มีดังต่อไปนี้:

- แอ็พพลิเคชันที่รันอยู่บนระบบที่ทำการอัพเดตแบบสุ่มของ NFS ที่ mount ระบบคลังสินค้า การสนับสนุนแอ็พพลิคชันการลงทะเบียนการค้าปลีก
- สภาวะแวดล้อมการพัฒนาที่ไดเร็กทอรีซอร์สโค้ดบนแต่ละระบบคือ NFS ที่ mount บนระบบอื่นๆ ทั้งหมดในสภาวะแวดล้อม ผู้พัฒนาจะล็อกออนเข้าสู่ระบบเพื่อทำการแก้ไขและคอมไพล์ สิ่งนี้ เป็นการรับประกันว่า การคอมไพล์ทั้งหมดจะได้รับซอร์สโค้ดมาจากที่ใด และเขียนเอาต์พุตไปยังที่ใดในระบบรีโมต
- การรันคำสั่ง **ld** บนระบบเดี่ยวจะแปลงสภาพไฟล์ `.o` ในไดเร็กทอรี NFS ที่ mount ลงในไฟล์ `a.out` ในไดเร็กทอรีเดียวกัน

- แอ็พพลิเคชันที่ออกคำสั่งเขียนซึ่งจะไม่จัดเรียงหน้า ตัวอย่างเช่น 10 KB การเขียนที่มีขนาดน้อยกว่า 4 KB จะส่งผลต่อ*pagein* และในกรณีของ NFS *pagein* จะผ่านเน็ตเวิร์ก

ซึ่งยังกล่าวได้ว่า นี่คือการใช้ที่มีความโปร่งใสที่ถูกต้อง ซึ่งจัดเตรียมไว้โดย NFS บางที เหตุการณ์นี้ก็เกิดขึ้นได้ แต่การใช้จะทำให้เกิดต้นทุนเวลาของตัวประมวลผลและ แบรนด์วิธของ LAN และลดระดับของเวลาตอบสนอง เมื่อคอนฟิกูเรชันของระบบเกี่ยวข้องกับการเข้าถึง NFS ซึ่งเป็นส่วนหนึ่งของรูปแบบการดำเนินการมาตรฐาน ผู้ออกแบบคอนฟิกูเรชันควรจัดเตรียมการป้องกันต้นทุนที่มีผลตามมากับการอ็อฟเซ็ตเชิงเทคนิค หรือผลประโยชน์ในเชิงธุรกิจ เช่น:

- การวางข้อมูลหรือซอร์สโค้ดทั้งหมดบนเซิร์ฟเวอร์ แทนที่จะเป็นเวิร์กสเตชัน ปรับปรุงการควบคุมซอร์สโค้ดและทำให้สำรองข้อมูลศูนย์กลางได้ง่ายขึ้น
- จำนวนของระบบที่ต่างกันจะเข้าถึงข้อมูลเดียวกัน สร้างเซิร์ฟเวอร์เฉพาะงานที่มีประสิทธิภาพมากกว่าระบบที่รวมบทบาทไคลเอ็นต์และเซิร์ฟเวอร์ เข้าด้วยกัน

ชนิดของแอ็พพลิเคชันอื่นๆ ที่ไม่ควรรันระหว่างระบบไฟล์ NFS คือแอ็พพลิเคชันที่ทำการเรียก **lockf( )** หรือ **flock( )** ที่บ่อยมาก ต่อวินาที สำหรับระบบไฟล์ NFS การเรียก **lockf( )** หรือ **flock( )** (และการเรียกการล็อกไฟล์อื่น) ต้องทำผ่าน **rpc.lockd** daemon นี่สามารถลดระดับผลการทำงานของระบบได้ เนื่องจาก lock daemon อาจไม่สามารถจัดการกับคำร้องขอล็อกจำนวนมาต่อวินาที

โดยไม่พิจารณาถึงผลการทำงานของความสามารถของไคลเอ็นต์และเซิร์ฟเวอร์ การดำเนินการล็อกทั้งหมดที่เกี่ยวข้องกับไฟล์ NFS จะช้าลงโดยไม่มีเหตุผล มีเหตุผลทางด้านเทคนิคอยู่หลายข้อ แต่ทั้งหมดจะอ้างอิงกับความเป็นจริงที่ว่า ถ้าไฟล์ถูกล็อก ข้อควรพิจารณาเป็นพิเศษถูกนำมาพิจารณา เพื่อมั่นใจว่า ไฟล์ถูกจัดการแบบซิงโครนัสบนฝั่งของการอ่านและเขียน นั่นหมายความว่า ไม่มีแคชของข้อมูลไฟล์ที่ไคลเอ็นต์ ซึ่งประกอบด้วยแอ็ตทริบิวต์ไฟล์ การดำเนินการกับไฟล์ทั้งหมดจะไปยังโหมดซิงโครนัสโดยไม่มีการแคช ความสับสนที่ว่า แอ็พพลิเคชันจะทำการล็อกไฟล์เน็ตเวิร์ก ถ้าเป็นการดำเนินการผ่าน NFS และแสดงผลการทำงานที่ไม่ดีเปรียบเทียบกับแอ็พพลิเคชันอื่น บนคู่ไคลเอ็นต์/เซิร์ฟเวอร์เดียวกัน

การมอนิเตอร์ประสิทธิภาพ NFS บนเซิร์ฟเวอร์

คุณควรตรวจสอบการใช้ประโยชน์ CPU, กิจกรรม I/O, และการใช้หน่วยความจำ โดยใช้คำสั่ง **vmstat** และ **iostat** บนเซิร์ฟเวอร์ NFS ในระหว่างกิจกรรมเวิร์กโหลด เพื่อดูว่าการตั้งค่าคอนฟิกตัวประมวลผล หน่วยความจำ และ I/O ของเซิร์ฟเวอร์สมบูรณ์หรือไม่

คุณสามารถใช้คำสั่ง **nfsstat** เพื่อมอนิเตอร์กิจกรรม การดำเนินงาน NFS บนเซิร์ฟเวอร์

คำสั่ง nfsstat -s

เซิร์ฟเวอร์ NFS แสดงจำนวนของการเรียก NFS ที่ได้รับ, การเรียก, และที่ปฏิเสธ, badcalls, เนื่องจากการพิสูจน์ตัวตน ตลอดจนจำนวน และเปอร์เซ็นต์ของการเรียกประเภทต่างๆ

ตัวอย่างต่อไปนี้แสดงส่วนเซิร์ฟเวอร์ของเอาต์พุตคำสั่ง **nfsstat** ที่ระบุโดยอ็อพชัน **–s**:

```
# nfsstat -s

Server rpc:
Connection oriented:
calls     badcalls  nullrecv badlen    xdrcall  dupchecks dupreqs
15835      0          0          0          0          772        0
Connectionless:
calls     badcalls  nullrecv badlen    xdrcall  dupchecks dupreqs
```

```
0          0          0          0          0          0          0

Server nfs:
calls      badcalls   public_v2  public_v3
15835      0          0          0
Version 2: (0 calls)
null       getattr    setattr    root       lookup     readlink   read
0 0%       0 0%       0 0%       0 0%       0 0%       0 0%       0 0%
wrcache    write      create     remove     rename     link       symlink
0 0%       0 0%       0 0%       0 0%       0 0%       0 0%       0 0%
mkdir      rmdir      readdir    statfs
0 0%       0 0%       0 0%       0 0%
Version 3: (15835 calls)
null       getattr    setattr    lookup     access     readlink   read
7 0%       3033 19%   55 0%      1008 6%    1542 9%    20 0%      9000 56%
write      create     mkdir      symlink    mknod      remove     rmdir
175 1%     185 1%     0 0%       0 0%       0 0%       120 0%     0 0%
rename     link       readdir    readdir+   fsstat     fsinfo     pathconf
87 0%      0 0%       1 0%       150 0%     348 2%     7 0%       0 0%
commit
97 0%
```

RPC เอาต์พุตสำหรับเซิร์ฟเวอร์, **-s**, เป็นดังนี้:

การเรียก
: จำนวนทั้งหมดของการเรียก RPC ที่ได้รับจากไคลเอ็นต์

badcalls
: จำนวนการเรียกทั้งหมดที่ปฏิเสธโดยชั้น RPC

nullrecv
: จำนวนครั้งที่การเรียก RPC ไม่มีอยู่เมื่อคาดว่าจะ ได้รับ

badlen แพ็กเก็ตที่ถูกตัดให้สั้นลงหรือเสียหาย (จำนวนของการเรียก RPC ที่มีความยาว น้อยกว่าการเรียก RPC ขนาดต่ำสุด)

xdrcall จำนวนของการเรียก RPC ที่มีส่วนหัวซึ่งไม่สามารถถอดรหัส External Data Representation (XDR) ได้

dupchecks
: จำนวนของการเรียก RPC ที่ค้นหาในแคชการร้องขอซ้ำ

dupreqs
: จำนวนของการเรียก RPC ซ้ำที่พบ

เอาต์พุตยังแสดงจำนวนของการเรียกประเภทต่างๆ และเปอร์เซ็นต์ของการเรียกประเภทนั้น

การตรวจสอบข้อมูลซ้ำมีการทำสำหรับการดำเนินงานที่ไม่สามารถทำสองครั้ง โดยมีผลลัพธ์เหมือนกัน ตัวอย่างที่นิยมใช้ตลอดกาลคือคำสั่ง **rm** คำสั่ง **rm** แรกจะสำเร็จ แต่ถ้าการตอบหายไป ไคลเอ็นต์จะส่งผ่านคำสั่งใหม่ เราต้องการให้การร้องขอซ้ำ เช่นนี้สำเร็จ ดังนั้นจึงใช้แคชข้อมูลซ้ำ และถ้านี่เป็น การร้องขอซ้ำ จะมีการส่งคืนผลลัพธ์เดียวกัน (สำเร็จ) บนการร้องขอซ้ำเมื่อการร้องขอนั้นถูกสร้างขึ้นบนการร้องขอแรกเริ่ม

โดยการดูเปอร์เซ็นต์ของการเรียกสำหรับการดำเนินงานประเภทต่างๆ เช่น **getattr( )**, **read( )**, **write( )**, หรือ **readdir( )** คุณสามารถตัดสินใจได้ว่าจะใช้การปรับชนิดใด ตัวอย่างเช่น ถ้าเปอร์เซ็นต์ของการเรียก **getattr( )** สูงมาก การปรับแอ็ตทริบิวต์

แคชอาจ มีประโยชน์ ถ้าเปอร์เซ็นต์ของการเรียก **write( )** สูงมาก การปรับ ดิสก์และ LVM เป็นสิ่งสำคัญ ถ้าเปอร์เซ็นต์ของการเรียก **read( )** สูงมาก การใช้ RAM มากขึ้นสำหรับการแคชไฟล์อาจช่วยให้ประสิทธิภาพดีขึ้น

การปรับประสิทธิภาพ NFS บนเซิร์ฟเวอร์

โดยหลักแล้ว ตัวแปรการปรับเฉพาะ NFS บนเซิร์ฟเวอร์สามารถเข้าถึงได้โดยใช้คำสั่ง **nfso**

โดยทั่วไป เมื่อดำเนินการอย่างเหมาะสม อ็อพชันการปรับเฉพาะ NFS สามารถช่วยในประเด็นดังต่อไปนี้:

- ลดโหลดบนเครือข่ายและบนเซิร์ฟเวอร์ NFS
- ปัญหาเกี่ยวกับงานบนเครือข่ายและการใช้หน่วยความจำของไคลเอ็นต์

จำนวนของเธรด nfsd ที่จำเป็น

มี **nfsd** daemon เดียวบนเซิร์ฟเวอร์ NFS ที่เป็นแบบ มัลติเธรด ซึ่งหมายความว่ามีหลายเคอร์นัลเธรดภายในกระบวนการ **nfsd** จำนวนของเธรดที่กำลังปรับตัวเองใน daemon เพื่อสร้างและทำลาย เธรดตามความจำเป็น ขึ้นอยู่กับ NFS โหลด

เนื่องจากความสามารถในการปรับตัวเองนี้ และเนื่องจากจำนวนดีฟอลต์ (3891) ของเธรด **nfsd** สูงสุดคือจำนวนสูงสุดที่ใช้ได้ จึงไม่จำเป็น ต้องเปลี่ยนค่านี้ อย่างไรก็ตาม คุณสามารถปรับจำนวนสูงสุดของเธรด **nfsd** ในระบบได้โดยใช้พารามิเตอร์ *nfs_max_threads* ของคำสั่ง **nfso**

ขีดจำกัดขนาดการอ่านและการบันทึกบนเซิร์ฟเวอร์

คุณสามารถใช้อ็อพชัน **nfs_max_read_size** และ **nfs_max_write_size** ของคำสั่ง **nfso** เพื่อควบคุมขนาดสูงสุดของ RPCs ที่ใช้สำหรับการตอบการอ่าน NFS และการร้องขอการบันทึก NFS ตามลำดับ

ส่วน “การปรับ NFS บนไคลเอ็นต์” ในหน้า 382 มีข้อมูลเกี่ยวกับสถานการณ์ซึ่งอาจเหมาะสมที่จะปรับขนาดการอ่านและการบันทึก RPC โดยปกติ การปรับดำเนินการบนไคลเอ็นต์ อย่างไรก็ตาม ในสภาพแวดล้อมที่การแก้ไขค่าเหล่านี้บนไคลเอ็นต์อาจจัดการได้ยาก อ็อพชัน **nfso** ของเซิร์ฟเวอร์อาจมีประโยชน์

ค่าสูงสุดของการแคชของการปรับข้อมูลไฟล์

NFS ไม่มีบัฟเฟอร์เฉพาะงานของตนเองสำหรับการแคชข้อมูลจากไฟล์ในระบบไฟล์ NFS ที่เอ็กซ์พอร์ต

Virtual Memory Manager (VMM) จะควบคุมการแคชของเพจไฟล์เหล่านี้ แทน ถ้าระบบทำหน้าที่เป็นเซิร์ฟเวอร์ NFS เฉพาะงาน ระบบจะเหมาะกับการอนุญาตให้ VMM ใช้หน่วยความจำมากเท่าที่จะมากได้สำหรับ ข้อมูลการแคช สำหรับเซิร์ฟเวอร์การเอ็กซ์พอร์ตระบบไฟล์ JFS การทำเช่นนี้จะถูกทำให้บรรลุเป้าหมาย ด้วยการตั้งค่าพารามิเตอร์ *maxperm* ซึ่งควบคุมเปอร์เซ็นต์สูงสุด ของหน่วยความจำที่ใช้โดยเพจไฟล์ JFS ถึง 100 เปอร์เซ็นต์ พารามิเตอร์นี้ จะตั้งค่าโดยใช้คำสั่ง **vmo** ตัวอย่างเช่น:

```
# vmo -o maxperm%=100
```

บนเซิร์ฟเวอร์การเอ็กซ์พอร์ตระบบไฟล์ JFS ที่ปรับปรุงแล้ว ทั้งพารามิเตอร์ *maxclient* และ *maxperm* ต้องถูกตั้งค่าไว้ พารามิเตอร์ *maxclient* จะควบคุมเปอร์เซ็นต์สูงสุดของหน่วยความจำที่ใช้โดยเพจเซ็กเมนต์ไคลเอ็นต์ ซึ่งเป็นตำแหน่งที่ข้อมูลไฟล์ JFS ที่ปรับปรุงแล้วถูกแคช หมายเหตุ ค่า *maxclient* ไม่สามารถมีค่าเกินกว่าค่าของ *maxperm* ตัวอย่างเช่น:

```
# vmo -o maxclient%=100
```

ภายใต้เงื่อนไขบางอย่าง ข้อมูลไฟล์ที่แคชมากเกินไปในหน่วยความจำ อาจไม่เป็นที่ต้องการ โปรดดู “ผลการทำงานของระบบไฟล์” ในหน้า 257 สำหรับคำอธิบายวิธีการที่คุณสามารถใช้กลไกที่เรียกว่า *release-behind* เพื่อล้างข้อมูลที่ไม่ได้นำกลับมาใช้โดยแอ็พพลิเคชัน

การปรับ RPC mount daemon

rpc.mountd daemon เป็นแบบหลายเธรดและโดยค่าดีฟอลต์ สามารถสร้างได้สูงสุด 16 เธรด

ในสภาพแวดล้อมที่มีการใช้งาน **automount** อย่างหนัก และเห็นไทม์เอาต์ของ **automount** daemon อยู่บ่อยครั้ง ควรเพิ่มจำนวนสูงสุดของเธรด **rpc.mountd** ดังนี้:

```
# chsys -s rpc.mountd -a -h <number of threads>
# stopsrc -s rpc.mountd
# startsrc -s rpc.mountd
```

การปรับ RPC lock daemon

rpc.lockd daemon เป็นแบบหลายเธรดและโดยค่าดีฟอลต์ สามารถสร้างได้สูงสุด 33 เธรด

ในสถานการณ์ที่มีกิจกรรมการล็อกไฟล์ RPC อย่างมาก **rpc.lockd** daemon อาจกลายเป็นปัญหาคอขวดเมื่อใช้งานจนถึงจำนวนเธรดสูงสุด เมื่อใช้งานถึงค่าสูงสุด การร้องขอในลำดับต่อมา ใดๆ ต้องรอ ซึ่งอาจส่งผลให้การร้องขออื่นไทม์เอาต์ได้ คุณสามารถปรับจำนวนของเธรด **rpc.lockd** ได้สูงสุดถึง 511 ข้อมูลต่อไปนี้เป็นตัวอย่าง:

```
# chsys -s rpc.lockd -a <number of threads>
# stopsrc -s rpc.lockd
# startsrc -s rpc.lockd
```

การมอนิเตอร์ประสิทธิภาพ NFS บนไคลเอ็นต์

คุณควรตรวจสอบการใช้ประโยชน์ CPU และการใช้หน่วยความจำโดยใช้คำสั่ง **vmstat** บนไคลเอ็นต์ NFS ในระหว่างกิจกรรมเวิร์กโหลด เพื่อดูว่าการตั้งค่าคอนฟิก ตัวประมวลผลและหน่วยความจำของไคลเอ็นต์สมบูรณ์หรือไม่

คุณสามารถใช้คำสั่ง **nfsstat** เพื่อมอนิเตอร์กิจกรรม การดำเนินงาน NFS โดยไคลเอ็นต์

คำสั่ง nfsstat -c

ไคลเอ็นต์ NFS แสดงจำนวนของการเรียก NFS ที่ส่งและถูกปฏิเสธ ตลอดจนจำนวนครั้งที่ได้รับการจัดการไคลเอ็นต์, `clgets`, และจำนวนของการเรียกประเภทต่างๆ และเปอร์เซ็นต์ของการเรียกประเภทนั้น

ตัวอย่างต่อไปนี้แสดงเอาต์พุตของคำสั่ง **nfsstat** ที่ระบุ สำหรับไคลเอ็นต์โดยใช้อ็อพชัน **–c**:

```
# nfsstat -c

Client rpc:
Connection oriented
calls     badcalls  badxids  timeouts  newcreds badverfs  timers
0          0          0          0          0          0          0
nomem     cantconn interrupts
0          0          0
Connectionless
calls     badcalls retrans   badxids   timeouts newcreds  badverfs
6553       0          0          0          0          0          0
timers    nomem     cantsend
0          0          0


Client nfs:
calls      badcalls   clgets     cltoomany
```

```
6541        0           0           0
Version 2: (6541 calls)
null        getattr     setattr     root        lookup      readlink    read
0 0%        590 9%      414 6%      0 0%        2308 35%    0 0%        0 0%
wrcache     write       create      remove      rename      link        symlink
0 0%        2482 37%    276 4%      277 4%      147 2%      0 0%        0 0%
mkdir       rmdir       readdir     statfs
6 0%        6 0%        30 0%       5 0%
Version 3: (0 calls)
null        getattr     setattr     lookup      access      readlink    read
0 0%        0 0%        0 0%        0 0%        0 0%        0 0%        0 0%
write       create      mkdir       symlink     mknod       remove      rmdir
0 0%        0 0%        0 0%        0 0%        0 0%        0 0%        0 0%
rename      link        readdir     readdir+    fsstat      fsinfo      pathconf
0 0%        0 0%        0 0%        0 0%        0 0%        0 0%        0 0%
commit
0 0%
```

RPC เอาต์พุตสำหรับไคลเอ็นต์, **-c**, เป็นดังนี้:

การเรียก
จำนวนทั้งหมดของการเรียก RPC ที่ดำเนินการไปยัง NFS

badcalls
จำนวนการเรียกทั้งหมดที่ปฏิเสธโดยชั้น RPC

retrans จำนวนครั้งที่ต้องส่งผ่านการเรียกใหม่เนื่องจากไทม์เอาต์ในขณะที่ กำลังรอการตอบจากเซิร์ฟเวอร์ ข้อมูลนี้ใช้เฉพาะกับ RPC บนการขนส่งที่ไม่กำหนด การเชื่อมต่อเท่านั้น

badxid จำนวนครั้งที่ได้รับการตอบจากเซิร์ฟเวอร์ซึ่งไม่สอดคล้องกับ การเรียกที่ค้างอยู่ใดๆ นี่หมายความว่าเซิร์ฟเวอร์ใช้เวลาในการตอบนานเกินไป

ไทม์เอาต์
จำนวนครั้งที่การเรียกไทม์เอาต์ในขณะที่กำลังรอการตอบจากเซิร์ฟเวอร์

newcreds
จำนวนครั้งที่ต้องรีเฟรชข้อมูลการพิสูจน์ตัวตน

badverfs
จำนวนครั้งที่การเรียกล้มเหลวเนื่องจากตัวตรวจสอบที่ไม่ดีในการตอบกลับ

timers จำนวนครั้งที่ค่าไทม์เอาต์ซึ่งคำนวณได้มากกว่าหรือเท่ากับ ค่าไทม์เอาต์ต่ำสุดที่ระบุสำหรับการเรียก

nomem จำนวนครั้งที่การเรียกล้มเหลวเนื่องจากความล้มเหลวในการจัดสรรหน่วยความจำ

cantconn
จำนวนครั้งที่การเรียกล้มเหลวเนื่องจากความล้มเหลวในการเชื่อมต่อไปยัง เซิร์ฟเวอร์

การขัดจังหวะ
จำนวนครั้งที่การเรียกถูกขัดจังหวะโดยสัญญาณรบกวนก่อนที่จะเสร็จสมบูรณ์

cantsend
จำนวนครั้งที่การส่งล้มเหลวเนื่องจากความล้มเหลวในการเชื่อมต่อไปยัง ไคลเอ็นต์

เอาต์พุตยังแสดงจำนวนของการเรียกประเภทต่างๆ และเปอร์เซ็นต์ของการเรียกประเภทนั้น

สำหรับการมอนิเตอร์ประสิทธิภาพ คำสั่ง **nfsstat –c** ให้ข้อมูลว่าเครือข่ายกำลังทิ้งแพ็กเก็ต UDP หรือไม่ เครือข่ายอาจ ทิ้งแพ็กเก็ตถ้าเครือข่ายไม่สามารถจัดการแพ็กเก็ตได้ แพ็กเก็ตที่ทิ้งอาจเป็นผลมาจาก เวลาตอบกลับของฮาร์ดแวร์หรือซอฟต์แวร์ เครือข่าย หรือ CPU ที่โอเวอร์โหลดบนเซิร์ฟเวอร์ แพ็กเก็ตที่ทิ้งไม่ได้สูญหายไปอย่างแท้จริง เนื่องจากมีการออก การร้องขอการแทนที่สำหรับแพ็กเก็ตนั้น

คอลัมน์ *retrans* ในส่วน RPC แสดงจำนวนครั้งที่ส่งผ่าน การร้องขอใหม่ เนื่องจากไทม์เอาต์ในขณะที่รอการตอบกลับ สถานการณ์นี้เกี่ยวข้องกับแพ็กเก็ต UDP ที่ทิ้งไป ถ้าค่า *retrans* เกินกว่า ห้าเปอร์เซ็นต์ของการเรียกทั้งหมดในคอลัมน์หนึ่งอย่างสม่ำเสมอ นั่นบ่งชี้ถึง ปัญหาเกี่ยวกับเซิร์ฟเวอร์ที่ไม่สามารถให้บริการได้เพียงพอต่อความต้องการ ใช้คำสั่ง **vmstat** และ **iostat** บนเครื่องเซิร์ฟเวอร์เพื่อตรวจสอบโหลด

จำนวน *badxid* สูงหมายความว่าการร้องขอ กำลังเข้าถึงเซิร์ฟเวอร์ NFS ต่างๆ แต่เซิร์ฟเวอร์มีโหลดมากเกินกว่าที่จะส่งการตอบก่อนไทม์เอาต์การเรียก RPC ของไคลเอ็นต์ และถูกส่งผ่านใหม่ ค่า *badxid* เพิ่มขึ้น ในแต่ละครั้งที่ได้รับการตอบซ้ำสำหรับการร้องขอที่ ส่งผ่าน การร้องขอ RPC รักษาค่า *XID* ของตนไว้ตลอด รอบการส่งผ่านทั้งหมด การส่งผ่านใหม่มากเกินไปทำให้เกิดภาระเพิ่มเติมบน เซิร์ฟเวอร์ และยังทำให้เวลาการตอบกลับแย่ลงด้วย ถ้าค่า *badxid* และ จำนวนไทม์เอาต์มากกว่าห้าเปอร์เซ็นต์ของการเรียกทั้งหมด ให้เพิ่มพารามิเตอร์ *timeo* ของอ็อพชันการติดตั้ง NFS โดยใช้คำสั่ง **smitty chnfsmnt** ถ้าค่า *badxid* เป็น 0 แต่ค่า *retrans* และจำนวนไทม์เอาต์มาก ให้พยายาม ลดขนาดของบัฟเฟอร์ NFS โดยใช้อ็อพชัน **rsize** และ **wsize** ของคำสั่ง **mount**

ถ้าจำนวนของการส่งผ่านใหม่และไทม์เอาต์ใกล้เคียงเกือบเป็นค่า เดียวกัน แพ็กเก็ตจะถูกทิ้งอย่างแน่นอน โปรดดู “แพ็กเก็ตที่ปล่อย” ในหน้า 373 สำหรับ คำอธิบายเพิ่มเติม

ในบางอินสแตนซ์ แอ็พพลิเคชันหรือผู้ใช้ได้รับประสิทธิภาพที่ไม่ดี แต่การตรวจสอบของเอาต์พุตคำสั่ง **nfsstat –c** บ่งชี้ว่า ไม่มีไทม์เอาต์และการส่งผ่านใหม่ หรือมีอยู่น้อยมาก ซึ่งหมายความว่าไคลเอ็นต์ กำลังได้รับการตอบกลับจากเซิร์ฟเวอร์เร็วมาก เกือบจะในทันทีที่สอบถามไป สิ่งแรก ที่จะตรวจสอบคือมี **biod** daemons จำนวนที่เหมาะสม กำลังรันอยู่บนเครื่องไคลเอ็นต์หรือไม่ ซึ่งยังสามารถสังเกตได้เมื่อแอ็พพลิเคชัน กำลังทำการล็อกไฟล์แบบรีโมต เมื่อมีการตั้งค่าล็อกไฟล์แบบรีโมตบนไฟล์ที่อยู่บน NFS ไคลเอ็นต์จะเข้าสู่โหมดซิงโครนัสแบบเต็มรูปแบบของการดำเนินงานที่ จะปิดการแคชข้อมูลและแอ็ตทริบิวต์ทั้งหมดสำหรับไฟล์ ผลลัพธ์คือประสิทธิภาพ ที่ช้ามาก และโชคไม่ดีที่ถือเป็นเรื่องปกติ คุณสามารถระบุแพ็กเก็ตที่ล็อกอยู่ในเอาต์พุต **ipreport** ได้โดยค้นหาการร้องขอ NLM

คำสั่ง nfsstat -m

คำสั่ง **nfsstat –m** แสดง ชื่อเซิร์ฟเวอร์และที่อยู่ แฟล็กติดตั้ง ขนาดการอ่านและการบันทึกปัจจุบัน จำนวน การส่งผ่านใหม่ และไทม์เมอร์ที่ใช้สำหรับการส่งผ่านใหม่แบบไดนามิกของแต่ละการติดตั้ง NFS บนไคลเอ็นต์

ข้อมูลต่อไปนี้เป็นตัวอย่าง:

```
# nfsstat -m
/SAVE from /SAVE:aixhost.ibm.com
 Flags:   vers=2,proto=udp,auth=unix,soft,intr,dynamic,rsize=8192,wsize=8192,retrans=5
 Lookups: srtt=27 (67ms), dev=17 (85ms), cur=11 (220ms)
 Reads:   srtt=16 (40ms), dev=7 (35ms), cur=5 (100ms)
 Writes:  srtt=42 (105ms), dev=14 (70ms), cur=12 (240ms)
 All:     srtt=27 (67ms), dev=17 (85ms), cur=11 (220ms)
```

ตัวเลขในวงเล็บในเอาต์พุตตัวอย่างคือเวลาจริงในหน่วย มิลลิวินาที ค่าอื่นคือค่าที่ไม่ได้ปรับสเกลซึ่งเก็บไว้โดยเคอร์นัลระบบปฏิบัติการ คุณสามารถละเว้นค่าที่ไม่ได้ปรับสเกลได้ เวลาตอบกลับแสดงขึ้น สำหรับการค้นหา การอ่าน การบันทึก และชุดของการดำเนินงานเหล่านี้ทั้งหมด ทั้งหมด คำนิยามอื่นที่ใช้ในเอาต์พุตนี้มีดังนี้:

srtt เวลาเดินทางไปกลับตามปกติ

dev การเบี่ยงเบนที่ประเมินไว้

cur ค่าไทม์เอาต์ backed-off ปัจจุบัน

การปรับ NFS บนไคลเอ็นต์

โดยหลักแล้ว ตัวแปรการปรับเฉพาะ NFS สามารถเข้าถึงได้โดยใช้คำสั่ง **nfso** และ **mount**

ก่อนคุณเริ่มต้นปรับค่าของตัวแปรการปรับ ต้องแน่ใจว่า คุณเข้าใจว่าคุณกำลังพยายามให้เกิดผลสำเร็จอะไรโดยการแก้ไขค่าเหล่านี้ และผลกระทบเชิงลบที่อาจเกิดขึ้นได้จากการเปลี่ยนแปลงเหล่านี้คืออะไร

คุณยังสามารถตั้งค่าอ็อพชัน **mount** โดยการแก้ไข `/etc/filesystems` stanza สำหรับระบบไฟล์เฉพาะ เพื่อให้ค่ามีผลบังคับใช้เมื่อบูตระบบไฟล์ที่ติดตั้ง

โดยทั่วไป เมื่อดำเนินการอย่างเหมาะสม อ็อพชันการปรับเฉพาะ NFS สามารถช่วยในประเด็นดังต่อไปนี้:

- ลดโหลดบนเครือข่ายและบนเซิร์ฟเวอร์ NFS
- ปัญหาเกี่ยวกับงานบนเครือข่ายและการใช้หน่วยความจำของไคลเอ็นต์

จำนวนของเธรด biod ที่จำเป็น

มี **biod** daemon เดียวบนไคลเอ็นต์ NFS ที่เป็นแบบ มัลติเธรด ซึ่งหมายความว่ามีหลายเคอร์นัลเธรดภายในกระบวนการ **biod** จำนวนของเธรดที่กำลังปรับตัวเองใน daemon เพื่อสร้างและทำลาย เธรดตามความจำเป็น ขึ้นอยู่กับ NFS โหลด

คุณสามารถปรับจำนวนสูงสุดของเธรด **biod** ต่อการติดตั้งได้โดยใช้อ็อพชัน การติดตั้ง **biod** จำนวนดีฟอลต์ของเธรด **biod** คือ 4 สำหรับการติดตั้ง NFS เวอร์ชัน 3 และ NFS เวอร์ชัน 4 และเป็น 7 สำหรับการติดตั้ง NFS เวอร์ชัน 2

เนื่องจากเธรด **biod** จัดการร้องขอการอ่านหรือการบันทึกครั้งละหนึ่งรายการ และเนื่องจากเวลาการตอบกลับ NFS มักเป็นส่วนประกอบที่ใหญ่ที่สุดของเวลาการตอบกลับ โดยรวม ดังนั้นจึงไม่ควรบล็อกแอ็พพลิเคชันให้ขาดแคลนเธรด **biod**

การกำหนดจำนวนที่ดีที่สุดของ **nfsd** และ **biod** daemons คือกระบวนการวนซ้ำ คำแนะนำที่แสดงรายการด้านล่างเป็นเพียงจุดเริ่มต้นที่เหมาะสม เท่านั้น ข้อควรพิจารณาทั่วไปสำหรับการตั้งค่าคอนฟิกเธรด **biod** มีดังนี้:

- การเพิ่มจำนวนเธรดไม่สามารถชดเชยไคลเอ็นต์ที่ไม่สมบูรณ์ หรือกำลังตัวประมวลผลหรือหน่วยความจำเซิร์ฟเวอร์ หรือแบนด์วิธของเซิร์ฟเวอร์ดิสก์ที่ไม่สมบูรณ์ ก่อนการเปลี่ยนจำนวนเธรด คุณควรตรวจสอบระดับการใช้ประโยชน์รีซอร์สของเซิร์ฟเวอร์และไคลเอ็นต์โดยใช้คำสั่ง **iostat** และ **vmstat**
- ถ้า CPU หรือระบบย่อยดิสก์ใกล้เต็มแล้ว การเพิ่มจำนวนเธรด จะไม่ทำให้ประสิทธิภาพดีขึ้น
- เฉพาะการอ่านและการบันทึกเท่านั้นที่ดำเนินการผ่านทางเธรด **biod**
- โดยทั่วไป ค่าดีฟอลต์นับเป็นจุดเริ่มต้นที่ดี แต่การเพิ่มจำนวนของเธรด **biod** สำหรับจุดติดตั้งอาจเป็นสิ่งที่ควรทำ ถ้าหลายแอ็พพลิเคชันเธรด กำลังเข้าถึงไฟล์บนจุดติดตั้งนั้นพร้อมกัน ตัวอย่างเช่น คุณอาจต้องการประเมินจำนวนไฟล์ที่จะถูกบันทึกพร้อมกัน ตรวจสอบให้แน่ใจว่าคุณมีเธรด **biod** อย่างน้อยสองรายการต่อไฟล์ เพื่อสนับสนุนกิจกรรม read ahead หรือ write behind

- ถ้าคุณมีไคลเอ็นต์เวิร์กสเตชันที่รวดเร็วเชื่อมต่ออยู่กับเซิร์ฟเวอร์ที่ช้า คุณ อาจต้องจำกัดอัตราการสร้างการร้องขอ NFS ของไคลเอ็นต์โซลูชันที่อาจเป็นไปได้คือการลดจำนวนของเธรด **biod** บนไคลเอ็นต์ การให้ความสนใจกับความสำคัญจากการเปรียบเทียบของแต่ละเวิร์กโหลดของไคลเอ็นต์ และความต้องการเวลาการตอบกลับ การเพิ่มจำนวนของเธรด **biod** บนไคลเอ็นต์ส่งผลกระทบด้านลบต่อประสิทธิภาพเซิร์ฟเวอร์ เนื่องจากทำให้ไคลเอ็นต์สามารถส่งการร้องขอได้หลายรายการพร้อมกัน ซึ่งเพิ่มการโหลดเครือข่าย และเซิร์ฟเวอร์ ในกรณีที่ไคลเอ็นต์โอเวอร์รันเซิร์ฟเวอร์ อาจจำเป็นต้อง ลดจำนวนของเธรด **biod** เป็นหนึ่ง ตัวอย่างเช่น:

  ```
  # stopsrc -s biod
  ```

  ตัวอย่างข้างบนปล่อยให้ไคลเอ็นต์ที่มีกระบวนการ **biod** เคอร์นัล เพียงอย่างเดียวยังคงรันต่อไป

การปรับขนาดการอ่านและการบันทึก

บางอ็อพชันการปรับ NFS ที่มีประโยชน์มากที่สุดคืออ็อพชัน **rsize** และ **wsize** ซึ่งกำหนดขนาดสูงสุดของแต่ละแพ็กเก็ต RPC สำหรับการอ่านและการบันทึก ตามลำดับ

เหตุผลต่อไปนี้เป็นคำตอบว่าเพราะเหตุใดคุณจึงอาจต้องการเปลี่ยนค่าขนาด การอ่านและการบันทึก:

- เซิร์ฟเวอร์อาจไม่สามารถจัดการข้อมูลในปริมาณและความเร็วที่ได้รับมาใน การโอนย้ายแพ็กเก็ตการอ่าน/การบันทึก ซึ่งเป็น 8 KB สำหรับ NFS เวอร์ชัน 2 และ 32 KB สำหรับ NFS เวอร์ชัน 3 และ NFS เวอร์ชัน 4 กรณีเช่นนี้อาจเกิด ขึ้นถ้าไคลเอ็นต์ NFS กำลังใช้เครื่องพีซีเป็นเซิร์ฟเวอร์ NFS เครื่องพีซีอาจมีหน่วยความจำที่ใช้ได้จำกัด สำหรับการบัฟเฟอร์แพ็กเก็ตขนาดใหญ่
- ถ้าค่าขนาดการอ่าน/การบันทึกลดลง อาจส่งผลให้ลดจำนวนของ IP fragments ที่สร้างขึ้นโดยการเรียกในเวลาต่อมา ถ้าคุณกำลังจัดการกับเครือข่าย ที่บกพร่อง โอกาสที่คู่ของการตอบและการเรียกที่ต้องการการแลกเปลี่ยนสองแพ็กเก็ต มีมากกว่ากรณีที่ต้องมีการแลกเปลี่ยนเจ็ดแพ็กเก็ตที่เสร็จเรียบร้อย ในลักษณะคล้ายกัน ถ้าคุณกำลังส่งแพ็กเก็ต NFS บนหลายเครือข่ายที่มีลักษณะ ประสิทธิภาพแตกต่างกัน แพ็กเก็ตเฟรกเมนต์อาจมาถึงไม่ครบทั้งหมดก่อนค่า ไทม์เอาต์ของ IP fragments

การลดค่า **rsize** และ **wsize** อาจช่วยปรับปรุงประสิทธิภาพ NFS ในเครือข่ายที่คับคั่งโดยการส่งแพ็กเก็ตที่สั้นลงสำหรับแต่ละการตอบการอ่าน NFS และการร้องขอการบันทึก แต่ผลข้างเคียงของการทำเช่นนี้คือต้องการแพ็กเก็ตจำนวนมากขึ้น ในการส่งข้อมูลบนเครือข่าย ส่งผลให้การจราจรเครือข่ายทั้งหมดเพิ่มขึ้น และทำให้การใช้ประโยชน์ CPU บนทั้งเซิร์ฟเวอร์และไคลเอ็นต์มากขึ้น

ถ้าระบบไฟล์ NFS ของคุณติดตั้งบนเครือข่ายความเร็วสูง เช่น กิกะบิตอีเทอร์เน็ต ขนาดแพ็กเก็ตการอ่านและการบันทึกที่ใหญ่ขึ้นอาจช่วยให้ประสิทธิภาพของระบบ ไฟล์ NFS พัฒนาขึ้น ด้วย NFS เวอร์ชัน 3 และ NFS เวอร์ชัน 4 คุณสามารถตั้งค่า **rsize** และ **wsize** ได้สูงถึง 65536 เมื่อการขนส่งเครือข่ายคือ TCP ค่าดีฟอลต์คือ 32768 ด้วย NFS เวอร์ชัน 2 ขนาดสูงสุดสำหรับอ็อพชัน **rsize** และ **wsize** คือ 8192 ซึ่งเป็นค่าดีฟอลต์ด้วย

การปรับการแคชของข้อมูลไฟล์ NFS

VMM ควบคุมการแคชของข้อมูลไฟล์ NFS บนไคลเอ็นต์ NFS ในหน้าไคลเอ็นต์เซกเมนต์

ถ้าไคลเอ็นต์ NFS กำลังรันเวิร์กโหลดที่มีความต้องการหน้าเซกเมนต์ การทำงานน้อย อาจเหมาะสมที่จะอนุญาตให้ VMM ใช้หน่วยความจำระบบ ได้มากเท่าที่มีอยู่สำหรับการแคชข้อมูลไฟล์ NFS คุณสามารถทำเช่นนี้ได้โดยการตั้งค่า ทั้งพารามิเตอร์ *maxperm* และ *maxclient* ค่าของ *maxclient* ต้องน้อยกว่าหรือเท่ากับ ค่าของ *maxperm* ตัวอย่างต่อไปนี้ ตั้งค่าจำนวนของหน่วยความจำที่มีอยู่สำหรับการแคชไฟล์เป็น 100%:

```
# vmo -o maxperm%=100
# vmo -o maxclient%=100
```

ผลกระทบของการทำแคชข้อมูล NFS สำหรับการอ่านทรูพุต:

การอ่านทรูพุตตามลำดับ NFS ตามที่ได้วัดค่าที่ไคลเอ็นต์ จะถูกปรับปรุงผ่านการอ่าน VMM และกลไกการทำแคช

การอ่านอนุญาตให้ข้อมูลไฟล์ถูกถ่ายโอนไปยังไคลเอ็นต์จากเซิร์ฟเวอร์ NFS ในการทำงานร่วมกับข้อมูลที่ร้องขอโดยไคลเอ็นต์แอ็พพลิเคชัน NFS ด้วยเวลาที่คำร้องขอสำหรับข้อมูลถูกออกคำสั่งโดยแอ็พพลิเคชัน จึงเป็นไปได้ที่ข้อมูลจะอยู่ในหน่วยความของไคลเอ็นต์ และคำร้องขอ สามารถตอบสนองได้โดยทันที การทำแคช VMM อนุญาตให้อ่านข้อมูลไฟล์ได้อีกครั้ง สมมุติว่า ข้อมูลไม่ได้ถูกเพจออกจากหน่วยความจำไคลเอ็นต์ ซึ่งจะเรียกคืนข้อมูลอีกครั้งจากเซิร์ฟเวอร์ NFS

ขณะที่แอ็พพลิเคชันจำนวนมากอาจได้รับผลประโยชน์จากการทำแคชจาก VMM ของข้อมูล NFS บนไคลเอ็นต์ จึงมีแอ็พพลิเคชันบางตัว เช่น ฐานข้อมูล ที่ออ่านดำเนินการกับ การจัดการแคชข้อมูลไฟล์ของตนเอง แอ็พพลิเคชันที่ดำเนินการจัดการกับแคชข้อมูลไฟล์ของตนเอง อาจได้รับประโยชน์จากการใช้ I/O โดยตรง หรือ DIO ผ่าน NFS คุณสามารถเปิดใช้งาน DIO ผ่าน NFS ด้วยอ็อพชัน **dio** ของคำสั่ง **mount** หรือโดยระบบแฟล็ก **O_DIRECT** ด้วยการเรียกของระบบ **open( )**

ต่อไปนี้จะแสดงรายละเอียดผลประโยชน์ของ DIO:

- คุณหลีกเลี่ยงการแคชซ้ำของข้อมูลไฟล์โดย VMM และแอ็พพลิเคชัน
- คุณสามารถรับ CPU ที่มีประสิทธิผลสำหรับการอ่านและเขียนไฟล์ เนื่องจากฟังก์ชัน DIO ส่งผ่านโค้ด VMM

แอ็พพลิเคชันที่ดำเนินการกับการจัดการแคชข้อมูลไฟล์ของตนเอง และการเข้าถึงไฟล์แบบการทำให้เป็นอนุกรม เช่น ฐานข้อมูล อาจได้รับประโยชน์จากการใช้ I/O หรือ CIO ที่พร้อมเพียงกัน นอกจากผลประโยชน์ของ DIO, CIO จะไม่ทำให้เป็นอนุกรมสำหรับการเข้าถึงไฟล์เพื่อ อ่านและเขียน ซึ่งอนุญาตให้เธรดจำนวนมากอ่านหรือเขียน ไฟล์เดียวกันอย่างพร้อมเพียงกัน

หมายเหตุ: การใช้ CIO หรือ DIO อาจลดระดับผลการทำงานสำหรับ แอ็พพลิเคชันที่อ้างอิงการทำแคชของไฟล์ VMM และการ optimization VMM แบบ read-ahead และ write-behind สำหรับผลการทำงานของระบบที่เพิ่มขึ้น

คุณสามารถใช้ CacheFS เพื่อปรับปรุงการอ่านทรูพุตในสภาวะแวดล้อม ที่มีไคลเอ็นต์ที่จำกัดหน่วยความจำ ไฟล์ขนาดใหญ่มาก และ/หรือเซ็กเมนต์เน็ตเวิร์กที่ช้า โดยเพิ่มการตอบสนองคำร้องขอการอ่านจากข้อมูลไฟล์ที่อยู่ใน แคชของโลคัลดิสก์บนไคลเอ็นต์ โปรดดู “ระบบไฟล์แคช” ในหน้า 387 สำหรับข้อมูลเพิ่มเติม

การทำข้อมูลแคชสำหรับการอ่านตามลำดับของไฟล์ที่มีขนาดใหญ่ ซึ่งส่งผลถึงกิจกรรมการแทนที่เพจตามหน่วยความจำที่กรอกด้วยแคชข้อมูล NFS คุณสามารถปรับปรุงประสิทธิภาพโดยหลีกเลี่ยงกิจกรรม การแทนที่เพจโดยใช้ release-behind สำหรับการอ่าน, **rbr**, อ็อพชัน **mount** หรืออาร์กิวเมนต์ **nfs4cl setfsoptions** สำหรับ NFS เวอร์ชัน 4 สำหรับ การอ่านตามลำดับของไฟล์ขนาดใหญ่ หน่วยความจำจริงสำหรับการอ่านก่อนหน้านี้ จะเป็นอิสระตามการอ่านตามลำดับต่อไป

ถ้าอ็อพชัน **rbr mount** เริ่มต้นปล่อยหน่วยความจำที่คุณกำลังต้องการอีกครั้งในไม่ช้า คุณสามารถใช้ **nfs_auto_rbr_trigger** ที่สามารถปรับแต่งได้ของ คำสั่ง **nfso** แทน **nfs_auto_rbr_trigger** ที่สามารถปรับแต่งได้ซึ่งจะวัดค่าในหน่วยเมกะไบต์ เพื่อใช้ threshold อ็อฟเซ็ตการอ่าน สำหรับ release-behind บนอ็อพชันการอ่านที่มีผลบังคับใช้ ตัวอย่างเช่น ถ้า **nfs_auto_rbr_trigger** สามารถปรับแต่งได้มีค่า 100 MB ค่าแรก 100 MB ของไฟล์การอ่านตามลำดับจะถูกแคช และที่ส่วนเหลือของไฟล์จะถูกปล่อยจากหน่วยความจำ

ผลกระทบของการทำแคชข้อมูล NFS สำหรับการเขียนทรูพุต:

ถ้าคุณกำลังพยายามดำเนินการกับการดำเนินการเขียนลำดับบนไฟล์โดยใช้ NFS เวอร์ชัน 3 หรือ NFS เวอร์ชัน 4 ที่มีขนาดใหญ่กว่าหน่วยความไคลเอ็นต์ คุณสามารถปรับปรุงผลการทำงานโดยใช้ commit-behind

การเขียนไฟล์ทั้งหมดที่มีขนาดใหญ่เกินกว่าจำนวนของหน่วยความจำในไคลเอ็นต์ ส่งผลทำให้เกิดกิจกรรมการแทนที่เพจขนาดใหญ่บนไคลเอ็นต์ ซึ่งอาจส่งผลทำให้การดำเนินการ commit ถูกดำเนินการแบบ over-the-wire สำหรับเพจของข้อมูลที่เขียนทุกเพจ การ commit-behind เปิดใช้งานตรรกะสำหรับการ commit ไคลเอ็นต์เพจ กับหน่วยความจำที่มั่นคงบนเซิร์ฟเวอร์ และส่งคืนเพจเหล่านั้น กลับสู่รายการอิสระ

คุณสามารถเปิดใช้งาน commit-behind เมื่อประกอบเข้ากับระบบไฟล์โดยระบุอ็อพชัน **combehind** ด้วยคำสั่ง **mount** คุณยังต้องการตั้งค่าที่เหมาะสมสำหรับตัวแปร *numclust* ที่มีคำสั่ง **mount** ตัวแปรนี้ระบุจำนวนของคลัสเตอร์ขนาด 16 KB ที่ประมวลผลโดยอัลกอริธึม write-behind ของ Virtual Memory Manager (VMM) เมื่อรูปแบบ I/O เป็นลำดับ ให้ใช้ค่าที่มีขนาดใหญ่สำหรับอ็อพชัน **numclust** หากต้องการเก็บเพจเพิ่มเติมใน RAM ก่อนที่จะกำหนดตารางเวลาสำหรับ I/O ให้เพิ่มค่าสำหรับอ็อพชัน **numclust** ถ้าโลจิคัลวอลุ่มที่ strip หรือดิสก์อาร์เรย์ถูกนำมาใช้

การปรับแคชของแอ็ตทริบิวต์ไฟล์ NFS

NFS เก็บรักษาแคชไว้บนแต่ละระบบไคลเอ็นต์ของแอ็ตทริบิวต์ของ ไดเร็กทอรีและไฟล์ที่เข้าถึงล่าสุด

คุณสามารถตั้งค่าพารามิเตอร์หลายตัวโดยใช้คำสั่ง **mount** เพื่อควบคุมว่าจะเก็บรายการที่กำหนดไว้ในแคชนานเพียงใด พารามิเตอร์ที่กล่าวถึงมีดังนี้:

actimeo
: เวลาสมบูรณ์สำหรับการเก็บรายการไฟล์และไดเร็กทอรีไว้ในแคชแอ็ตทริบิวต์ไฟล์ หลังจากการอัพเดต ถ้าระบุ ค่านี้จะแทนที่ค่า **min* และ **max* ต่อไปนี้ เพื่อตั้งค่าเหล่านั้นทั้งหมดเป็นค่า **actimeo**

acregmin
: เวลาต่ำสุดหลังจากการอัพเดตที่จะเก็บรักษารายการไฟล์ไว้ ค่าดีฟอลต์ คือ 3 วินาที

acregmax
: เวลาสูงสุดหลังจากการอัพเดตที่จะเก็บรักษารายการไฟล์ไว้ ค่าดีฟอลต์ คือ 60 วินาที

acdirmin
: เวลาต่ำสุดหลังจากการอัพเดตที่จะเก็บรักษารายการไดเร็กทอรีไว้ ค่าดีฟอลต์คือ 30 วินาที

acdirmax
: เวลาสูงสุดหลังจากการอัพเดตที่จะเก็บรักษารายการไดเร็กทอรีไว้ ค่าดีฟอลต์คือ 60 วินาที

ในแต่ละครั้งที่อัพเดตไฟล์หรือไดเร็กทอรี การลบไฟล์หรือไดเร็กทอรีนั้นจะถูกเลื่อนออกไปเป็นเวลา อย่างน้อย **acregmin** หรือ **acdirmin** วินาที ถ้านี่เป็นการอัพเดตครั้งที่สอง หรือในลำดับต่อมา จะมีการเก็บรายการไว้นานอย่างน้อยเท่ากับช่วงเวลาระหว่าง การอัพเดตสองครั้งล่าสุด แต่ไม่นานเกินกว่า **acregmax** หรือ **acdirmax** วินาที

ความหมายของประสิทธิภาพการติดตั้ง NFS แบบฮาร์ดหรือซอฟต์

ตัวเลือกหนึ่งที่คุณมีเมื่อตั้งค่าคอนฟิกไดเร็กทอรีที่ติดตั้ง NFS คือ คุณต้องการการติดตั้งแบบฮาร์ด (**-o hard**) หรือซอฟต์ (**-o soft**)

เมื่อการเข้าถึงไดเร็กทอรีที่ติดตั้งแบบซอฟต์พบข้อผิดพลาด (โดยปกติ ไทม์เอาต์) หลังจากการติดตั้งเสร็จเรียบร้อยแล้ว ข้อผิดพลาดจะถูกรายงานไปยังโปรแกรมที่ร้อง ขอการเข้าถึงแบบรีโมตในทันที เมื่อการเข้าถึงไดเร็กทอรีที่ติดตั้งแบบฮาร์ด พบข้อผิดพลาด NFS จะลองการดำเนินงานใหม่

ข้อผิดพลาดถาวรในการเข้าถึงไดเร็กทอรีที่ติดตั้งแบบฮาร์ดอาจเลื่อนขั้นเป็นปัญหา ประสิทธิภาพที่สังเกตได้ เนื่องจากจำนวนดีฟอลต์ของการลองใหม่จำนวน 1000 และค่าไทม์เอาต์ดีฟอลต์ 0.7 วินาที รวมเข้ากับขั้นตอนวิธีที่เพิ่มค่าไทม์เอาต์ ของการลองใหม่ที่ต่อเนื่อง หมายความว่า NFS จะพยายามทำการดำเนินงาน ให้สมบูรณ์ต่อไป

ในด้านเทคนิค อาจลดจำนวนการลองใหม่ หรือเพิ่มค่าไทม์เอาต์ หรือทั้งสองอย่างได้โดยใช้อ็อพชันของคำสั่ง **mount** โชคไม่ดีที่การเปลี่ยนค่าเหล่านี้เพียงพอที่จะแก้ไขปัญหาประสิทธิภาพที่สังเกตได้ อาจทำให้เกิดการรายงานข้อผิดพลาดฮาร์ดที่ไม่จำเป็น ให้ใช้อ็อพชัน **intr** เพื่อติดตั้งไดเร็กทอรีที่ติดตั้งแบบฮาร์ดแทน ซึ่งช่วยให้ผู้ใช้ สามารถขัดจังหวะกระบวนการที่อยู่ในลูปลองใหม่โดยใช้คีย์บอร์ดได้

แม้ว่าการติดตั้งไดเร็กทอรีแบบซอฟต์ส่งผลให้สามารถตรวจพบข้อผิดพลาด ได้เร็วขึ้น แต่ก็มีความเสี่ยงที่ข้อมูลจะเสียหายสูง โดยทั่วไป ไดเร็กทอรีการอ่าน/การบันทึก ควรมีการติดตั้งแบบฮาร์ด

การส่งผ่านใหม่ที่ไม่จำเป็น

สิ่งที่เกี่ยวข้องกับคำถามการติดตั้งแบบ hard–และ–soft คือคำถาม ของช่วงเวลาไทม์เอาต์ที่เหมาะสมสำหรับการตั้งค่าคอนฟิกเครือข่ายที่กำหนด

ถ้าเซิร์ฟเวอร์มีโหลดหนัก แยกจากไคลเอ็นต์ด้วยบริดจ์หรือ gateways หนึ่ง รายการขึ้นไป หรือเชื่อมต่อกับไคลเอ็นต์โดย WAN เกณฑ์ดีฟอลต์ ไทม์เอาต์อาจไม่เป็นจริง ถ้าเป็นเช่นนั้น ทั้งเซิร์ฟเวอร์และไคลเอ็นต์ต้องมีภาระ การส่งผ่านใหม่ที่ไม่จำเป็น ตัวอย่างเช่น ถ้าคำสั่งต่อไปนี้:

```
# nfsstat -c
```

รายงานค่าจำนวนมาก เช่น มากกว่าห้าเปอร์เซ็นต์ของผลรวม ของทั้ง `timeouts` และ `badxids` คุณควรเพิ่มพารามิเตอร์ *timeo* ด้วย คำสั่ง **mount**

ระบุไดเร็กทอรีที่คุณต้องการเปลี่ยน และป้อนค่าใหม่ในหน่วยมิลลิวินาที บนบรรทัด **NFS TIMEOUT**

เวลาดีฟอลต์คือ 0.7 วินาที, **timeo=7**, แต่ค่านี้มีการจัดดำเนินการ ในส่วนขยายเคอร์นัล NFS ขึ้นอยู่กับชนิดของการเรียก ตัวอย่างเช่น สำหรับการเรียกการอ่าน ค่าจะเป็นสองเท่าคือ 1.4 วินาที

เพื่อให้สามารถควบคุมค่า *timeo* สำหรับไคลเอ็นต์ระบบ ปฏิบัติการเวอร์ชัน 4 คุณต้องตั้งค่าอ็อพชัน **nfs_dynamic_retrans** ของ คำสั่ง **nfso** เป็น 0 คุณสามารถเปลี่ยนค่า *timeo* ได้สองวิธี และในกรณีที่กำหนดใดๆ มีวิธีที่ถูกต้องในการเปลี่ยนเพียงวิธีเดียวเท่านั้น วิธีที่ถูกต้อง การทำให้ไทม์เอาต์นานขึ้น หรือสั้นลง ขึ้นอยู่กับว่าเพราะเหตุใดแพ็กเก็ตจึงมาไม่ถึง ในเวลาที่กำหนด

ถ้าแพ็กเก็ตเพียงแต่มาสายและมาถึงในท้ายที่สุด คุณอาจต้องการทำให้ตัวแปร *timeo* ยาวขึ้นเพื่อให้การตอบมีโอกาสส่งคืนก่อนการร้องขอจะถูกส่งผ่านใหม่

อย่างไรก็ตาม ถ้าแพ็กเก็ตถูกทิ้งไปและไม่มีทางมาถึงไคลเอ็นต์ เวลาที่ใช้ในการรอการตอบเป็นเวลาสูญเปล่า และคุณต้องการทำให้ *timeo* สั้นลง

วิธีหนึ่งในการประเมินว่าจะใช้อ็อพชันใดคือ ดูที่เอาต์พุต **nfsstat –cr** ของไคลเอ็นต์ และดูว่าไคลเอ็นต์กำลังรายงานค่า `badxid` จำนวนมากหรือไม่ ค่า `badxid` หมายความว่า ไคลเอ็นต์ RPC ได้รับการตอบการเรียก RPC สำหรับการเรียกอื่น ที่ไม่ใช่การเรียกซึ่งกำลังรออยู่โดยทั่วไป นี่หมายความว่าไคลเอ็นต์ได้รับ การตอบซ้ำสำหรับการเรียกที่ส่งผ่านใหม่ก่อนหน้านี้ ดังนั้นแพ็กเก็ต จึงมาถึงช้าและควรทำให้ *timeo* ยาวขึ้น

นอกจากนี้ ถ้าคุณมีตัววิเคราะห์เครือข่าย คุณสามารถใช้ตัววิเคราะห์เพื่อกำหนดว่า สถานการณ์ใดจากสองสถานการณ์เกิดขึ้น ถ้าไม่มีตัววิเคราะห์นั้น คุณสามารถลองตั้งค่าอ็อพชัน the *timeo* ให้สูงขึ้นและต่ำลง และดูว่าแบบใดให้ประสิทธิภาพ โดยรวมดีกว่ากัน ในบางกรณี มีพฤติกรรมที่ไม่สม่ำเสมอ อ็อพชันที่ดีที่คุณของคุณคือติดตามสาเหตุแท้จริงของความล่าช้า/การทิ้งแพ็กเก็ต และแก้ไขปัญหาจริง นั่นคือเซิร์ฟเวอร์หรือเครือข่าย/อุปกรณ์เครือข่าย

สำหรับการจราจร LAN-to-LAN ผ่านทางบริดจ์ ลองใช้ค่าของ 50 ซึ่งอยู่ในหน่วย มิลลิวินาที สำหรับการเชื่อมต่อ WAN ให้ลองค่า 200 ตรวจสอบสถิติ NFS อีกครั้งหลังจากรออย่างน้อยหนึ่งวัน ถ้าสถิติยังคงบ่งชี้การส่งผ่านใหม่มากเกินไป ให้เพิ่มค่า *timeo* เป็น 50 เปอร์เซ็นต์ แล้วลองอีกครั้ง คุณยังอาจต้องการตรวจสอบเซิร์ฟเวอร์เวิร์กโหลด และโหลดบนบริดจ์และ gateways ที่ขวางอยู่ เพื่อดูว่าองค์ประกอบใดๆ เต็มความสามารถเนื่องจากการจราจรอื่นหรือไม่

การสนับสนุน NFS ACL ที่ไม่ได้ใช้

ถ้าเวิร์กโหลดของคุณไม่ได้ใช้รายการควบคุมการเข้าถึง NFS หรือ ACL บนระบบไฟล์ที่ติดตั้ง คุณสามารถลดเวิร์กโหลดบนทั้งไคลเอ็นต์และเซิร์ฟเวอร์ ได้ในระดับหนึ่งโดยการระบุอ็อพชัน **noacl**

ซึ่งสามารถทำได้ดังนี้:

```
options=noacl
```

ตั้งค่าอ็อพชันนี้เป็นส่วนประกอบของ `/etc/filesystems` stanza ของไคลเอ็นต์สำหรับระบบไฟล์นั้น

การใช้การดำเนินงาน READDIRPLUS

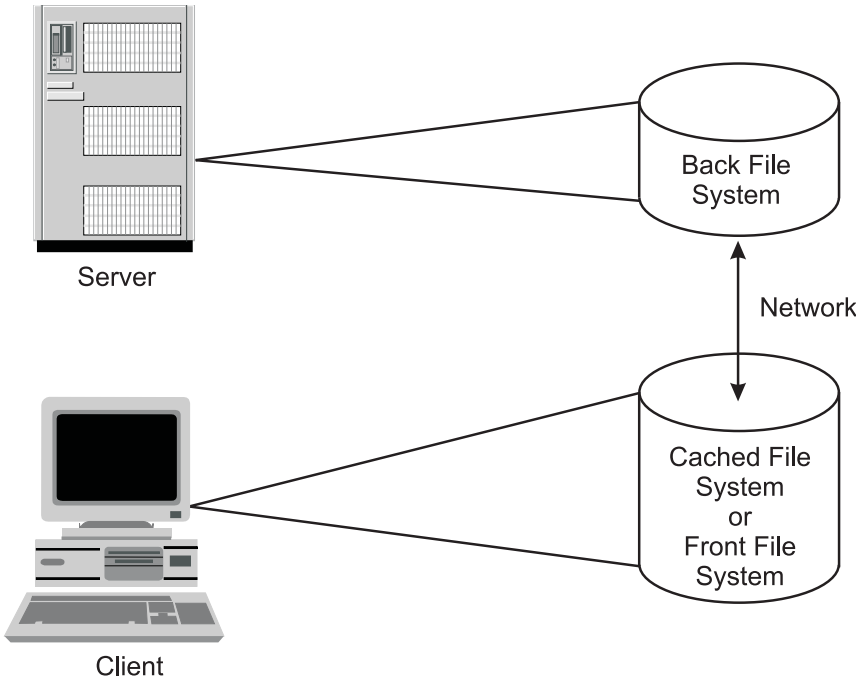
ใน NFS เวอร์ชัน 3 ข้อมูลการจัดการไฟล์และแอ็ตทริบิวต์มีการส่งคืนพร้อม กับรายการไดเร็กทอรีผ่านทางการดำเนินงาน READDIRPLUS ลักษณะนี้ช่วยให้ ไคลเอ็นต์ไม่ต้องเคียวรีเซิร์ฟเวอร์เกี่ยวกับข้อมูลดังกล่าวแยกต่างหากกัน สำหรับแต่ละรายการ ดังที่ทำกับ NFS เวอร์ชัน 2 และเพิ่มประสิทธิภาพขึ้นมาก

อย่างไรก็ตาม ในบางสภาพแวดล้อมที่มีไดเร็กทอรีขนาดใหญ่ซึ่งมีเฉพาะข้อมูล เกี่ยวกับชุดย่อยขนาดเล็กของรายการไดเร็กทอรีที่ใช้โดยไคลเอ็นต์ การดำเนินงาน READDIRPLUS ใน NFS เวอร์ชัน 3 อาจทำให้ประสิทธิภาพช้าลง ในกรณีดังกล่าว สามารถใช้อ็อพชัน **nfs_v3_server_readdirplus** ของคำสั่ง **nsfo** เพื่อปิดใช้งานการใช้ READDIRPLUS แต่โดยทั่วไป ไม่แนะนำให้ทำเช่นนั้นเนื่องจาก ไม่เป็นไปตามมาตรฐาน NFS เวอร์ชัน 3

ระบบไฟล์แคช

คุณสามารถใช้ระบบไฟล์แคช หรือ CacheFS เพื่อปรับปรุงผลการทำงานของระบบรีโมตไฟล์ เช่น NFS หรืออุปกรณ์ที่ช้า เช่น ซีดีรอม

เมื่อระบบรีโมตไฟล์ถูกแคชแล้ว ข้อมูลที่อ่านจากระบบรีโมตไฟล์ หรือซีดีรอมที่ถูกเก็บในแคชบนระบบโลคัล ด้วยเหตุนี้ การหลีกเลี่ยงการใช้เน็ตเวิร์กและเซิร์ฟเวอร์ NFS เมื่อเข้าถึงข้อมูลที่เหมือนกันในครั้งที่สอง CacheFS ถูกออกแบบมาเป็นระบบไฟล์ที่ทำเลเยอร์ ซึ่งหมายความว่า CacheFS จะจัดเตรียมความสามารถในการแคชระบบไฟล์หนึ่งระบบ (ระบบไฟล์ NFS ยังถูกเรียกว่า ระบบไฟล์ส่วนหลัง) สำหรับระบบอื่นๆ (ระบบโลคัลไฟล์ของคุณ ยังเรียกว่าระบบไฟล์ส่วนหน้า) ดังที่แสดงอยู่ในภาพประกอบต่อไปนี้:



รูปที่ 24. ระบบไฟล์แคช (CacheFS). รูปประกอบนี้ แสดงเครื่องไคลเอ็นต์และเซิร์ฟเวอร์ที่เชื่อมต่อด้วยเน็ตเวิร์ก สื่อบันทึกหน่วยเก็บบนเซิร์ฟเวอร์จะมีระบบไฟล์ส่วนหลัง สื่อบันทึกหน่วยเก็บ บนไคลเอ็นต์จะมีระบบไฟล์ที่แคชแล้ว หรือระบบไฟล์ส่วนหน้า

ฟังก์ชันของ CacheFS มีดังต่อไปนี้:

1. หลังจากที่สร้างระบบไฟล์ CacheFS บนระบบไคลเอ็นต์แล้ว คุณสามารถระบุระบบไฟล์ที่ต้องถูกประกอบเข้าในแคช
2. เมื่อผู้ใช้บนไคลเอ็นต์พยายามเข้าถึงไฟล์ที่เป็นส่วนหนึ่งของระบบไฟล์ส่วนหลัง ไฟล์เหล่านั้นจะถูกวางอยู่ในแคช แคชจะไม่ได้รับการเติม จนกว่าผู้ใช้ร้องขอการเข้าถึงไฟล์ หรือไฟล์ต่างๆ ดังนั้น คำร้องขอเริ่มต้นในการเข้าถึงไฟล์ จะอยู่ที่ความเร็วของ NFS แต่การเข้าถึงไฟล์เดียวกันในลำดับถัดมา จะอยู่ที่ความเร็วของ JFS บนโลคัล
3. เพื่อมั่นใจว่า ไดเร็กทอรีและไฟล์ที่แคชแล้วถูกทำให้ทันสมัยอยู่เสมอ CacheFS จะตรวจสอบความสอดคล้องกันของไฟล์ที่เก็บอยู่ในแคช เป็นระยะๆ ซึ่งจะทำการเปรียบเทียบเวลาในการแก้ไขในปัจจุบัน กับเวลาในการแก้ไขก่อนหน้านี้
4. ถ้าเวลาในการแก้ไขแตกต่างกัน ข้อมูลและแอ็ตทริบิวต์ทั้งหมด สำหรับไดเร็กทอรีหรือไฟล์นั้นจะถูกลบออกจากแคช และข้อมูลและแอ็ตทริบิวต์ใหม่จะถูกเรียกคืนจากระบบไฟล์ส่วนหลัง

ตัวอย่างของตำแหน่งที่เหมาะสมกับ CacheFS ในสภาวะแวดล้อมแบบ CAD ซึ่งสำเนาต้นฉบับของคอมโพเนนต์ที่วาดสามารถจัดการได้บนเซิร์ฟเวอร์ และสำเนาที่แคชแล้วบนไคลเอ็นต์เวิร์กสเตชันเมื่อใช้งาน

CacheFS จะไม่อนุญาตให้อ่านหรือเขียนบนไฟล์ที่มีขนาด 2 GB หรือใหญ่กว่า

ข้อได้เปรียบเกี่ยวกับผลการทำงานของ CacheFS

เนื่องจากข้อมูล NFS ถูกแคชไว้บนโลคัลดิสก์หากข้อมูลนั้นอ่านมาจากเซิร์ฟเวอร์ คำร้องขอการอ่านไปยังระบบไฟล์ NFS สามารถตอบสนองได้อย่างรวดเร็ว หากข้อมูลได้ถูกเรียกคืนผ่านเครือข่ายอีกครั้ง

ขึ้นอยู่กับขนาดของหน่วยความจำและการใช้ไคลเอ็นต์ จำนวนข้อมูลที่มีขนาดเล็ก อาจถูกจัดการและเรียกคืนจากหน่วยความจำ ดังนั้นข้อได้เปรียบของข้อมูลที่แคชแล้ว บนดิสก์จะใช้กับจำนวนข้อมูลที่มีขนาดใหญ่ซึ่งไม่สามารถเก็บไว้ใน หน่วยความจำได้ ข้อได้เปรียบเพิ่มเติมคือ ข้อมูลบนดิสก์ที่แคชจะถูกพักไว้ที่การปิดระบบ ซึ่งข้อมูลที่แคชในหน่วยความจำจะต้องเรียกคืนจากเซิร์ฟเวอร์อีกครั้ง หลังจากที่รีบูต

ปัญหาคอขวดของ NFS ที่อาจเกิดขึ้นได้จะทำให้เน็ตเวิร์กช้าหรือไม่ว่าง และใช้ดำเนินการของเซิร์ฟเวอร์กับไคลเอ็นต์ NFS มากเกินไป ดังนั้น การเข้าถึงจากระบบไคลเอ็นต์ไปยังเซิร์ฟเวอร์ดูเหมือนจะช้าลงด้วยเช่นกัน CacheFS จะไม่ป้องกันคุณ จากการอ่านในครั้งแรกผ่านเน็ตเวิร์ก และเพื่อเข้าถึงเซิร์ฟเวอร์ แต่คุณสามารถหลีกเลี่ยงการอ่านผ่านเน็ตเวิร์กได้สำหรับคำร้องขอในอนาคตสำหรับข้อมูลที่เหมือนกัน

ถ้าคำร้องขอการอ่านที่มากขึ้นสามารถตอบสนองได้จากโลคัลดิสก์ของไคลเอ็นต์ จำนวนของการเข้าถึง NFS ไปยังเซิร์ฟเวอร์จะลดลง นั่นหมายความว่า ไคลเอ็นต์ที่มากขึ้น สามารถให้บริการได้โดยเซิร์ฟเวอร์ ดังนั้น อัตราส่วนไคลเอ็นต์ต่อเซิร์ฟเวอร์จะเพิ่มขึ้น

คำร้องขอการอ่านที่เพิ่มขึ้นผ่านเน็ตเวิร์กจะลดโหลดของเน็ตเวิร์กลง ดังนั้นจึงอนุญาตให้คุณขอรับการเลี่ยงเน็ตเวิร์กที่ไม่ว่าง หรือพื้นที่สำหรับการถ่ายโอนข้อมูลอื่นๆ

ไม่ใช่ทุกแอ็พพลิเคชันได้รับผลประโยชน์จาก CacheFS เนื่องจาก CacheFS จะเพิ่มความเร็วของผลการทำงานของการอ่าน แอ็พพลิเคชันที่มีคำร้องขอการอ่านขนาดใหญ่ สำหรับข้อมูลเดียวกันอีกครั้งและอีกครั้งที่ได้รับผลประโยชน์จาก CacheFS แอ็พพลิเคชัน CAD ขนาดใหญ่จะได้รับประโยชน์จาก CacheFS เนื่องจากความถี่ของแบบจำลองขนาดใหญ่มาก ที่ถูกโหลดไว้เพื่อการคำนวณ

การทดสอบผลการทำงานจะแสดงว่า ลำดับการอ่านจากระบบไฟล์ CacheFS คือ 2.4 ถึง 3.4 ครั้งซึ่งเร็วกว่าการอ่านจากหน่วยความจำหรือดิสก์ของ NFS

ผลกระทบต่อผลการทำงานของ CacheFS

CacheFS จะไม่เพิ่มผลการทำงานของการเขียนลงในระบบไฟล์ NFS อย่างไรก็ตาม คุณมีอ็อพชันของการเขียนเพื่อเลือกเป็นพารามิเตอร์ให้กับอ็อพชัน **-o** ของคำสั่ง **mount** เมื่อประกอบเข้ากับ CacheFS ซึ่งจะมีอิทธิพลต่อ ลำดับการอ่านผลการทำงานของข้อมูล

อ็อพชันการเขียนมีดังต่อไปนี้:

การเขียนโดยรอบ

โหมดการเขียนโดยรอบคือโหมดดีฟอลต์ และจะจัดการกับการเขียนด้วยวิธีเดียวกันกับที่ NFS ทำ การเขียนจะถูกทำในระบบไฟล์ส่วนหลัง และ ไฟล์ที่ได้รับผลกระทบจะถูกลบออกจากแคช นั่นหมายความว่า การเขียนโดยรอบจะยกเลิกแคช และข้อมูลใหม่จะถูกขอรับกลับจากเซิร์ฟเวอร์หลังจากที่เขียน

ไม่แบ่งใช้

คุณสามารถใช้โหมดไม่แบ่งใช้เมื่อคุณมั่นใจว่า ไม่มีผู้ใดต้องการเขียนลงใน ระบบไฟล์ที่แคช ในโหมดนี้ การเขียนทั้งหมดจะถูกทำทั้งในระบบไฟล์ส่วนหน้าและระบบไฟล์ส่วนหลัง และไฟล์จะเหลืออยู่ในแคช นั่นหมายความว่า สิทธิการอ่านในอนาคตสามารถทำได้กับแคช แทนการไปยังเซิร์ฟเวอร์

การอ่านขนาดเล็กอาจเก็บอยู่ในหน่วยความจำใดๆ (ขึ้นอยู่กับการใช้หน่วยความจำของคุณ) ดังนั้น จึงไม่มีประโยชน์ในการทำแคชข้อมูลบนดิสก์ การทำแคชของการสุ่มการอ่าน กับบล็อกข้อมูลที่ต่างกันจะไม่ช่วยคุณ ยกเว้นเสียแต่คุณจะเข้าถึงข้อมูลอีกครั้งและอีกครั้ง

คำร้องขอการอ่านเริ่มต้นต้องไปยังเซิร์ฟเวอร์ เนื่องจากเวลาที่ผู้ใช้พยายามเข้าถึงไฟล์ที่เป็นส่วนหนึ่งขอระบบไฟล์ส่วนหลัง จะวางไฟล์เหล่านั้นไว้ในแคช สำหรับคำร้องขอการอ่านเริ่มต้น คุณจะมองเห็นความเร็วของ NFS ปกติ เฉพาะการเข้าถึงข้อมูลเดียวกันตามลำดับเท่านั้น ที่คุณจะมองเห็นผลการทำงานของการเข้าถึง JFS บนโลคัล

ความสอดคล้องกันของข้อมูลแคชจะถูกตรวจสอบที่ช่วงเวลา ดังนั้น จึงเป็นสิ่งที่อันตรายในการแคชข้อมูลที่มีการเปลี่ยนแปลงบ่อย CacheFS ควรใช้สำหรับการอ่านอย่างเดียวหรือการอ่านข้อมูลเป็นส่วนใหญ่

ผลการทำงานของการเขียนผ่านระบบไฟล์ NFS ที่แคชแล้วจะแตกต่างจาก NFS ในเวอร์ชัน 2 ถึง NFS เวอร์ชัน 3 การทดสอบผลการทำานจะแสดงดังต่อไปนี้:

- การเขียนตามลำดับลงในไฟล์ใหม่ผ่าน NFS เวอร์ชัน 2 ถึงจุดประกอบเข้ากับ CacheFS สามารถความเร็ว 25 เปอร์เซ็นต์ซึ่งช้ากว่าการเขียนไปยังจุดประกอบเข้ากับ NFS เวอร์ชัน 2 โดยตรง
- การเขียนตามลำดับลงในไฟล์ใหม่ผ่าน NFS เวอร์ชัน 3 ไปยังจุดประกอบเข้ากับ CacheFS สามารถมีความเร็ว 6 เท่าซึ่งช้ากว่าการเขียนไปยังจุดประกอบเข้ากับ NFS เวอร์ชัน 3 โดยตรง

การปรับแต่ง CacheFS

CacheFS ไม่ได้ถูกนำไปใช้ตามค่าดีฟอลต์ หรือแสดงพร้อมต์ในเวลาที่สร้าง ระบบไฟล์ NFS คุณต้องระบุระบบไฟล์ที่ชัดเจนซึ่งต้องการประกอบเข้ากับแคช

หากต้องการระบุระบบไฟล์ที่ประกอบเข้าในแคช ให้ทำดังต่อไปนี้:

1. สร้างระบบไฟล์แคชบนโลคัลโดยใช้คำสั่ง **cfsadmin** :

   ```
   # cfsadmin -c -o parameters cache-directory
   ```

   โดยที่ *parameters* จะระบุพารามิเตอร์รีซอร์ส และ *cache-directory* คือชื่อของไดเร็กทอรีที่แคชควรถูกสร้าง
2. ประกอบระบบไฟล์ส่วนหลังเข้ากับแคช:

   ```
   # mount -V cachefs -o backfstype=nfs,cachedir=/cache-directory remhost:/rem-directory local-mount-point
   ```

   โดยที่ *rem-directory* คือชื่อของโฮสต์แบบรีโมตและระบบไฟล์ที่มีข้อมูลตั้งอยู่ และ *local-mount-point* คือจุดประกอบบนไคลเอ็นต์ที่ระบบรีโมตไฟล์ควรถูกประกอบ
3. หรือ คุณสามารถดูแล CacheFS โดยใช้คำสั่ง SMIT (ใช้วิธีลัด **smitty cachefs**)

พารามิเตอร์ทั้งหลายสามารถตั้งค่าในเวลาที่สร้างได้ดังต่อไปนี้:

maxblocks
ตั้งค่าจำนวนสูงสุดของบล็อกที่ CacheFS อนุญาตให้เรียกคืนได้ ภายในระบบไฟล์ส่วนหน้า ค่าดีฟอลต์ = 90 เปอร์เซ็นต์

minblocks
ตั้งค่าจำนวนต่ำสุดของบล็อกที่ CacheFS อนุญาตให้เรียกคืนได้ ภายในระบบไฟล์ส่วนหน้า ค่าดีฟอลต์ = 0 เปอร์เซ็นต์

threshblocks
ตั้งค่าจำนวนของบล็อกที่ต้องพร้อมใช้งานในระบบไฟล์ JFS บนฝั่งไคลเอ็นต์ก่อนที่ CacheFS สามารถเรียกคืนบล็อกได้มากกว่าที่ระบุโดย *minblocks* ค่าดีฟอลต์ = 85 เปอร์เซ็นต์

maxfiles

จำนวนสูงสุดของไฟล์ที่ CacheFS สามารถใช้ได้ โดยแสดงเป็นเปอร์เซ็นต์ ของจำนวนทั้งหมดของ i-nodes ในระบบไฟล์ส่วนหน้า ค่าดีฟอลต์ = 90 เปอร์เซ็นต์

minfiles

จำนวนต่ำสุดของไฟล์ที่ CacheFS อนุญาตให้ใช้ โดยแสดงเป็นเปอร์เซ็นต์ของจำนวนทั้งหมดของ i-nodes ในระบบไฟล์ส่วนหน้า ค่าดีฟอลต์ = 0 เปอร์เซ็นต์

maxfilesize

ขนาดไฟล์ใหญ่สุด แสดงในหน่วยเมกะไบต์ ที่ CacheFS อนุญาตให้แคช ดีฟอลต์คือ = 3

ข้อมูลอ้างอิง NFS

มีไฟล์ คำสั่ง, daemons, และรูทีนย่อยจำนวนมากที่เชื่อมโยง กับ NFS

โปรดดู *Networks and communication management* และ *Commands Reference* สำหรับรายละเอียด

รายชื่อของไฟล์ NFS

มีไฟล์จำนวนมากที่เชื่อมโยงกับ NFS

ต่อไปนี้คือรายชื่อของไฟล์ NFS ที่มีข้อมูลคอนฟิกูเรชัน:

bootparams

แสดงรายการไคลเอ็นต์ที่ไคลเอ็นต์แบบ diskless สามารถใช้สำหรับบูตได้

exports แสดงรายการไดเร็กทอรีที่สามารถเอ็กซ์พอร์ตไปยังไคลเอ็นต์ NFS ได้

networks

มีข้อมูลเกี่ยวกับเน็ตเวิร์กบนอินเตอร์เน็ตเน็ตเวิร์ก

pcnfsd.conf

จัดเตรียมอ็อพชันของคอนฟิกูเรชันสำหรับ **rpc.pcnfsd** daemon

rpc มีรายละเอียดฐานข้อมูลสำหรับโปรแกรม Remote Procedure Call (RPC)

xtab แสดงรายการไดเร็กทอรีที่เอ็กซ์พอร์ตอยู่ในปัจจุบัน

/etc/filesystems

แสดงระบบไฟล์ทั้งหมดที่พยายามประกอบเข้า ขณะที่รีสตาร์ทระบบ

รายการของคำสั่ง NFS

มีคำสั่งจำนวนมากที่เชื่อมโยงกับ NFS

ต่อไปนี้คือรายการของคำสั่ง NFS

chnfs สตาร์ทจำนวนของ **biod** และ **nfsd** daemon ที่ระบุไว้

mknfs ปรับแต่งระบบให้รัน NFS และสตาร์ท NFS daemon

nfso ปรับแต่งเน็ตเวิร์กอ็อพชัน NFS

automount
ประกอบเข้ากับระบบไฟล์ NFS โดยอัตโนมัติ

chnfsexp
เปลี่ยนแอ็ตทริบิวต์ของไดเร็กทอรี NFS ที่เอ็กซ์พอร์ต

chnfsmnt
เปลี่ยนแอ็ตทริบิวต์ของไดเร็กทอรี NFS ที่ประกอบเข้า

exportfs
เอ็กซ์พอร์ตและยกเลิกการเอ็กซ์พอร์ตไดเร็กทอรีกับไคลเอ็นต์ NFS

lsnfsexp
แสดงคุณสมบัติของไดเร็กทอรีที่เอ็กซ์พอร์ตด้วย NFS

lsnfsmnt
แสดงคุณสมบัติของระบบ NFS ที่ประกอบเข้า

mknfsexp
เอ็กซ์พอร์ตไดเร็กทอรีโดยใช้ NFS

mknfsmnt
ประกอบเข้ากับไดเร็กทอรีโดยใช้ NFS

rmnfs หยุดทำงาน NFS daemon

rmnfsexp
ลบไดเร็กทอรี NFS ที่เอ็กซ์พอร์ตแล้วออกจากรายการของเอ็กซ์พอร์ตของเซิร์ฟเวอร์

rmnfsmnt
ลบระบบไฟล์ที่ประกอบเข้ากับ NFS ออกจากรายการของการประกอบเข้าของไคลเอ็นต์

รายชื่อของ NFS daemon

มี daemon จำนวนมากที่เชื่อมโยงกับ NFS

ต่อไปนี้คือรายการของ daemon ที่ล็อก NFS:

lockd ประมวลผลคำร้องขอล็อกผ่านแพ็กเกจ RPC

statd จัดเตรียมฟังก์ชัน crash-and-recovery สำหรับเซอร์วิสการล็อกบน NFS

ต่อไปนี้คือรายการของ daemon เซอร์วิสเน็ตเวิร์กและยูทิลิตี:

biod ส่งคำร้องขออ่านและเขียนของไคลเอ็นต์ไปยังเซิร์ฟเวอร์

mountd
ตอบคำร้องขอจากไคลเอ็นต์สำหรับ mount ระบบไฟล์

nfsd สตาร์ท daemons ที่จัดการกับคำร้องขอของไคลเอ็นต์สำหรับการดำเนินการกับระบบไฟล์

pcnfsd จัดการกับคำร้องขอเซอร์วิสจากไคลเอ็นต์ PC-NFS

nfsstat แสดงข้อมูลเกี่ยวกับความสามารถของเครื่องเพื่อรับการเรียก

on เรียกใช้งานคำสั่งบนเครื่องรีโมต

portmap
แม็พหมายเลขโปรแกรม RPC กับหมายเลขอินเตอร์เน็ตพอร์ต

rexd ยอมรับคำร้องขอเพื่อรันโปรแกรมจากเครื่องรีโมต

rpcgen สร้างโค้ด C เพื่อนำโปรโตคอล RPC ไปใช้งาน

rpcinfo รายงานสถานะของเซิร์ฟเวอร์ RPC

rstatd ส่งคืนข้อมูลสถิติผลการทำงานที่ได้รับจากเคอร์เนล

rup แสดงสถานะของโฮสต์แบบรีโมตบนโลคัลเน็ตเวิร์ก

rusers รายงานรายการของผู้ใช้ที่ล็อกออนไปยังเครื่องรีโมต

rusersd ตอบกลับเคียวรี่จากคำสั่ง **rusers**

rwall ส่งข้อความไปยังผู้ใช้ทั้งหมดบนเน็ตเวิร์ก

rwalld จัดการคำร้องขอจากคำสั่ง **rwall**

showmount
แสดงรายการของไคลเอ็นต์ที่ mount กับระบบรีโมตไฟล์

spray ส่งจำนวนแพ็กเก็ตที่ระบุไปยังโฮสต์

sprayd รับแพ็กเก็ตที่ส่งโดยคำสั่ง **spray**

ต่อไปนี้คือรายการของ daemon สำหรับการสร้างเน็ตเวิร์กที่มีความปลอดภัยและยูทิลิตี:

chkey เปลี่ยนคีย์การเข้ารหัสลับของผู้ใช้

keyenvoy
จัดเตรียมตัวกลางระหว่างการประมวลผลผู้ใช้และคีย์เซิร์ฟเวอร์

keylogin
ถอดรหัสและเก็บคีย์ลับของผู้ใช้

keyserv
เก็บพับลิกคีย์และไพรเวตคีย์

mkkeyserv
สตาร์ท **keyserv** daemon และยกเลิกการใส่ข้อคิดเห็นในรายการที่เหมาะสมในไฟล์ `/etc/rc.nfs`

newkey สร้างคีย์ใหม่ในไฟล์พับลิกคีย์

rmkeyserv
หยุด **keyserv** daemon และใส่ข้อคิดเห็นในรายการสำหรับ **keyserv** daemon ในไฟล์ `/etc/rc.nfs`

ypupdated
อัพเดตข้อมูลในแม็พ Network Information Service (NIS)

ต่อไปนี้คือไคลเอ็นต์แบบ diskless ที่สนับสนุนไฟล์คอนฟิกูเรชัน:

bootparamd
จัดเตรียมข้อมูลที่จำเป็นสำหรับการบูตไปยังไคลเอ็นต์ diskless

ต่อไปนี้คือรายการรูทีนย่อย NFS:

cbc_crypt( ), des_setparity( ), or ecb_crypt( )
นำรูทีน Data Encryption Standard (DES) ไปใช้งาน

ผลการทำงาน LPAR

หัวข้อนี้แสดงถึงข้อมูลส่วนลึกและคำแนะนำสำหรับการพิจารณา การมอนิเตอร์ และการปรับผลการทำงานของ AIX ในพาร์ติชันที่รันอยู่บนระบบแบบอิง POWER4

สำหรับข้อมูลเพิ่มเติมเกี่ยวกับพาร์ติชันและการนำไปปฏิบัติ โปรดดู *AIX 5L™ Version 5.3 AIX Installation in a Partitioned Environment* หรือ *Hardware Management Console Installation and Operations Guide*

ข้อควรพิจารณาเกี่ยวกับประสิทธิภาพของการแบ่งพาร์ติชันแบบโลจิคัล

คุณสามารถตั้งค่าคอนฟิกระบบที่ใช้ POWER4 ได้หลายวิธี เช่น ระบบขนาดใหญ่ที่มี POWER4 CPUs packaged เป็น Multi Chip Modules (MCM) หรือระบบขนาดเล็กที่มี POWER4 CPUs packaged เป็น Single Chip Modules (SCM)

แอ็พพลิเคชันเวิร์กโหลดอาจมีลักษณะประสิทธิภาพแตกต่างกันไป บนระบบเหล่านี้

LPAR นำเสนอการใช้ฮาร์ดแวร์ที่ ยืดหยุ่นเมื่อแอ็พพลิเคชันซอฟต์แวร์ไม่ได้ปรับสเกลอย่างเหมาะสมบนตัวประมวลผลจำนวนมาก หรือเมื่อต้องการความยืดหยุ่นของพาร์ติชัน ในกรณีเหล่านี้ การรันหลายอินสแตนซ์ของแอ็พพลิเคชันบนพาร์ติชันขนาดเล็กลงที่แยกต่างหาก อาจให้ผลผลิตที่ดีกว่าการรันอินสแตนซ์ขนาดใหญ่ของแอ็พพลิเคชันเพียง อินสแตนซ์เดียว ตัวอย่างเช่น ถ้าแอ็พพลิเคชันได้รับการออกแบบมาเป็นกระบวนการเดียว ที่มีการเธรดน้อยมากหรือไม่มีเลย แอ็พพลิเคชันจะรันได้ดีบนระบบ 2-way หรือ 4-way แต่อาจรันโดยไม่ข้อจำกัดบนระบบ SMP ที่ใหญ่ขึ้น แทนที่จะ ออกแบบแอ็พพลิเคชันใหม่ให้มี CPUs จำนวนมากขึ้น แอ็พพลิเคชันสามารถรันในชุดขนานของ CPU พาร์ติชันที่เล็กลงได้

ควรพิจารณาประสิทธิภาพของการแบ่งพาร์ติชันแบบโลจิคัลเมื่อทำการวิเคราะห์ การแปรผันเล็กน้อยโดยละเอียด Hypervisor และเฟิร์มแวร์ จัดการกับการแม็พของหน่วยความจำ, CPUs และอะแด็ปเตอร์สำหรับพาร์ติชัน โดยทั่วไป แอ็พพลิเคชันไม่ทราบว่าหน่วยความจำของพาร์ติชันตั้งอยู่ที่ใด มีการกำหนด CPUs ใด หรืออะแด็ปเตอร์ใดถูกใช้งานอยู่ มีข้อควรพิจารณาเกี่ยวกับการมอนิเตอร์และ การปรับประสิทธิภาพเป็นจำนวนมากสำหรับแอ็พพลิเคชัน ซึ่งเกี่ยวข้องกับ ที่ตั้งของหน่วยความจำและ CPUs, การแบ่งใช้แคช L2 และ L3 และโอเวอร์เฮด ของ hypervisor ในการจัดการสภาพแวดล้อมที่แบ่งพาร์ติชันบนระบบ

ข้อควรพิจารณาเกี่ยวกับระบบปฏิบัติการ LPAR

มีหลายๆ ปัญหาเพื่อพิจารณาเกี่ยวกับระบบปฏิบัติการ LPAR

พาร์ติชันบนระบบแบบอิง POWER4 สามารถรันอยู่บนระบบปฏิบัติต่อไปนี้:

- ระบบปฏิบัติการ AIX ที่มีเคอร์เนล 32 บิต
- AIX พร้อมกับเคอร์เนลแบบ 64 บิต เคอร์เนล AIX ขนาดแบบ 64 บิต จะถูกออปติไมซ์สำหรับการรันแอ็พพลิเคชันแบบ 64 บิต และปรับปรุงความสามารถในการวัด โดยอนุญาตให้แอ็พพลิเคชันใช้ขนาดของหน่วยความจำที่มีขนาดใหญ่กว่าของหน่วยความจำฟิสิคัล ซึ่งกำหนดให้กับพาร์ติชันนั้น

- Linux พร้อมกับเคอร์เนลแบบ 64 บิต

แต่ละพาร์ติชันบนระบบสามารถรันระดับที่ต่างกันของระบบปฏิบัติการ พาร์ติชันจะถูกออกแบบมาเพื่อแยกซอฟต์แวร์ที่รันอยู่ในหนึ่งพาร์ติชัน จากซอฟต์แวร์ที่รันในพาร์ติชันอื่นๆ เหตุการณ์นี้จะสอดแทรกการปกป้องพร้อมกับซอฟต์แวร์ที่หยุดการทำงาน และตรวจสอบซอฟต์แวร์ที่พยายามหยุดทำงาน LPAR การเข้าถึงข้อมูลระหว่างพาร์ติชัน จะถูกป้องกันไว้ นอกเหนือจากการเข้าถึงภาวะเชื่อมต่อเน็ตเวิร์กปกติ พาร์ติชันซอฟต์แวร ที่ขัดข้องในหนึ่งพาร์ติชันจะไม่เป็นสาเหตุทำให้เกิดการรบกวนพาร์ติชันอื่น ซึ่งประกอบด้วยความล้มเหลวทั้งในส่วนของแอ็พพลิเคชันซอฟต์แวร์ และซอฟต์แวร์ระบบปฏิบัติการ พาร์ติชันไม่สามารถใช้ฮาร์ดแวร์ที่แบ่งใช้รีซอร์สอย่างกว้างขวาง ซึ่งพาร์ติชันอื่นที่ใช้รีซอร์สนั้นต้องการ ตัวอย่างเช่น พาร์ติชันที่แบ่งใช้ชิปของบริดจ์ PCI ที่เหมือนกัน ไม่สามารถล็อกบัสได้อย่างแน่นอน

คอมโพเนนต์ระบบ

คอมโพเนนต์ระบบหลายอย่างต้องทำงานร่วมกันเพื่อดำเนินการและสนับสนุน สภาพแวดล้อม LPAR

ความสัมพันธ์ระหว่างตัวประมวลผล เฟิร์มแวร์ และระบบปฏิบัติการทำให้ ฟังก์ชันเฉพาะจำเป็นต้องได้รับการสนับสนุนจากแต่ละคอมโพเนนต์เหล่านี้ ด้วยเหตุนี้ การนำ LPAR ไปใช้จึงไม่ได้ขึ้นอยู่ กับซอฟต์แวร์ ฮาร์ดแวร์ หรือเฟิร์มแวร์เพียงอย่างเดียวเท่านั้น แต่ยังขึ้นอยู่กับ ความสัมพันธ์ระหว่างคอมโพเนนต์ทั้งสามด้วย POWER4 microprocessor สนับสนุนรูปแบบที่พัฒนาขึ้นของการเรียกระบบ ที่เรียกกันว่าโหมด Hypervisor ซึ่งอนุญาตให้โปรแกรมที่ได้รับสิทธิ เข้าถึงฟังก์ชันฮาร์ดแวร์บางอย่างได้ การสนับสนุนยังรวมถึงการป้องกัน ฟังก์ชันดังกล่าวในตัวประมวลผลด้วย โหมดนี้ช่วยให้ตัวประมวลผลสามารถเข้าถึงข้อมูลเกี่ยวกับ ระบบต่างๆ ที่ตั้งอยู่ภายนอกขอบเขตของพาร์ติชันซึ่งตัวประมวลผล ตั้งอยู่ Hypervisor ใช้ CPU ระบบและ รีซอร์สหน่วยความจำเพียงเล็กน้อย ดังนั้นเมื่อเปรียบเทียบเวิร์กโหลดที่กำลังรัน โดยมี Hypervisor กับ เวิร์กโหลดที่รันโดยไม่มี Hypervisor โดยปกติ จะแสดงผลกระทบเล็กน้อยเท่านั้น

ระบบบน POWER4 สามารถบูตได้ในการตั้งค่าคอนฟิกพาร์ติชันหลายรูปแบบ รวมถึงดังต่อไปนี้:

- ระบบฮาร์ดแวร์เฉพาะที่ไม่มีการสนับสนุน LPAR รันอยู่ ดังนั้น Hypervisor ไม่ได้ รันอยู่ นี่เรียกว่าพาร์ติชันเต็มระบบ
- พาร์ติชันที่กำลังรันอยู่บนระบบที่มี Hypervisor รันอยู่

ความเกี่ยวข้องในการแบ่งโลจิคัลพาร์ติชัน

ระบบ POWER processor-based platform บางระบบจะมีความสามารถในการสร้างความเกี่ยวข้องในการแบ่งโลจิคัลพาร์ติชัน คุณลักษณะนี้จะพิจารณาว่า ระบบ CPU และรีซอร์สหน่วยความจำจะถูกใช้สำหรับแต่ละพาร์ติชัน ซึ่งอ้างอิงตามตำแหน่งฟิสิคัลที่เกี่ยวข้องกับพาร์ติชันอื่น

Hardware Management Console, HMC จะแบ่งระบบออกเป็น LPAR อย่างเป็นสัดส่วนด้วยตัวประมวลผลแบบ 4 โพรเซสเซอร์ หรือพาร์ติชันแบบ 8 โพรเซสเซอร์สำหรับการเลือกของผู้ดูแลระบบ ในการประมวลผลการติดตั้ง ตัวประมวลผลและหน่วยความจำจะจัดตำแหน่งอยู่บนขอบเขต MCM ซึ่งเป็นการออกแบบมาเพื่ออนุญาตให้ระบบใช้เป็นชุดของโหนดคลัสเตอร์เฉพาะ และจัดเตรียมผลการทำงานที่มีประโยชน์เวิร์กโหลดที่เป็นระบบ และเวิร์กโหลดที่เป็นเทคนิค ถ้าระบบบูตในโหมดนี้ ความสามารถในการปรับรีซอร์สโดยการเพิ่มและลบ CPU และหน่วยความจำจะไม่พร้อมใช้งาน นี่คือผลการทำงานที่ได้รับในเวิร์กโหลดที่รันอยู่ในพาร์ติชันที่มีความเกี่ยวข้อง เหนือโลจิคัลพาร์ติชันแบบปกติ

หมายเหตุ: หน่วยความจำ AIX ที่เกี่ยวข้องกัน จะไม่มีอยู่ในโหมด LPAR

การจัดการเวิร์กโหลดในพาร์ติชัน

ฟังก์ชันการจัดการเวิร์กโหลดเดียวกันใน AIX มีอยู่ ภายในแต่ละ AIX พาร์ติชัน

ไม่พบความแตกต่างใน AIX Workload Manager, หรือ WLM ที่กำลังรันภายในพาร์ติชัน WLM ไม่ได้จัดการเวิร์กโหลดข้ามพาร์ติชัน เจ้าของแอ็พพลิเคชันสามารถ ระบุ CPUs หรือหน่วยความจำสำหรับเวิร์กโหลด และสามารถทำเช่นเดียวกัน นี้สำหรับพาร์ติชัน อย่างไรก็ตาม ในพาร์ติชัน CPUs จะถูกกำหนดให้กับแต่ละ พาร์ติชันภายนอกขอบเขตของผู้จัดการเวิร์กโหลด ดังนั้น ความสามารถในการระบุ ชุดของ CPUs จาก MCM เฉพาะให้กับเวิร์กโหลดเฉพาะจึง ไม่มีอยู่ Workload Manager และคำสั่ง **bindprocessor** ยังคงสามารถยึด CPUs ที่กำหนดก่อนหน้านี้กับเวิร์กโหลดเฉพาะ

ตัวเลือกระหว่างการแบ่งพาร์ติชันและการจัดการกับเวิร์กโหลด

เมื่อต้องทำการเลือกระหว่างการแบ่งพาร์ติชันหรือการใช้การจัดการกับเวิร์กโหลด สำหรับชุดของเวิร์กโหลดเฉพาะ แอ็พพลิเคชัน หรือโซลูชัน มีหลายสถานการณ์ที่ควรพิจารณา

โดยทั่วไป การแบ่งพาร์ติชันจะพิจารณาโหมดของการจัดการที่เหมาะสมมากกว่า เมื่อสิ่งต่อไปนี้แสดงอยู่:

- การพึ่งพาแอ็พพลิเคชันที่ต้องการเวอร์ชันที่ต่างกัน หรือระดับของโปรแกรมฟิกซ์ของระบบปฏิบัติการ
- ข้อกำหนดด้านการรักษาความปลอดภัยที่ต้องการเจ้าของ/ผู้ดูแลระบบที่ต่างกัน การแยกจากกันของข้อมูลที่สำคัญหรือแอ็พพลิเคชันที่กระจายพร้อมกับไฟร์วอลล์เน็ตเวิร์ก ระหว่างแอ็พพลิเคชัน
- โพรซีเดอร์การกู้คืนอื่นๆ ตัวอย่างเช่น การทำคลัสเตอร์ **HA** และความล้มเหลวของแอ็พพลิเคชันหรือการนำเสนอโพรซีเดอร์การกู้คืน
- การแยกความล้มเหลวที่จำเป็นต้องมี ดังนั้น แอ็พพลิเคชันหรือระบบปฏิบัติการที่ล้มเหลว จะไม่มีผลกระทบต่อกัน
- การแยกของผลการทำงานจำเป็นต้องมี ดังนั้น คุณสมบัติของผลการทำงานของเวิร์กโหลด ต้องไม่ก้าวก่ายกับรีซอร์สที่แบ่งใช้

การแยกผลการทำงานเป็นสิ่งที่สำคัญ เมื่อคุณกำลังมอนิเตอร์ หรือปรับเวิร์กโหลดของแอ็พพลิเคชันบนระบบที่สนับสนุนการแบ่งพาร์ติชัน ซึ่งสามารถท้าท้ายการสร้างการควบคุมการจัดการเวิร์กโหลด AIX อย่างมีประสิทธิภาพ เมื่อคุณทำงานอยู่ในการเชื่อมกับเวิร์กโหลดอื่นๆ ที่สำคัญ ในเวลาเดียวกัน การมอนิเตอร์และการปรับแอ็พพลิเคชันจำนวนมาก คือบทเรียนเพิ่มเติมในการแยกพาร์ติชันซึ่งมีรีซอร์สเล็กๆ จำนวนมากที่สามารถกำหนด ให้กับพาร์ติชันได้

ผลกระทบที่มีต่อผลการทำงานของ LPAR

ผลกระทบของการรันใน LPAR จะไม่แตกต่างจากการรันบนตัวประมวลผลที่คล้ายกันในโหมด SMP

ฟังก์ชัน hypervisor ที่รันบนระบบในโหมด LPAR จะเพิ่มการใช้งานที่น้อยกว่า 5% ให้กับหน่วยความจำปกติและการดำเนินการ I/O การรันพาร์ติชันจำนวนมากอย่างพร้อมเพียงกันจะกระทบต่อผลการทำงานเพียงเล็กน้อย บนพาร์ติชันอื่นๆ แต่จะมีสถานการณ์ที่สามารถส่งผลต่อ ผลการทำงานได้ ซึ่งจะมีการใช้งานเป็นพิเศษที่เชื่อมโยงกับ hypervisor สำหรับการจัดการหน่วยความจำเสมือน และควรมีสถานะที่ไม่รุนแรงสำหรับเวิร์กโหลดโดยส่วนใหญ่ แต่ผลกระทบจะเพิ่มจำนวนของกิจกรรมการแม็พเพจอย่างกว้างขวาง การแบ่งพาร์ติชันอาจช่วยให้มีผลการทำงานที่ดีขึ้น ในกรณีของแอ็พพลิเคชันที่ไม่มีสเกลที่ไม่ค่อยดีบนระบบ SMP ขนาดใหญ่ ด้วยการบังคับให้มีการแบ่งแยกระหว่างเวิร์กโหลดที่รันอยู่ในพาร์ติชันที่แยกออก

การจำลองของระบบที่เล็กกว่า

วิธีที่ดีที่สุดในการจำลองหน่วยความจำจำนวนน้อยลงคือการลด จำนวนของหน่วยความจำที่มีอยู่สำหรับพาร์ติชัน

เมื่อใช้ระบบ POWER4-based MCM คำสั่ง **rmss** จะจัดสรร หน่วยความจำจากระบบ โดยไม่คำนึงถึงที่ตั้งของหน่วยความจำนั้น ใน MCM ลักษณะประสิทธิภาพเฉพาะโดยละเอียดอาจเปลี่ยนแปลงได้ขึ้นอยู่กับ ชนิดหน่วยความจำที่มีอยู่และชนิดหน่วยความจำที่กำหนดให้กับพาร์ติชัน ตัวอย่าง เช่น ถ้าคุณเคยใช้คำสั่ง **rmss** เพื่อจำลอง พาร์ติชัน 8-way ที่ใช้หน่วยความจำโลคัล

หน่วยความจำที่กำหนดจริง อาจไม่ใช่หน่วยความจำฟิสิคัลที่อยู่ใกล้เคียงกับ MCM มากที่สุด ในข้อเท็จจริง 8 ตัวประมวลผลไม่ใช่ 8 ตัวประมวลผลบน MCM แต่จะถูกกำหนด จากรายการที่มีอยู่แทน

เมื่อยกเลิกการตั้งค่าคอนฟิก CPUs บนระบบที่ทำงานบน MCM มีรายละเอียด เกี่ยวกับเวลาที่ hypervisor ใช้พาธระหว่าง MCMs และหน่วยความจำ ในขณะที่ มีผลกระทบต่อประสิทธิภาพเล็กน้อย อาจมีความแตกต่างเล็กน้อยซึ่ง อาจกระทบต่อการวิเคราะห์ประสิทธิภาพโดยละเอียด

ไมโครโพรเซสเซอร์ในพาร์ติชัน

ไมโครโพรเซสเซอร์สามารถกำหนดให้กับ LPAR

ไมโครโพรเซสเซอร์ที่ได้กำหนดไว้

เพื่อดูรายการของไมโครโพรเซสเซอร์ที่ได้กำหนดให้กับ LPAR ให้เลือกอ็อบเจ็กต์ของระบบที่ถูกจัดการ (CEC) บน HMC และดูคุณสมบัติของอ็อบเจ็กต์

มีแท็บที่แสดงสถานะการจัดสรรตัวประมวลผลทั้งหมดในปัจจุบัน ซึ่งได้กำหนดให้กับพาร์ติชันที่รันอยู่ AIX ใช้หมายเลขของเฟิร์มแวร์ที่จัดเตรียมไว้ ซึ่งอนุญาตให้คุณมองเห็นได้ภายในพาร์ติชันที่ตัวประมวลผลถูกใช้โดยการมองหาหมายเลขไมโครโพรเซสเซอร์และโค้ดที่ตั้ง AIX

การตรวจสอบสถานะของไมโครโพรเซสเซอร์ที่กำหนดให้กับพาร์ติชันแบบสองตัวประมวลผล จะดูคล้ายกับที่แสดงต่อไปนี้:

```
 > lsdev -C | grep proc
proc17   Available 00-17   Processor
proc23   Available 00-23   Processor
```

ผลกระทบของการปิดใช้งานไมโครโพรเซสเซอร์

ขณะปิดใช้งานไมโครโพรเซสเซอร์บนระบบแบบอิง POWER4 ด้วย MCM ระบบจะยังคงเราต์การควบคุมสายงาน และความสามารถในการเข้าถึงหน่วยความจำ ผ่านไมโครโพรเซสเซอร์ที่มีอยู่บนระบบทั้งหมด ซึ่งอาจกระทบกับ ผลการทำงานของเวิร์กโหลดทั้งหมด

การจัดการตัวประมวลผลเสมือนภายในพาร์ติชันหนึ่ง

ตัวจัดตารางเวลาเคอร์นัลมีการพัฒนาขึ้นเพื่อเพิ่มและลดการใช้ ตัวประมวลผลเสมือนพร้อมกับโหลดฉับพลันของพาร์ติชันอย่างมาก ตามที่ประเมินโดยการใช้ประโยชน์ฟิสิคัลของพาร์ติชัน

ในทุกวินาที ตัวจัดตารางเวลาเคอร์นัลจะประเมินจำนวนของตัวประมวลผล เสมือนที่ควรจะเรียกใช้เพื่อสนับสนุนการใช้ประโยชน์ฟิสิคัลของพาร์ติชัน ถ้าจำนวนแสดงการใช้ประโยชน์ตัวประมวลผลเสมือนสูง จำนวนพื้นฐานของ ตัวประมวลผลเสมือนที่ต้องการจะเพิ่มขึ้นเพื่อให้เวิร์กโหลดสามารถขยายได้ คุณสามารถร้องขอตัวประมวลผลเสมือนเพิ่มเติมด้วยคำสั่ง **schedo** ค่านี้ใช้เพื่อกำหนดว่าต้องเปิดใช้งานหรือปิดใช้งานตัวประมวลผลเสมือน เนื่องจากตัวจัดตารางเวลาปรับเฉพาะจำนวนของตัวประมวลผลเสมือน ที่ใช้งานในแต่ละวินาทีทีละหนึ่งเท่านั้น ดังนั้น ถ้าตัวเลขที่คำนวณได้มากกว่าจำนวนของ ตัวประมวลผลเสมือนที่เรียกใช้งานในปัจจุบัน จะมีการเรียกใช้ตัวประมวลผล เสมือน ถ้าตัวเลขที่คำนวณได้น้อยกว่าจำนวนของตัวประมวลผลเสมือนที่เรียกใช้งาน ในปัจจุบัน จะมีการยกเลิกการเรียกใช้ตัวประมวลผลเสมือน

เมื่อตัวประมวลผลเสมือนถูกยกเลิกการเรียกใช้ จะไม่มีการลบตัวประมวลผลนั้นออก จากพาร์ติชันแบบไดนามิกดังเช่น DLPAR ตัวประมวลผลเสมือนจะไม่เป็นตัวเลือกในการรันหรือได้รับงานที่ไม่ได้ยึดอีกต่อไป อย่างไรก็ตาม ตัวประมวลผลนั้นยังคงสามารถรันงานที่ยึดไว้ได้ จำนวนของตัวประมวลออนไลน์โลจิคัล และตัวประมวลเสมือนแบบออนไลน์ที่มองเห็นได้โดย

ผู้ใช้หรือแอ็พพลิเคชัน ไม่เปลี่ยน ไม่มีผลกระทบต่อมิดเดิลแวร์หรือแอ็พพลิเคชันที่กำลังรันบนระบบ เนื่องจากตัวประมวลผลเสมือนที่ใช้งานอยู่และที่ไม่ได้ใช้งานอยู่เป็นตัวประมวลผล ภายในระบบ

ค่าดีฟอลต์ของพารามิเตอร์ *vpm_xvcpus* ที่ปรับได้ คือ 0 ซึ่ง หมายความว่ามีการเปิดใช้งาน folding ซึ่งหมายความว่าตัวประมวลผลเสมือน กำลังมีการจัดการอยู่ คุณสามารถใช้ คำสั่ง schedo เพื่อปรับเปลี่ยนพารามิเตอร์ *vpm_xvcpus* ที่ปรับได้

เมื่อต้องการ ทราบว่าคุณลักษณะการจัดการตัวประมวลผลเสมือนมีการเปิดใช้งานหรือไม่ คุณสามารถใช้คำสั่งต่อไปนี้:

```
# schedo -o vpm_xvcpus
```

เมื่อ ต้องการเพิ่มจำนวนของตัวประมวลผลเสมือนที่ใช้อยู่ขึ้น 1 คุณสามารถใช้คำสั่ง ต่อไปนี้:

```
# schedo -o vpm_xvcpus=1
```

ตัวประมวลผลเสมือนแต่ละตัวสามารถใช้ตัวประมวลผลฟิสิคัลได้สูงสุดหนึ่งตัว จำนวนของตัวประมวลผลเสมือนที่ต้องการมีการกำหนดโดยคำนวณผลรวม ของการใช้ประโยชน์ฟิสิคัล CPU และค่าของ *vpm_xvcpus* ที่ปรับ ได้ ดังแสดงในสมการต่อไปนี้:

```
Number of virtual processors needed =
Physical CPU utilization + Number of additional virtual processors to enable
```

ถ้าจำนวนของตัวประมวลผลเสมือนที่ต้องการน้อยกว่าจำนวนปัจจุบันของตัวประมวลผลเสมือนที่เปิดใช้งาน จะมีการปิดใช้งานตัวประมวลผลเสมือน ถ้าจำนวนของตัวประมวลผลเสมือนที่ต้องการมากกว่าจำนวนปัจจุบันของตัวประมวลผลเสมือนที่เปิดใช้งาน จะมีการเปิดใช้งานตัวประมวลผลเสมือนที่ถูกปิดใช้งาน เธรดที่แนบกับตัวประมวลผล เสมือนที่ปิดใช้งานยังคงสามารถรันได้

หมายเหตุ: คุณควรปัดเศษค่าที่คำนวณจากสมการข้างบนเป็น จำนวนเต็มถัดไปเสมอ

ตัวอย่างต่อไปนี้อธิบายวิธีการคำนวณจำนวนของตัวประมวลผลเสมือน ที่จะใช้:

ในช่วงเวลาล่าสุด พาร์ติชัน A ใช้ตัวประมวลผลสองตัวครึ่ง *vpm_xvcpus* ที่ปรับได้มีการตั้งค่าเป็น 1 โดยใช้สมการ ข้างบน

```
Physical CPU utilization = 2.5
Number of additional virtual processors to enable (vpm_xvcpus) = 1

Number of virtual processors needed = 2.5 + 1 = 3.5
```

การปัดเศษค่าที่คำนวณได้ขึ้นเป็นจำนวนเต็มถัดไป ทำให้ได้ค่า 4 ดังนั้น จำนวนของตัวประมวลผลเสมือนที่ต้องการบนระบบ คือ 4 และถ้าพาร์ติชัน A กำลังรันโดยมี 8 ตัวประมวลผลเสมือน, 4 ตัวประมวลผล เสมือนถูกปิดใช้งานและ 4 ตัวประมวลผลเสมือนยังคงเปิดใช้งานอยู่ ถ้า SMT มีการเปิดใช้งาน แต่ละตัวประมวลผลเสมือนจะให้ตัวประมวลผลโลจิคัล 2 ตัว ดังนั้น 8 ตัวประมวลผล โลจิคัลมีการปิดใช้งานและ 8 ตัวประมวลผลโลจิคัลมีการเปิดใช้งาน

ในตัวอย่างต่อไปนี้ เวิร์กโหลดล่าสุดที่กำลังรันโดยไม่มีการเปิดใช้งานคุณลักษณะ folding จะใช้จำนวนต่ำสุดของแต่ละตัวประมวลผลเสมือน ที่จัดสรรให้กับพาร์ติชัน เอาต์พุตต่อไปนี้จากเครื่องมือ **mpstat –s** บนระบบที่มี 4 CPUs เสมือน บ่งชี้การใช้ประโยชน์ตัวประมวลผลเสมือน และตัวประมวลผลโลจิคัลสองตัวที่เชื่อมโยง:

```
        Proc0                  Proc2                  Proc4                  Proc6
        19.15%                 18.94%                 18.87%                 19.09%
    cpu0       cpu1        cpu2       cpu3        cpu4       cpu5        cpu6       cpu7
   11.09%    8.07%        10.97%    7.98%        10.93%    7.93%        11.08%    8.00%
```

เมื่อเปิดใช้งานคุณลักษณะ folding ระบบจะคำนวณจำนวนของตัวประมวลผล เสมือนที่ต้องการด้วยสมการข้างบน จากนั้นใช้ค่าที่คำนวณได้ เพื่อลดจำนวนของตัวประมวลผลเสมือนให้เท่ากับที่ต้องการสำหรับการรัน เวิร์กโหลดล่าสุดโดยไม่ทำให้ประสิทธิภาพด้อยลง เอาต์พุตต่อไปนี้จากเครื่องมือ **mpstat –s** บนระบบที่มี 4 CPUs เสมือน บ่งชี้การใช้ประโยชน์ตัวประมวลผลเสมือน และตัวประมวลผลโลจิคัลสองตัวที่เชื่อมโยง:

```
        Proc0                  Proc2                  Proc4                  Proc6
        54.63%                 0.01%                  0.00%                  0.08%
    cpu0      cpu1         cpu2      cpu3         cpu4      cpu5         cpu6      cpu7
   38.89%    15.75%       0.00%     0.00%         0.00%     0.00%        0.03%     0.05%
```

ตามที่คุณเห็นได้จากข้อมูลข้างบน เวิร์กโหลดได้รับประโยชน์จาก การลดการใช้ประโยชน์และการเก็บรักษาตัวประมวลผลเสริม และ affinity ที่เพิ่มขึ้นเมื่อมีงานมากเป็นพิเศษบนตัวประมวลผลเสมือนหนึ่ง อย่างไรก็ตาม เมื่อเวิร์กโหลดหนัก คุณลักษณะ folding ไม่กระทบต่อความสามารถในการใช้ CPUs เสมือนทั้งหมด ถ้าต้องการ

ข้อควรพิจารณาเกี่ยวกับแอ็พพลิเคชัน

คุณควรตื่นตัวกับหลายๆ สิ่งที่เกี่ยวข้องกับแอ็พพลิเคชัน ที่เชื่อมโยงกับ LPAR

โดยทั่วไป แอ็พพลิเคชันจะไม่รับรู้ว่ากำลังทำงานอยู่ใน LPAR ซึ่งมีความแตกต่างที่คุณรับรู้ได้ แต่สิ่งเหล่านี้เป็นตัวพราง แอ็พพลิเคชัน นอกเหนือจากข้อควรพิจารณาเหล่านี้ AIX จะรันอยู่ภายในพาร์ติชันด้วยวิธีเดียวกับที่รันอยู่บนเซิร์ฟเวอร์แบบสแตนด์อะโลน ไม่มีความแตกต่างใดๆ ที่สังเกตเห็นได้จากแอ็พพลิเคชัน หรือมุมมองของคุณ LPAR คือ การมองผ่านแอ็พพลิเคชัน AIX และทูลด้านประสิทธิภาพการทำงานของ AIX เป็นส่วนใหญ่ แอ็พพลิเคชันในกลุ่มที่สามจำเป็นต้องได้รับการรับรองสำหรับระดับของ AIX

คำสั่ง uname ที่รันใน LPAR

ใช้คำสั่ง **uname** เพื่อให้ได้รับข้อมูล เกี่ยวกับระบบที่เกี่ยวข้องกับ LPAR

```
> uname -L
-1 NULL
```

"–1" บ่งชี้ว่าระบบไม่ได้กำลังรันด้วยโลจิคัลพาร์ติชันใดๆ แต่กำลังรันในโหมดพาร์ติชันเต็มระบบ

ตัวอย่างต่อไปนี้แสดงให้เห็นว่าคำสั่ง **uname** นำเสนอ หมายเลขพาร์ติชันและชื่อพาร์ติชันได้อย่างไร ภายใต้การจัดการโดย HMC:

```
 > uname -L
3 Web Server
```

การทราบว่าแอ็พพลิเคชันกำลังรันใน LPAR มีประโยชน์เมื่อคุณกำลังประเมิน ความแตกต่างด้านประสิทธิภาพเล็กน้อย

คอนโซลเสมือน

ไม่มีคอนโซลเสมือนบนแต่ละพาร์ติชัน

ในขณะที่สามารถกำหนดพอร์ตอนุกรมฟิสิคัลให้กับพาร์ติชันได้ แต่สามารถ อยู่ได้ในหนึ่งพาร์ติชันเท่านั้นในแต่ละครั้ง เพื่อการวิเคราะห์และเพื่อให้เอาต์พุต สำหรับข้อความคอนโซล เฟิร์มแวร์จึงใช้ tty เสมือนที่มองเห็น โดย AIX เป็นอุปกรณ์ tty มาตรฐาน เอาต์พุต tty เสมือนมีการสตรีมไปยัง HMC ระบบย่อยการวิเคราะห์ AIX ใช้ tty เสมือนเป็นคอนโซลระบบ จากมุมมองด้านประสิทธิภาพ ถ้ามีการบันทึกข้อมูลจำนวนมากลงในคอนโซลระบบ ซึ่งกำลังมอนิเตอร์อยู่บน คอนโซลของ HMC การเชื่อมต่อไปยัง HMC จะถูกจำกัดโดยการเชื่อมต่อ สายเคเบิลอนุกรม

นาฬิกาเวลาของวัน

แต่ละพาร์ติชันมีค่านาฬิกาเวลาของวันของตัวเอง ดังนั้นพาร์ติชันจึง สามารถทำงานกับโซนเวลาที่แตกต่างกันได้

วิธีเดียวที่พาร์ติชันสามารถสื่อสารระหว่างกันคือ ผ่านทาง การเชื่อมต่อเครือข่ายมาตรฐาน เมื่อดูที่ข้อมูลการติดตามหรือเวลาประทับ จากแต่ละพาร์ติชันบนระบบ แต่ละเวลาประทับจะแตกต่างกันไป ตามวิธีการตั้งค่าคอนฟิกพาร์ติชัน

หมายเลขประจำผลิตภัณฑ์ของระบบ

คำสั่ง **uname -m** แสดงข้อมูลระบบ หลายอย่างเมื่อมีการกำหนดพาร์ติชัน

หมายเลขประจำผลิตภัณฑ์คือหมายเลขประจำผลิตภัณฑ์ของระบบ ซึ่งเหมือนกัน บนพาร์ติชันต่างๆ จะเห็นหมายเลขประจำผลิตภัณฑ์ของระบบเดียวกันในแต่ละ พาร์ติชัน

ข้อควรพิจารณาเกี่ยวกับหน่วยความจำ

มีปัญหาหลายอย่างที่ต้องพิจารณา เมื่อทำงานกับหน่วยความจำ

พาร์ติชันที่ถูกกำหนดด้วยจำนวนของหน่วยความจำ "ที่ต้องมี" "ที่ต้องการ" และ "ต่ำสุด" เมื่อคุณกำลังเข้าถึงการเปลี่ยนแปลงเงื่อนไขของผลการทำงานระหว่างที่ระบบรีบูต จึงเป็นสิ่งสำคัญที่ต้องทราบว่า การจัดสรรหน่วยความจำและ CPU อาจเปลี่ยนไปตามสภาพพร้อมใช้งานของรีซอร์ส และ ควรระลึกว่า จำนวนของหน่วยความจำที่จัดสรรให้กับพาร์ติชันจาก HMC คือจำนวนทั้งหมดที่จัดสรรไว้ ภายในตัวของพาร์ติชัน หน่วยความจำฟิสิคัลบางส่วน จะใช้สำหรับส่วนสนับสนุน hypervisor-page-table-translation

หน่วยความจำจะถูกจัดสรรโดยระบบระหว่างระบบ แอ็พพลิเคชันในพาร์ติชัน ไม่สามารถพิจารณาถึงหน่วยความจำที่ได้จัดสรรไว้แบบฟิสิคัล

การแบ่งพาร์ติชันแบบไดนามิกโลจิคัล

DLPAR พร้อมใช้งานกับระบบ System p แบบอิง POWER4 พร้อมกับไมโครโค้ดอัพเดตเมื่อเดือนตุลาคม 2002 หรือหลังจากนี้ เป็นไปได้ที่จะรันพาร์ติชัน หลากหลายด้วยระดับของระบบปฏิบัติการที่ต่างกัน

ก่อนการเปิดใช้งาน DLPAR คุณต้องรีบูตพาร์ติชันเพื่อเพิ่มรีซอร์สเพิ่มเติมกับระบบ DLPAR จะเพิ่มความยืดหยุ่นของระบบที่แบ่งพาร์ติชันโดยอนุญาตให้คุณเพิ่มและลบตัวประมวลผล หน่วยความจำ สล็อต I/O และลิ้นชัก I/O แบบไดนามิกได้จากโลจิคัลพาร์ติชัน ที่แอ็คทีฟ คุณสามารถกำหนดฮาร์ดแวร์รีซอร์สได้ใหม่อีกครั้ง และปรับเปลี่ยน เพื่อเปลี่ยนความต้องการความสามารถของระบบโดยไม่กระทบกับสภาพพร้อมใช้งานของพาร์ติชัน

คุณสามารถดำเนินการกับการดำเนินการพื้นฐานต่อไปนี้ด้วย DLPAR:

- ย้ายรีซอร์สจากพาร์ติชันหนึ่งไปยังอีกพาร์ติชันหนึ่ง
- ลบรีซอร์สออกจากพาร์ติชัน
- เพิ่มรีซอร์สให้กับพาร์ติชัน

ตัวประมวลผล หน่วยความจำ และสล็อต I/O ที่ไม่ได้กำหนดให้กับพาร์ติชันที่มีอยู่ใน "พูลว่าง" ในปัจจุบัน พาร์ติชันที่มีอยู่บนระบบจะไม่สามารถมองเห็นได้ในพาร์ติชันอื่น บนระบบหรือพูลว่าง ด้วย DLPAR เมื่อคุณลบตัวประมวลผลจากพาร์ติชันที่แอ็คทีฟ ระบบที่รีลีสไปยังพูล และตัวประมวลผลที่สามารถเพิ่มให้กับ พาร์ติชันที่แอ็คทีฟได้ เมื่อตัวประมวลผลที่เพิ่มให้กับ

พาร์ติชันที่แอ็คทีฟ ตัวประมวลผลนั่นจะมีสิทธิเข้าถึงแบบเต็มกับหน่วยความจำของพาร์ติชัน พื้นที่แอดเดรส I/O และอินเตอร์รัปต์ I/O ทั้งหมด ตัวประมวลผลสามารถมีส่วนร่วมอย่างสมบูรณ์ในเวิร์กโหลด ของพาร์ติชันนั้น

คุณสามารถเพิ่มหรือลบหน่วยความจำในส่วนหรือชิ้นของหน่วยความจำขนาด 256 MB ได้ ผลกระทบของการลบหน่วยความจำ บนแอ็พพลิเคชันที่รันอยู่ในพาร์ติชัน AIX จะลดลงตามความเป็นจริงที่ว่า เคอร์เนล AIX จะรันอยู่ในโหมดเสมือนทั้งหมด แอ็พพลิเคชัน ส่วนขยายเคอร์เนล และเคอร์เนลส่วนใหญ่จะใช้หน่วยความจำสเมือนเท่านั้น เมื่อหน่วยความจำถูกลบทิ้ง พาร์ติชันอาจเริ่มต้นการเพจ เนื่องจากส่วนของเคอร์เนล AIX สามารถเพจได้ สิ่งนี้สามารถลดระดับของผลการทำงานลงได้ เมื่อคุณลบหน่วยความจำออก คุณต้องมอนิเตอร์ข้อมูลสถิติ เพื่อมั่นใจว่า การเพจไม่ได้รับอิทธิพล

และมีความเป็นไปได้ที่จะเพิ่มหรือลบสล็อต I/O เช่น เน็ตเวิร์กอะแด็ปเตอร์ อุปกรณ์ซีดีรอม หรืออุปกรณ์เทปออกจากพาร์ติชันที่แอ็คทีฟ การดำเนินการนี้จะหลีกเลี่ยงปัญหา ของการซื้อและติดตั้งอุปกรณ์แบบฟิสิคัลที่ซ้ำกันเพื่อให้เหมาะกับพาร์ติชันจำนวนมาก เมื่อตัวของอุปกรณ์อาจไม่ได้ถูกนำมาใช้บ่อย ซึ่งไม่เหมือนกับการเพิ่ม หรือการลบตัวประมวลผลหรือหน่วยความจำ การทำคอนฟิกูเรชันใหม่ของสล็อต I/O ยังต้องการฮ็อตปลั๊กโพรซีเดอร์ก่อนที่จะเพิ่มหรือลบอุปกรณ์ ที่อยู่ในพาร์ติชันที่แอ็คทีฟ ฮ็อตปลั๊กโพรซีเดอร์จะพร้อมใช้งานผ่าน SMIT

Hardware Management Console หรือ HMC จะถูกพ่วงต่อกับระบบ และอนุญาตให้คุณดำเนินการกับการดำเนินการทำคอนฟิกูเรชันแบบไดนามิกใหม่ (DR) HMC ต้องรัน R3V1.0 หรือรุ่นถัดมาเพื่อสนับสนุน DLPAR สำหรับรายการของการดำเนินการ HMC ที่เกี่ยวข้องกับ DLPAR โปรดอ้างถึง *คำแนะนำในการแบ่งพาร์ติชันให้เสร็จสิ้นสำหรับเซิร์ฟเวอร์ IBM eServer pSeries*

hypervisor คือเลเยอร์ขนาดเล็กของซอฟต์แวร์ซึ่งจัดเตรียมความสามารถในการจัดการฮาร์ดแวร์ และการแยกกันกับเครื่องเสมือน (พาร์ติชัน) ที่รันอยู่บนระบบฟิสิคัลเดี่ยว คำสั่งในการควบคุมการเคลื่อนไหวของรีซอร์สระหว่างพาร์ติชัน สามารถส่งผ่านไปยัง LPAR hypervisor ผ่าน HMC graphical user interface หรือผ่านบรรทัดรับคำสั่ง HMC คุณสามารถมีได้เพียงหนึ่งอินสแตนซ์ของ hypervisor ที่รันอยู่ และเฉพาะ hypervisor ที่มีความสามารถในการมองเห็นและกำหนดรีซอร์สของระบบเท่านั้น DLPAR ไม่ได้ยอมรับการรักษาความปลอดภัยของพาร์ติชัน รีซอร์สที่ย้ายระหว่างพาร์ติชัน จะถูกกำหนดค่าเริ่มต้นใหม่ ดังนั้น จึงไม่มีข้อมูลที่เหลืออยู่ด้านหลัง

ผลกระทบของผลการทำงาน DLPAR

มีผลกระทบหลายอย่างของการเพิ่มขึ้นและลดผลการทำงาน DLPAR

คุณสามารถเพิ่มหรือลดหน่วยความจำในบล็อกหน่วยความจำแบบโลจิคัลจำนวนมาก ขณะลบหน่วยความจำจากพาร์ติชัน เวลาที่ใช้เพื่อเสร็จสิ้นการดำเนินการ DLPAR จะเกี่ยวข้องกับจำนวนของส่วนของหน่วยความจำที่จะถูกลบ ตัวอย่างเช่น การดำเนินการ DR ที่ต้องลบหน่วยความจำขนาด 4 GB ออกจาพาร์ติชันที่ไม่ได้ทำงานจะใช้เวลา 1 ถึง 2 นาที อย่างไรก็ตาม การแบ่งพาร์ติชันเพจของหน่วยความจำขนาดใหญ่แบบไดนามิกจะไม่ได้รับการสนับสนุน ส่วนของหน่วยความจำ ที่มีเพจขนาดใหญ่จะไม่สามารถลบทิ้งได้

คอนฟิกูเรชันการแบ่งโลจิคัลพาร์ติชันที่เกี่ยวข้องจะจัดสรร CPU และรีซอร์สหน่วยความจำในรูปแบบที่ไม่เปลี่ยนแปลงบนโมดูลแบบมัลติชิป MCM และขอบเขต HMC จะไม่ได้จัดเตรียมตัวประมวลผล DR หรือหน่วยความจำที่สนับสนุนพาร์ติชันที่เกี่ยวข้อง เฉพาะรีซอร์สอะแด็ปเตอร์ I/O เท่านั้น ที่สามารถปรับแต่งใหม่ได้ เมื่อคุณกำลังรันการแบ่งโลจิคัลพาร์ติชันที่เกี่ยวข้อง

คุณยังสามารถใช้ประโยชน์จากการจัดสรรรีซอร์สแบบไดนามิกและการจัดสรรคืน โดยการเปิดใช้งานแอ็พพลิเคชันและมิดเดิลแวร์ให้รับรู้ DLPAR นั่นหมายความว่า แอ็พพลิเคชันสามารถปรับขนาดแอ็พพลิเคชันเองให้เหมาะสมกับรีซอร์ส

ฮาร์ดแวร์ใหม่ AIX จัดเตรียมสคริปต์ DLPAR และ APIs เพื่อปรับขนาดแอ็พพลิเคชันของผู้ขายแบบไดนามิก คุณสามารถค้นหาคำสั่ง สำหรับการใช้สคริปต์เหล่านี้ หรือ API ในส่วนของ DLPAR ของ *General Programming Concepts*

เครื่องมือในการปรับ DLPAR

มีเครื่องมือจำนวนมากที่สามารถใช้เพื่อมอนิเตอร์และสนับสนุน DLPAR

ด้วย DLPAR จำนวนของตัวประมวลผลแบบออนไลน์สามารถเปลี่ยนแปลงแบบไดนามิกได้ เพื่อติดตามความแตกต่างระหว่าง จำนวนของตัวประมวลผลแบบออนไลน์และจำนวนสูงสุดของตัวประมวลผลที่เป็นไปได้ ในระบบ คุณสามารถใช้พารามิเตอร์ต่อไปนี้ได้:

_system_configuration.ncpus
เคียวรี่จำนวนของตัวประมวลผลแบบออนไลน์

_system_configuration.max_ncpus
จัดเตรียมจำนวนสูงสุดของตัวประมวลผลที่อาจเป็นไปได้ในระบบ

AIX สนับสนุน hook การติดตาม **38F** เพื่อเปิดใช้งานตัวช่วยการติดตามในการดักจับการเพิ่มและการลดตัวประมวลผล และหน่วยความจำ

เครื่องมือสำหรับการมอนิเตอร์ผลการทำงาน เช่น **curt**, **splat**, **filemon**, **netpmon**, **tprof** และ **pprof** ไม่ได้ออกแบบมาเพื่อจัดการกับกิจกรรม DR เครื่องมือเหล่านี้จะขึ้นอยู่กับตัวประมวลผลแบบสแตติกหรือจำนวนของหน่วยความจำ ในบางกรณี การดำเนินการ DR ที่ดำเนินการในขณะที่เครื่องมือกำลังรันอยู่อาจไม่มีผลกระทบใดๆ ตัวอย่างเช่น เครื่องมือ **tprof** และ **pprof** อย่างไรก็ตาม การดำเนินการ DR อาจส่งผลทำให้เกิดลักษณะการทำงานที่ไม่ได้กำหนดไว้ หรือมีผลทำให้ได้รับผลลัพธ์ที่ไม่ถูกต้องของเครื่องมืออื่นๆ เช่น **curt**

มีเครื่องมือต่างๆ ที่สนับสนุนการดำเนินการ DR เครื่องมือเหล่านั้นได้รับการออกแบบมาเพื่อจดจำการเปลี่ยนแปลง และปรับเปลี่ยนการรายงานอย่างสอดคล้องกัน เครื่องมือที่จัดเตรียมส่วนสนับสนุน DLPAR มีดังต่อไปนี้:

topas ไม่มีการเปลี่ยนแปลงในอินเตอร์เฟสหรือการตีความของเอาต์พุต เมื่อคุณดำเนินการกับการดำเนินการ DR เครื่องมือ **topas** จะจดจำการเพิ่มขึ้นและการลดรีซอร์ส และจะรายงานข้อมูลสถิติที่เหมาะสม ตามชุดของรีซอร์สใหม่

sar, vmstat และ iostat
ไม่มีการเปลี่ยนแปลงในอินเตอร์เฟสหรือการตีความของเอาต์พุต ของคำสั่งเหล่านี้ เมื่อคุณดำเนินการกับการดำเนินการ DR ข้อความจะถูกส่งไปยังคุณ และแจ้งให้คุณทราบถึงการเปลี่ยนแปลงคอนฟิกูเรชัน ดังนั้น เครื่องมือจะเริ่มต้นการรายงานข้อมูลสถิติที่เหมาะสม ตามชนิดของรีซอร์สใหม่

rmss ไม่มีการเปลี่ยนแปลงกับการเรียกใช้งานของคำสั่งนี้ และยังคงทำงานตามที่คาดการณ์ไว้ หากการดำเนินการ DR เกิดขึ้นในขณะที่เครื่องมือ **rmss** กำลังรันอยู่

คำแนะนำเกี่ยวกับ DLPAR สำหรับการเพิ่มไมโครโพรเซสเซอร์หรือหน่วยความจำ

มีหลายวิธีในการพิจารณาเมื่อเกิดการขาดแคลนหน่วยความจำหรือตัวประมวลผล หรือเมื่อสล็อต I/O ต้องถูกเพิ่มเข้ามา

เมื่อคุณลบหน่วยความจำออกจากพาร์ติชัน การดำเนินการ DR จะประสบผลสำเร็จ แม้ว่าจะมีฟิสิคัลหน่วยความจำที่ไม่เพียงพอต่อความพร้อมใช้งานเพื่อรับหน่วยความขาออก ซึ่งจะมีการจัดเตรียมพื้นที่การเพจที่เพียงพอแทนการใช้หน่วยความฟิสิ

คัล ดังนั้น จึงเป็นสิ่งสำคัญที่จะมอนิเตอร์ข้อมูลสถิติการเพจของพาร์ติชัน ก่อนและหลังการลบหน่วยความจำ DR ตัวจัดการหน่วยความจำเสมือนต้องสร้างขึ้นเพื่อจัดการกับการเพจ อย่างไรก็ตาม การเพจที่เกินปกติสามารถนำไปสู่การลดระดับผลการทำงานลง

คุณสามารถใช้คำแนะนะที่พร้อมใช้งานใน “ผลการทำงานของหน่วยความจำ” ในหน้า 139 และ “ผลการทำงานของไมโครโพรเซสเซอร์” ในหน้า 112 เพื่อพิจารณาเมื่อเกิดการขาดแคลนหน่วยความจำหรือตัวประมวลผล คุณสามารถมองเห็นคำแนะนำที่มีอยู่ใน “ประสิทธิภาพเครือข่าย” ในหน้า 282 เพื่อพิจารณาถึง เวลาที่สล็อต I/O ต้องถูกเพิ่ม คำแนะนำเหล่านี้ยังสามารถนำมาใช้ เพื่อจำกัดผลกระทบของการลดหนึ่งในรีซอร์สเหล่านี้

Micro-Partitioning

โลจิคัลพาร์ติชันให้คุณรันหลายระบบปฏิบัติการ บนระบบเดียวกันโดยไม่ต้องมีการแทรกแซง ในเวอร์ชันก่อนหน้า ของ AIX คุณไม่สามารถแบ่งใช้ตัวประมวลผลกับพาร์ติชัน อื่น คุณสามารถใช้พาร์ติชันตัวประมวลผลที่แบ่งใช้ หรือ SPLPAR ที่ เรียกอีกอย่างว่า Micro-Partitioning

Micro-Partitioning ที่เป็นจริง

Micro-Partitioning จะแม็พตัวประมวลผลเสมือนกับตัวประมวลผลฟิสิคัล และตัวประมวลผลเสมือน จะถูกกำหนดให้กับพาร์ติชันแทนการกำหนดให้กับตัวประมวลผลฟิสิคัล

คุณสามารถใช้ Hypervisor เพื่อระบุเปอร์เซ็นต์ของการใช้ตัวประมวลผล และให้สิทธิกับพาร์ติชันที่แบ่งใช้ ซึ่งถูกกำหนดไว้เป็นการให้สิทธิ์ การให้สิทธิ์ตัวประมวลผลต่ำสุดคือสิบเปอร์เซ็นต์

คุณจะเข้าใจถึงประโยชน์ต่อไปนี้ด้วย Micro-Partitioning:

- การใช้ประโยชน์จากรีซอร์สมากที่สุด
- การนำไปใช้งานของเซิร์ฟเวอร์ใหม่อย่างรวดเร็ว
- การแยกกันของแอ็พพลิเคชัน

Micro-Partitioning จะพร้อมใช้งานบนระบบ System p5 ซึ่งมีความเป็นไปได้ ที่จะรันพาร์ติชันที่หลากหลายด้วยระดับของระบบปฏิบัติการที่ต่างกัน แต่คุณสามารถใช้ Micro-Partitioning ได้บนพาร์ติชันที่รัน AIX Version 6.1 หรือรุ่นถัดมาได้เท่านั้น

ด้วยเซิร์ฟเวอร์ IBM eServer p5 คุณสามารถเลือกชนิดของพาร์ติชันต่อไปนี้ได้จาก Hardware Management Console หรือ HMC:

ตัวประมวลผลพาร์ติชันเฉพาะงาน

ถ้าคุณใช้ตัวประมวลผลพาร์ติชันเฉพาะงาน ตัวประมวลผลทั้งหมดจะถูกกำหนดให้กับ โลจิคัลพาร์ติชันโดยเฉพาะ

และ จำนวนของความสามารถในการประมวลผลบนพาร์ติชันจะถูกจำกัดไว้โดยจำนวนความสามารถในการประมวลผลทั้งหมด ของตัวประมวลผลที่ปรับแต่งอยู่ในพาร์ติชันนั้น และไม่สามารถใช้เกินความสามารถนี้ได้ เว้นเสียแต่คุณเพิ่มตัวประมวลผลเพิ่มเติมภายในพาร์ติชัน โดยใช้การดำเนินการ DLPAR

พาร์ติชันตัวประมวลผลแบบแบ่งใช้

ถ้าคุณใช้พาร์ติชันตัวประมวลผลแบบแบ่งใช้ จะมีการทำเสมือนตัวประมวลผล ฟิสิคัล จากนั้นกำหนดให้กับพาร์ติชันต่างๆ

ตัวประมวลผลเสมือนมีความสามารถในช่วงตั้งแต่ 10 เปอร์เซ็นต์ของตัวประมวลผล ฟิสิคัล จนถึง 100 เปอร์เซ็นต์ของตัวประมวลผล ดังนั้น ระบบจึงสามารถมีได้หลายพาร์ติชัน ซึ่งแบ่งใช้ตัวประมวลผลเดียวกันและแบ่งความสามารถในการประมวลผล ระหว่างพาร์ติชันกันเอง จำนวนสูงสุดของตัวประมวลผลเสมือนต่อพาร์ติชันคือ 64 หากต้องการข้อมูลเพิ่มเติม ให้ดู “การจัดการตัวประมวลผลเสมือนภายในพาร์ติชันหนึ่ง” ในหน้า 397

การนำไปใช้งานของ Micro-Partitioning

เนื่องด้วย LPAR คุณสามารถนิยามพาร์ติชันใน Micro-Partitioning ด้วย HMC

ตารางต่อไปนี้แสดงชนิดของตัวประมวลผลที่แตกต่างกัน ซึ่งคุณสามารถใช้ได้กับ Micro-Partitioning:

ชนิดของตัวประมวลผล	คำอธิบาย
ตัวประมวลผลแบบฟิสิคัล	ตัวประมวลผลแบบฟิสิคัลคือ ฮาร์ดแวร์รีซอร์สจริง ซึ่งแสดงจำนวนของตัวประมวลผลหลัก ซึ่งไม่ใช่จำนวนของชิปตัวประมวลผล แต่ละชิปมีสองตัวประมวลผลหลัก จำนวนสูงสุดของฟิสิคัลโพรเซสเซอร์คือ 64 บนระบบแบบอิง POWER5
ตัวประมวลผลแบบโลจิคัล	ตัวประมวลผลแบบโลจิคัลคือ มุมมองของหน่วยตัวประมวลผลที่ถูกจัดการ จำนวนสูงสุดของตัวประมวลผลแบบโลจิคัล ที่มีขนาด 128
ตัวประมวลผลเสมือน	ตัวประมวลผลเสมือนคือหน่วยที่แสดงเปอร์เซ็นต์ของตัวประมวลผลแบบโลจิคัล ที่แบ่งใช้โดยพาร์ติชันที่ต่างกัน จำนวนสูงสุดของตัวประมวลผลเสมือนคือ 64

เมื่อคุณสร้างพาร์ติชัน คุณต้องเลือกว่าคุณต้องการสร้างพาร์ติชันตัวประมวลผลที่แบ่งใช้ หรือพาร์ติชันตัวประมวลผลเฉพาะงาน ซึ่งเป็นไปไม่ได้ที่จะมีทั้งตัวประมวลผลที่แบ่งใช้และตัวประมวลผลเฉพาะงาน ในหนึ่งพาร์ติชัน หากต้องการเปิดใช้งานการแบ่งใช้ตัวประมวลผล คุณต้องปรับแต่งอ็อพชันต่อไปนี้:

- โหมดการแบ่งใช้ตัวประมวลผล: Capped หรือ Uncapped[1]
- ความสามารถในการประมวลผล: หนัก[2]
- จำนวนของตัวประมวลผลเสมือน: ต้องการ ต่ำสุด สูงสุด

หมายเหตุ: โหมด Capped หมายความว่า ความสามารถในการประมวลผลที่ไม่เกินกว่าความสามารถที่กำหนดไว้ และโหมด uncapped หมายความถึง ความสามารถในการประมวลผลที่สามารถมีค่าเกินได้ เมื่อพูลการประมวลผลที่แบ่งใช้มีรีซอร์สที่พร้อมใช้งาน

หมายเหตุ: ความสามารถในการประมวลผลนี้ถูกระบุในเรื่องของหน่วยการประมวลผล ที่ถูกวัดจากเศษของตัวประมวลผลขนาด 0.01 ดังนั้น หากกำหนดความสามารถในการประมวลผลสำหรับครึ่งหนึ่งของตัวประมวลผล คุณต้องระบุหน่วยการประมวลผล 0.50 บน HMC

ผลกระทบของผลการทำงาน Micro-Partitioning

คุณอาจมีประสบการณ์เกี่ยวกับผลกระทบในด้านบวกหรือลบเกี่ยวกับผลการทำงานของ Micro-Partitioning

ข้อได้เปรียบของ Micro-Partitioning คือ อนุญาตให้มีการใช้ประโยชน์โดยรวมที่เพิ่มขึ้นของรีซอร์สของระบบ โดยการใช้จำนวนของรีซอร์สตัวประมวลผลที่ต้องการเท่านั้นที่แต่ละพาร์ติชันต้องการ แต่ เนื่องจากการใช้ที่เชื่อมโยงกับการรักษาตัวประมวลผลเสมือนออนไลน์ ให้พิจารณาข้อกำหนดเกี่ยวกับความสามารถเมื่อเลือกค่าสำหรับแอ็ตทริบิวต์

สำหรับผลการทำงานสูงสุด ตรวจสอบให้มั่นใจว่า คุณได้สร้างจำนวนของพาร์ติชันเพียงเล็กน้อย ซึ่งลดการใช้ของการกำหนดตารางเวลาตัวประมวลผลเสมือน

แอ็พพลิเคชันที่ใช้ CPU เช่น แอ็พพลิเคชันการคำนวณผลการทำงานสูง อาจไม่เหมาะสมสำหรับสภาวะแวดล้อม Micro-Partitioning ถ้าแอ็พพลิเคชันใช้ความสามารถของการประมวลผลที่ให้สิทธิในระหว่างการประมวลผล คุณควรใช้พาร์ติชันที่มีตัวประมวลผลเฉพาะงานเพื่อจัดการกับความต้องการของแอ็พพลิเคชัน

Active Memory Expansion (AME)

Active Memory Expansion (AME) คือเทคโนโลยีใหม่สำหรับการขยาย ความจุหน่วยความจำที่มีประสิทธิภาพของระบบ AME ใช้เทคโนโลยีการบีบอัดหน่วยความจำ เพื่อบีบอัดข้อมูลในหน่วยความจำอย่างโปร่งใส เพื่อให้สามารถวางข้อมูล ในหน่วยความจำได้มากขึ้น ส่งผลให้ขยายความจุหน่วยความจำของ ระบบที่ตั้งค่าคอนฟิก

ภาพรวม

Active Memory Expansion (AME) ขึ้นอยู่กับการบีบอัดของข้อมูลในหน่วยความจำเพื่อเพิ่มจำนวนของข้อมูลที่สามารถเก็บหน่วยความจำ และขยายประสิทธิภาพของความจุของหน่วยความจำของระบบ POWER7 ที่ใช้ตัวประมวลผล การบีบอัดข้อมูลในหน่วยความจำ อยู่ภายใต้การจัดการโดยระบบปฏิบัติการ และการบีบอัดนี้โปร่งใสสำหรับ แอ็พพลิเคชันและผู้ใช้ AME สามารถตั้งค่าคอนฟิกได้สำหรับแต่ละโลจิคัลพาร์ติชัน (LPAR) ดังนั้น จึงสามารถเลือกเปิดใช้งาน AME สำหรับหนึ่งหรือหลาย LPARs บนระบบได้

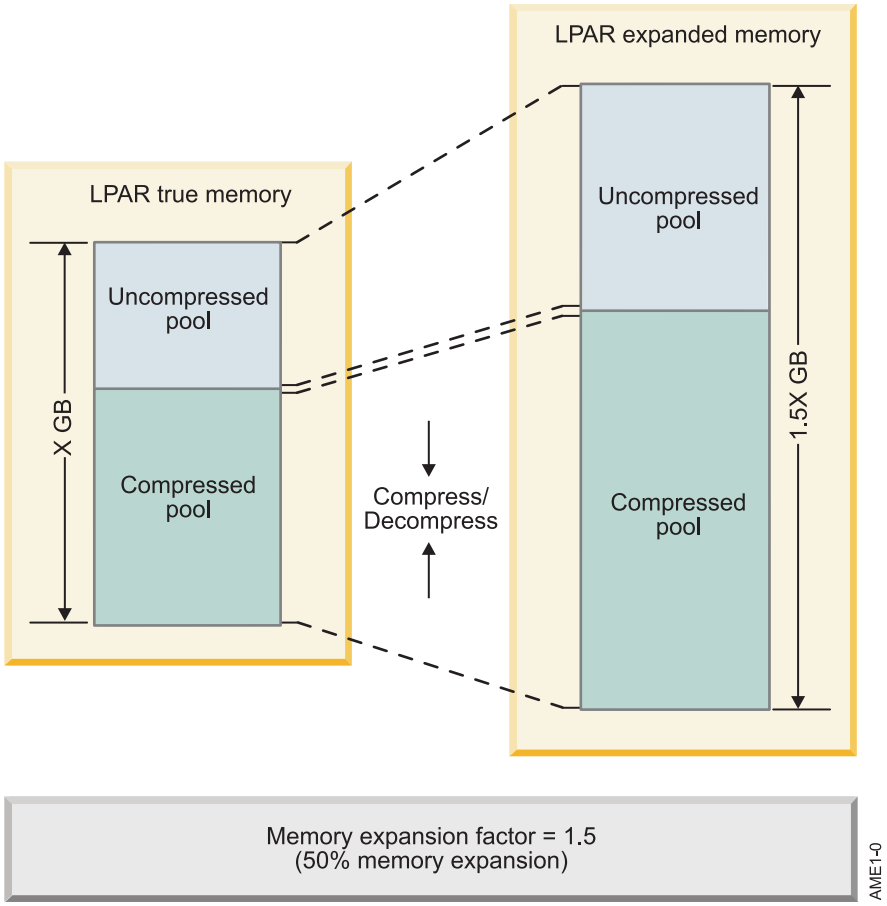
เมื่อเปิดใช้งาน Active Memory Expansion สำหรับ LPAR ระบบปฏิบัติการจะบีบอัดส่วนของหน่วยความจำของ LPAR และปล่อยหน่วยความจำส่วนที่เหลือไว้เป็นแบบไม่บีบอัด การทำเช่นนี้ ส่งผลให้หน่วยความจำถูกแบ่งออกเป็นสองพูลอย่างมีประสิทธิภาพ สองพูลคือ:

- พูลที่บีบอัด
- พูลที่ไม่บีบอัด

ระบบปฏิบัติการจะบีบอัดหน่วยความจำในจำนวนที่แตกต่างกันไป ไม่แน่นอน ทั้งนี้ขึ้นอยู่กับเวิร์กโหลดและการตั้งค่าคอนฟิกของ LPAR

ระบบปฏิบัติการย้ายข้อมูลระหว่างพูลหน่วยความจำที่บีบอัด และที่ไม่บีบอัดตามรูปแบบการเข้าถึงหน่วยความจำของ แอ็พพลิเคชัน เมื่อแอ็พพลิเคชันต้องการเข้าถึงข้อมูลที่บีบอัด ระบบปฏิบัติการจะยกเลิกบีบอัดข้อมูลโดยอัตโนมัติ และย้ายข้อมูลนั้นจากพูลที่บีบอัดไปยังพูลที่ไม่บีบอัด เพื่อทำให้ข้อมูลนั้น พร้อมใช้งานสำหรับแอ็พพลิเคชัน เมื่อพูลที่ไม่บีบอัดเต็ม ระบบปฏิบัติการจะบีบอัดข้อมูลและย้ายข้อมูลนั้นจากพูลที่ไม่บีบอัด ไปยังพูลที่บีบอัด

กิจกรรมการบีบอัดและการยกเลิกบีบอัดนี้โปร่งใสสำหรับแอ็พพลิเคชัน เนื่องจาก AME ใช้การบีบอัดหน่วยความจำ ดังนั้นจึงมีการใช้ CPU เพิ่มเติมบางส่วนเมื่อ ใช้งาน AME จำนวน CPU เพิ่มเติมที่ต้องใช้สำหรับ AME แตกต่างกันไปขึ้นอยู่กับเวิร์กโหลดและระดับของการขยายหน่วยความจำ ที่ใช้อยู่



หมายเหตุ: เมื่อเปิดใช้งาน AME ระบบปฏิบัติการ AIX ใช้เพจ 4 KB

ปัจจัยการขยายหน่วยความจำและขนาดหน่วยความจำที่ขยาย

เมื่อ ตั้งค่าคอนฟิก Active Memory Expansion มีอ็อพชันการตั้งค่าคอนฟิกอย่างเดียว ที่ต้องตั้งค่าสำหรับ LPAR นั่นคือปัจจัยการขยาย หน่วยความจำ ปัจจัยการขยายหน่วยความจำของ LPAR ระบุความจุหน่วยความจำ ที่มีประสิทธิภาพเป้าหมายของ LPAR ความจุหน่วยความจำเป้าหมายนี้ บ่งชี้ให้ระบบปฏิบัติการทราบว่า จะบีบอัดหน่วยความจำเป็นจำนวน เท่าใด ความจุหน่วยความจำเป้าหมายที่ระบุ มีการอ้างอิงเป็นขนาดหน่วยความจำที่ขยาย ปัจจัยการขยายหน่วยความจำ มีการระบุเป็นตัวคูณของขนาดหน่วยความจำจริงของ LPAR

```
LPAR_expanded_mem_size = LPAR_true_mem_size * LPAR_mem_exp_factor
```

ตัวอย่าง เช่น การใช้ปัจจัยการขยายหน่วยความจำเป็น 2.0 สำหรับ LPAR บ่งชี้ว่า ต้องใช้การบีบอัดหน่วยความจำเป็นสองเท่าของความจุหน่วยความจำของ LPAR หาก LPAR มีการตั้งค่าคอนฟิกด้วยปัจจัยการขยายหน่วยความจำเป็น 2.0 และขนาดความจำคือ 20 GB ผลคือขนาดหน่วยความจำที่ขยายของ LPAR จะเป็น 40 GB

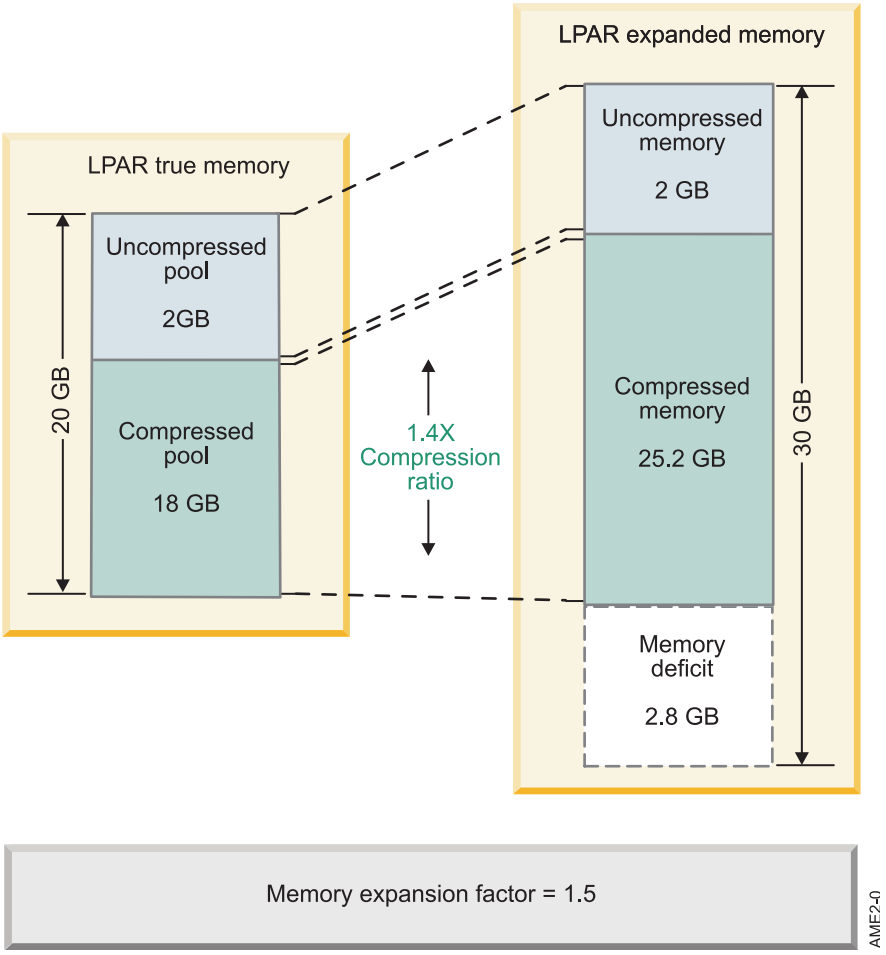
```
40 GB = 20 GB * 2.0
```

ระบบ ปฏิบัติการจะบีบอัดข้อมูลในหน่วยความจำให้เพียงพอที่จะจัดเก็บข้อมูลจำนวน 40 GB ไว้ในพื้นที่ข้อมูล 20 GB เดิม ปัจจัยการขยายหน่วยความจำและขนาดหน่วยความจำ ที่ขยายสามารถเปลี่ยนแปลงได้แบบไดนามิกที่รันไทม์โดยใช้ Hardware Management Console (HMC) ผ่านทางการดำเนินงานไดนามิก LPAR ขนาดหน่วยความจำที่ ขยายถูกปัดเศษลงให้เป็นตัวคูณ logical memory block (LMB) ซึ่งใกล้เคียงที่สุดเสมอ

การขาดหน่วยความจำ

เมื่อตั้งค่าคอนฟิกปัจจัยการขยาย หน่วยความจำสำหรับ LPAR อาจเป็นไปได้ที่เลือกปัจจัยการขยาย หน่วยความจำที่ใหญ่เกินไปและการบีบอัดเวิร์กโหลดไม่สามารถทำให้บรรลุผล ตามปัจจัยที่เลือกนั้นได้ เมื่อปัจจัยการขยายหน่วยความจำ สำหรับ LPAR ใหญ่เกินไป การขาดการขยายหน่วยความจำจะเกิดขึ้น เพื่อบ่งชี้ว่า LPAR ไม่สามารถทำให้สำเร็จตามเป้าหมายปัจจัยการขยายหน่วยความจำ ตัวอย่างเช่น ถ้า LPAR มีการตั้งค่าคอนฟิกด้วยขนาดหน่วยความจำ จำนวน 20 GB และปัจจัยการขยายหน่วยความจำคือ 1.5 ซึ่งส่งผลให้ขนาด หน่วยความจำที่ขยายเป้าหมายทั้งหมดเป็น 30 GB อย่างไรก็ตาม เวิร์กโหลดที่กำลังรันใน LPAR บีบอัดได้ไม่ดีนัก และข้อมูลของเวิร์กโหลด บีบอัดในอัตราส่วนเพียง 1.4 ต่อ 1 เท่านั้น ในกรณีนี้ เวิร์กโหลด ไม่สามารถบรรลุผลตามปัจจัยการขยายหน่วยความจำเป้าหมาย จำนวน 1.5 ได้ ระบบปฏิบัติการจะจำกัดจำนวนของหน่วยความจำฟิสิคัล ที่สามารถใช้ในพูลที่บีบอัดเป็นสูงสุดไม่เกิน 95% ในตัวอย่างนี้ ถ้า 19 GB ถูกสำรองไว้สำหรับบีบอัดขนาดหน่วยความจำที่ขยายสูงสุด ซึ่งสามารถบรรลุผลได้จะเป็น 27.6 GB ผลคือมีหน่วยความจำ ขาดไป 2.4 GB ส่วนที่ขาดไปนี้มีการอ้างอิงเป็นการขาด หน่วยความจำ

ผลกระทบของการขาดหน่วยความจำเหมือนกับผลกระทบของ การตั้งค่าคอนฟิก LPAR ด้วยหน่วยความจำน้อยเกินไป เมื่อเกิดการขาดหน่วยความจำ ระบบปฏิบัติการไม่สามารถดำเนินการตามเป้าหมายหน่วยความจำที่ขยาย ซึ่งตั้งค่าคอนฟิกสำหรับ LPAR และระบบปฏิบัติการอาจต้องใช้ การ paging out หน้าหน่วยความจำเสมือนในพื้นที่ว่าง paging ดังนั้น ในตัวอย่าง ที่ระบุข้างบน ถ้าเวิร์กโหลดใช้หน่วยความจำมากกว่า 27.6 GB ระบบปฏิบัติการอาจเริ่มต้นการ paging out หน้าหน่วยความจำเสมือน ในพื้นที่ว่าง paging เพื่อบ่งชี้ว่าเวิร์กโหลด สามารถมีขนาดหน่วยความจำที่ขยายตามที่ตั้งค่าคอนฟิกได้หรือไม่ ระบบปฏิบัติการ จะรายงานเมทริกการขาดหน่วยความจำ นี่เป็น “ช่องโหว่” ในขนาดหน่วยความจำ ที่ขยายซึ่งไม่สามารถทำให้บรรลุผลได้ หากการขาดนี้เป็นศูนย์ แสดงว่าสามารถทำให้บรรลุผล ตามปัจจัยการขยายหน่วยความจำเป้าหมายได้ และปัจจัยการขยายหน่วยความจำ ของ LPAR มีการตั้งค่าคอนฟิกอย่างถูกต้อง หากเมทริกการขาดหน่วยความจำ ที่ขยายไม่เป็นศูนย์ แสดงว่าเวิร์กโหลดขาดแคลนขนาดหน่วยความจำที่ขยาย เป็นจำนวนเท่ากับค่าของการขาด เพื่อตัดปัญหาการขาด หน่วยความจำควรลดปัจจัยการขยายหน่วยความจำของ LPAR ลง อย่างไรก็ตาม การลดปัจจัยการขยายหน่วยความจำจะลด ขนาดหน่วยความจำที่ขยายของ LPAR ดังนั้นจึงควรรักษาขนาดหน่วยความจำที่ขยายของ LPAR ไว้เท่าเดิม ต้องลดปัจจัยการขยายหน่วยความจำ และต้องเพิ่มหน่วยความจำให้มากขึ้นลงใน LPAR ทั้งขนาดหน่วยความจำของ LPAR และปัจจัยการขยายหน่วยความจำสามารถเปลี่ยนแปลงได้แบบไดนามิก



ข้อควรพิจารณาในการวางแผน

ก่อนนำเวิร์กโหลดไปใช้ในสภาวะแวดล้อม Active Memory Expansion (AME) จำเป็นต้องมี การวางแผนเบื้องต้นเพื่อให้แน่ใจว่าเวิร์กโหลดจะได้รับประโยชน์สูงสุด จาก AME ประโยชน์ของ AME ที่มีต่อเวิร์กโหลดแตกต่างกันไป ขึ้นอยู่ กับลักษณะของเวิร์กโหลด บางเวิร์กโหลดสามารถขยายหน่วยความจำ ได้ในระดับที่สูงกว่าเวิร์กโหลดอื่น เครื่องมือการวางแผนและการให้คำปรึกษาเกี่ยวกับ Active Memory Expansion **amepat** ช่วยในการวางแผน การนำเวิร์กโหลดไปใช้ในสภาวะแวดล้อม Active Memory Expansion และแสดงแนวทางระดับของการขยายหน่วยความจำที่เวิร์กโหลด สามารถทำได้

เครื่องมือการวางแผน AME

เครื่องมือการวางแผน AME (ตั้งอยู่ใน `/usr/bin/amepat`) มีวัตถุประสงค์หลักสอง ข้อ ดังนี้

- เพื่อวางแผนการตั้งค่าคอนฟิก Active Memory Expansion แรกเริ่ม
- เพื่อมอนิเตอร์และปรับการตั้งค่าคอนฟิก AME ที่ใช้งานอยู่โดยละเอียด

เครื่องมือการวางแผน AME สามารถรันบน LPARs โดยที่เปิดใช้งานหรือไม่เปิดใช้งาน AME ก็ได้ใน LPAR ที่ยังไม่ได้เปิดใช้งาน AME ให้รัน **amepat** กับ เวิร์กโหลดตัวแทน ควรตั้งค่าให้มอนิเตอร์เวิร์กโหลดนั้น ในรอบระยะเวลาที่สำคัญ ตัวอย่างเช่น ตั้งค่าเครื่องมือ **amepat** เพื่อ รันในช่วงเวลาที่มีการใช้รีซอร์สสูงสุดของ เวิร์กโหลด หลังจากเสร็จสมบูรณ์แล้ว เครื่องมือจะแสดงรายงานปัจจัยการขยายหน่วย ความจำที่เป็นไปได้หลายรูปแบบและการใช้ CPU ที่คาดไว้ ซึ่งเกิดจาก AME สำหรับแต่ละปัจจัย นอกจากนี้ เครื่องมือยังแสดงปัจจัยการขยายหน่วยความจำที่ แนะนำซึ่งช่วยประหยัดหน่วยความจำได้สูงสุดในขณะที่ใช้

CPU เพิ่มเติมน้อยที่สุดรายงานและคำแนะนำเหล่านี้ สามารถเป็นการตั้งค่าคอนฟิกแรกเริ่มที่มีประโยชน์สำหรับการนำ AME ไปใช้ใน LPAR ที่มีการเปิดใช้งาน AME **amepat** มีวัตถุประสงค์คล้ายกัน เมื่อรันในเวลาที่มีการใช้งานสูงสุดของเวิร์กโหลดตัวแทน เครื่องมือแสดง รายงานการใช้ CPU จริงซึ่งเกิดจาก AME สำหรับ ปัจจัยการขยายหน่วยความจำปัจจุบัน นอกจากนี้ ยังแสดงข้อมูลการขาดหน่วยความจำด้วย ถ้ามีอยู่ เนื่องจากมีการเปิดใช้งาน AME เครื่องมือยังแสดง ตัวแทนที่ถูกต้องมากขึ้นของระดับการใช้ CPU ซึ่งสามารถคาดการณ์ได้ ที่ปัจจัยการขยายหน่วยความจำต่างๆ จะมีการแสดงคำแนะนำใหม่ ที่พิจารณาจากข้อมูลนี้ให้แก่ผู้ใช้

ตัวอย่าง ของรายงานที่สร้างขึ้นในพาร์ติชันที่ปิดใช้งาน AME และตัวอย่าง เวิร์กโหลดมีดังนี้:

```
# amepat 5 2

Command Invoked               : amepat 2 5

Date/Time of invocation       : Wed Dec  2 11:29:29 PAKST 2009
Total Monitored time          : 10 mins 58 secs
Total Samples Collected       : 5

System Configuration:
---------------------
Partition Name                : aixfvt19
Processor Implementation Mode : POWER5
Number Of Logical CPUs        : 8
Processor Entitled Capacity   : 4.00
Processor Max. Capacity       : 4.00
True Memory                   : 4.25 GB
SMT Threads                   : 2
Shared Processor Mode         : Disabled
Active Memory Sharing         : Disabled
Active Memory Expansion       : Disabled

System Resource Statistics:   Average       Min         Max
---------------------------   -----------   ----------  -----------
CPU Util (Phys. Processors)   2.00 [ 50%]   1.00 [ 25%] 3.00 [ 75%]
Virtual Memory Size (MB)      1366 [ 31%]   1113 [ 26%] 2377 [ 55%]
True Memory In-Use (MB)       1758 [ 40%]   1234 [ 28%] 3834 [ 88%]
Pinned Memory (MB)            673 [ 15%]    673 [ 15%]  675 [ 16%]
File Cache Size (MB)          391 [  9%]    124 [  3%] 1437 [ 33%]
Available Memory (MB)         841 [ 65%]    1812 [ 42%] 3099 [ 71%]

Active Memory Expansion Modeled Statistics
------------------------------------------
Modeled Expanded Memory Size  :   4.25 GB
Average Compression Ratio     :   5.29

Expansion    Modeled True     Modeled             CPU Usage
Factor       Memory Size      Memory Gain         Estimate
---------    -------------    -----------------   -----------
     1.00         4.25 GB         0.00 KB [  0%]   0.00 [  0%]
     1.31         3.25 GB         1.00 GB [ 31%]   0.34 [  8%]
     1.55         2.75 GB         1.50 GB [ 55%]   0.39 [ 10%]
     1.89         2.25 GB         2.00 GB [ 89%]   0.45 [ 11%]
     2.12         2.00 GB         2.25 GB [112%]   0.50 [ 12%]
     2.43         1.75 GB         2.50 GB [143%]   0.65 [ 16%]
     2.83         1.50 GB         2.75 GB [183%]   0.70 [ 18%]
```

```
Active Memory Expansion Recommendation:
---------------------------------------
The recommended AME configuration for this workload is to configure the LPAR
with a memory size of 1.50 GB and to configure a memory expansion factor
of 2.83.  This will result in a memory gain of 183%. With this
configuration, the estimated CPU usage due to AME is approximately 0.50
physical processors, and the estimated overall peak CPU resource required for
the LPAR is 3.50 physical processors.

NOTE: amepat's recommendations are based on the workload's utilization level
during the monitored period. If there is a change in the workload's utilization
level or a change in workload itself, amepat should be run again.

The modeled Active Memory Expansion CPU usage reported by amepat is just an
estimate.  The actual CPU usage used for Active Memory Expansion may be lower
or higher depending on the workload.
```

รายงานประกอบด้วยส่วนทั้งหมด หกส่วนที่มีการอธิบายตามลำดับ

ข้อมูลคำสั่ง

ส่วนนี้แสดง รายละเอียดเกี่ยวกับอาร์กิวเมนต์ที่ส่งผ่านไปยัง **amepat**, เวลาการเรียกใช้ เวลาทั้งหมดที่ระบบถูกมอนิเตอร์ และจำนวนตัวอย่างที่รวบรวม ในรายงาน **amepat**

มีการเรียกใช้นาน 10 นาที โดยแบ่งออกเป็นช่วงละ 2 นาทีสำหรับ 5 ตัวอย่าง

หมายเหตุ: สามารถสังเกตได้ว่า เวลาที่มอนิเตอร์ทั้งหมดแสดงค่าเป็น 10 นาทีและ 58 วินาที 58 วินาทีที่เพิ่มขึ้นมาใช้สำหรับการรวบรวมสถิติระบบที่ต้องการ สำหรับการจัดทำโมเดล Active Memory Expansion

การตั้งค่าคอนฟิกระบบ

ส่วนนี้แสดง ข้อมูลการตั้งค่าคอนฟิก เช่น ชื่อโฮสต์ สถาปัตยกรรมตัวประมวลผล, CPUs, การให้สิทธิ ขนาดหน่วยความจำจริง สภาพ SMT, ตัวประมวลผล และโหมด หน่วยความจำ ในรายงานข้างบน สถานะที่ปิดใช้งานของ Active Memory Expansion บ่งชี้ว่ามีการเรียกใช้ **amepat** ในพาร์ติชันที่ปิดใช้งาน AME

หมายเหตุ: เครื่องมือ **amepat** ยังสามารถเรียกใช้ในพาร์ติชันที่เปิดใช้งาน AME ได้ เช่นกันเพื่อมอนิเตอร์และปรับการตั้งค่าคอนฟิก AME ที่ใช้งานอยู่โดยละเอียด ในพาร์ติชัน ที่เปิดใช้งาน AME ส่วนการตั้งค่าคอนฟิกระบบยังแสดง ขนาดหน่วยความจำที่ขยายเป้าหมายและปัจจัยการขยายหน่วยความจำเป้าหมายด้วย

สถิติรีซอร์สระบบ

ส่วนนี้แสดง การใช้ประโยชน์รีซอร์สระบบในรอบระยะเวลาที่มอนิเตอร์ ส่วนนี้แสดงค่าเฉลี่ย ค่าต่ำสุด และค่าสูงสุดควบคู่ไปกับเปอร์เซ็นต์ที่สอดคล้องกัน ของการใช้รีซอร์สรีซอร์ส

ในรายงานที่ ระบุ เวิร์กโหลดใช้ตัวประมวลผลฟิสิคัลโดยเฉลี่ย 2.00 (ดังแสดงในแถว CPU Util) ซึ่งคิดเป็น 50% ของความสามารถฟิสิคัลสูงสุด (4.00 ดังแสดงในแถว Processor Max Capacity) ของ LPAR

หมายเหตุ: รายงาน การใช้ CPU รวมหน่วยประมวลผลที่ใช้สำหรับ AME Modeling ในพาร์ติชันที่เปิดใช้งาน AME มีการแสดงการใช้ CPU ที่เป็นผลมาจาก กิจกรรมการบีบอัดหรือการยกเลิกบีบอัดเช่นกัน

ใน รายงานที่ระบุข้างบน ยังสามารถสังเกตการใช้หน่วยความจำของ เวิร์กโหลดได้ด้วย เปอร์เซ็นต์หน่วยความจำทั้งหมดสัมพันธ์กับ ขนาดหน่วยความจำจริงทั้งหมดของ LPAR

หมายเหตุ: เปอร์เซ็นต์ที่สูงของ Pinned Memory และ File Cache บ่งชี้ว่าเวิร์กโหลดอาจไม่ได้รับประโยชน์มากนัก จาก AME

สถิติการขยายหน่วยความจำที่ใช้งานอยู่

ส่วนนี้ แสดงขึ้นเฉพาะถ้ามีการเรียกใช้เครื่องมือ **amepat** ใน LPAR ที่ เปิดใช้งาน AME เท่านั้น

ค้นหาตัวอย่างของตัวอย่างเอาต์พุต:

```
AME Statistics:                  Average        Min       Max
---------------                  -----------    --------  ---------
AME CPU Usage (Phy. Proc Units) 0.25 [  6%]   0.01 [  0%] 0.50 [ 13%]
Compressed Memory (MB)          264 [ 13%]    264 [ 13%]  264 [ 13%]
Compression Ratio               2.15          2.15        2.16
Deficit Memory Size (MB)        562 [ 55%]    562 [ 55%]   562 [ 55%]

This section of the report shows the AME CPU Usage, Compressed Memory,
Compression Ratio & Deficit Memory.
The Deficit Memory will be reported only if there is a memory deficit
in achieving the expanded memory size.
Otherwise the tool will not report this information.
For example in the above report, it can be observed that there is
an average memory deficit of 562 MB which is 55% of the Target
Expanded Memory Size of 2 GB (which is reported in the
System Configuration Section when AME is enabled).
The report also shows on an average 264 MB out of 2GB of expanded memory is in
compressed form as reported against Compressed Memory.
```

สถิติที่จัดทำโมเดล Active Memory Expansion

ส่วนนี้แสดง ขนาดหน่วยความจำที่ขยายซึ่งจัดทำโมเดล อัตราส่วนการบีบอัด และตารางที่มีตัวเลขของการตั้งค่าคอนฟิก AME ที่เป็นไปได้ ในรายงานที่ระบุ ขนาดหน่วยความจำที่ขยายซึ่งจัดทำโมเดลมีการรายงาน

เป็น 4.25 GB ซึ่งเป็นขนาดหน่วยความจำจริงของ LPAR โดยค่าดีฟอลต์ **amepat** ใช้ หน่วยความจำจริงของพาร์ติชันเป็นหน่วยความจำที่ขยายซึ่งจัดทำโมเดล ค่านี้สามารถ เปลี่ยนได้โดยใช้อ็อพชัน

-t หรือ **-a** รายงาน แสดงอัตราส่วนการบีบอัดเฉลี่ยเป็น 5.29 ซึ่งบ่งชี้ว่า เวิร์กโหลดบีบอัดได้ดี อัตราส่วนการบีบอัดใกล้กับ 1 อาจ บ่งชี้ว่าสามารถขยาย

หน่วยความจำได้เพียงเล็กน้อย การตั้งค่าคอนฟิกที่ แสดงในตารางการจัดทำโมเดลใช้ข้อมูล Modeled Expanded Memory Size เป็นหน่วยความจำเป้าหมาย

ตารางแสดงขนาดหน่วยความจำจริงที่ จัดทำโมเดล หน่วยความจำที่ได้รับที่จัดทำโมเดล และการใช้ CPU เพิ่มเติม จาก AME สำหรับปัจจัยการขยายต่างๆ

ตัวอย่างเช่น ให้ดูที่ แถวต่อไปนี้ในตาราง

```
1.55          2.75 GB         1.50 GB [ 55%]   0.39 [ 10%]
```

ที่นี่ ขนาดหน่วยความจำจริงดั้งเดิมจำนวน 4.25 GB สามารถบรรลุผลได้ด้วยขนาด หน่วยความจำฟิสิคัลจำนวน 2.75 GB และปัจจัยการขยาย 1.55 การตั้งค่าคอนฟิกนี้ อาจส่งผลให้การใช้ CPU เพิ่มขึ้นเป็น

0.39 ตัวประมวลผลฟิสิคัล (10% ของความสามารถสูงสุด)

คำแนะนำเกี่ยวกับ Active Memory Expansion

ส่วนนี้แสดง คำแนะนำเกี่ยวกับการตั้งค่าคอนฟิก Active Memory Expansion ที่จัดทำขึ้นสำหรับเวิร์กโหลด การตั้งค่าคอนฟิกที่ดีที่สุดจะไม่มีทางใช้ AME CPU เกิน 15% ส่วนนี้ยังแสดงข้อมูลเกี่ยวกับ การใช้ AME CPU และหน่วยความจำที่ได้รับสำหรับการตั้งค่าคอนฟิกที่แนะนำด้วย ในรายงานที่ระบุ สังเกตว่าปัจจัยการขยายที่แนะนำคือ 2.12 เป้าหมายการใช้ AME CPU สามารถเปลี่ยนได้โดยใช้ อ็อพชัน **-c** หรือ **-C**

หมายเหตุ: คำแนะนำทั้งหมดซึ่งจัดทำใน ส่วนนี้สร้างขึ้นจากข้อมูลเวิร์กโหลดที่กำลังรันอยู่ในรอบระยะเวลาที่ มอนิเตอร์การใช้ AME CPU ที่รายงานในส่วนนี้เป็นค่าประเมินและ อาจแตกต่างออกไปได้ เอาต์พุตการจัดทำโมเดลของ **amepat** ไม่ได้ประเมินการเปลี่ยน แปลงในปริมาณงานของเวิร์กโหลดหรือเวลาตอบกลับที่การขยายหน่วยความจำ ระดับต่างๆ ก่อนการนำเวิร์กโหลดไปใช้ในการผลิตร่วมกับ Active Memory Expansion ควรมอนิเตอร์การตั้งค่าคอนฟิกที่ใช้อย่างใกล้ชิด เพื่อให้แน่ใจว่าตรงตามเป้าหมายประสิทธิภาพของเวิร์กโหลด เมื่อ **amepat** รันในสภาวะแวดล้อมที่เปิดใช้งาน AME จะมีการจัดทำคำเตือนเมื่อขนาดหน่วยความจำที่ขยายขาดไป

หมายเหตุ: เอาต์พุต ปรากฏขึ้นดังนี้:

```
WARNING: This LPAR currently has a memory deficit of 562 MB.
A memory deficit is caused by a memory expansion factor that is too
high for the current workload.  It is recommended that you reconfigure
the LPAR to eliminate this memory deficit.  Reconfiguring the LPAR
with one of the recommended configurations in the above table should
eliminate this memory deficit.

The recommended AME configuration for this workload is to configure
the LPAR with a memory size of ...
```

สำหรับข้อมูลเพิ่มเติม เกี่ยวกับเรื่องเหล่านี้และการใช้เครื่องมือการวางแผน AME ในแบบอื่นๆ โปรด อ้างอิงหน้า **amepat** man

การมอนิเตอร์ประสิทธิภาพ

มีเครื่องมือประสิทธิภาพ AIX หลายอย่าง ที่สามารถใช้เพื่อมอนิเตอร์สถิติ Active Memory Expansion และ รวบรวมข้อมูลเกี่ยวกับ Active Memory Expansion

ตารางต่อไปนี้ สรุปเครื่องมือและอ็อพชันต่างๆ ที่สามารถใช้เพื่อ มอนิเตอร์สถิติ Active Memory Expansion:

เครื่องมือ	อ็อพชัน	คำอธิบาย
amepat	**-N**	แสดงสถิติการใช้ CPU และหน่วยความจำ ทั่วไป นอกจากนี้ ยังบ่งชี้การใช้ CPU ของกิจกรรมการบีบอัด และการยกเลิกบีบอัด AME ตลอดจนสถิติการบีบอัด การยกเลิกบีบอัด และการขาดหน่วยความจำ
vmstat	**-c**	แสดงสถิติการบีบอัด การยกเลิกบีบอัด และ สถิติการขาดหน่วยความจำ
lparstat	**-c**	บ่งชี้การใช้ CPU ของกิจกรรมการบีบอัด และการยกเลิกบีบอัด AME และแสดงข้อมูลการขาด หน่วยความจำด้วย
svmon	**-O summary=ame**	แสดงมุมมองสรุปของการใช้หน่วยความจำ ซึ่งแบ่งย่อยเป็นหน้าที่บีบอัดและหน้าที่ไม่ได้บีบอัด
topas		หน้าจอ topas ดีฟอลต์แสดงสถิติการบีบอัดหน่วยความจำ เมื่อรันในสภาวะแวดล้อม AME

คำสั่ง vmstat

คำสั่ง **vmstat** สามารถใช้ ร่วมกับอ็อพชัน **-c** ของคำสั่งเพื่อแสดงสถิติ AME

```
# vmstat -c 2 1

System configuration: lcpu=2 mem=1024MB tmem=512MB ent=0.40 mmode=dedicated-E

kthr                memory                          page          faults
 r   b     avm   fre    csz   cfr    dxm   ci   co  pi  po  in   sy   cs
 0   0  309635  2163  43332   943  26267  174  386   0   0  93  351  339

cpu
us sy id wa    pc    ec
2  3 89  7  0.02   5.3
```

ในเอาต์พุตข้างบน มีการแสดงสถิติการบีบอัดหน่วยความจำ ต่อไปนี้:

- ขนาดหน่วยความจำที่ขยาย **mem** ของ LPAR คือ 1024 MB
- ขนาดหน่วยความจำจริง **tmem** ของ LPAR คือ 512 MB
- โหมดหน่วยความจำ **mmode** ของ LPAR คือ Active Memory Sharing ที่ปิดใช้งานและ Active Memory Expansion ที่เปิดใช้งาน
- ขนาดพูลที่บีบอัด **csz** คือ 43332 4K- หน้า
- จำนวนของหน่วยความจำว่าง **cfr** ในพูลที่บีบอัดคือ 943 4K- หน้า
- ขนาดของการขาดหน่วยความจำที่ขยาย **dxm** คือ 26267 4K- หน้า
- จำนวนของการดำเนินงานบีบอัดหรือ page-outs ในพูลที่บีบอัด ต่อวินาที **co** คือ 386
- จำนวนของการดำเนินงานยกเลิกบีบอัดหรือ page-ins จากพูลที่บีบอัด ต่อวินาที **ci** คือ 174

คำสั่ง lparstat

คำสั่ง **lparstat** สามารถใช้ ร่วมกับอ็อพชัน **-c** ของคำสั่งเพื่อแสดงสถิติ AME

```
# lparstat -c 2 5

System configuration: type=Shared mode=Uncapped mmode=Ded-E smt=On
lcpu=2 mem=1024MB tmem=512MB psize=14 ent=0.40

%user  %sys  %wait  %idle physc %entc  lbusy   app  vcsw phint  %xcpu    dxm
----- ----- ------ ------ ----- ----- ------   --- ----- ----- ------ ------
 45.6  51.3    0.2    2.8  0.95 236.5   62.6 11.82  7024     2    5.8    165
 46.1  50.9    0.1    2.8  0.98 243.8   64.5 11.80  7088     7    6.0    162
 46.8  50.9    0.3    2.1  0.96 241.1   69.6 11.30  5413     6   19.4    163
 49.1  50.7    0.0    0.3  0.99 247.3   60.8 10.82   636     4    8.6    152
 49.3  50.5    0.0    0.3  1.00 248.9   56.7 11.47   659     1    0.3    153
```

ใน เอาต์พุต มีการแสดงสถิติการบีบอัดหน่วยความจำต่อไปนี้

- โหมดหน่วยความจำ **mmode** ของ LPAR คือ Active Memory Sharing ที่ปิดใช้งานและ AME ที่เปิดใช้งาน
- ขนาดหน่วยความจำที่ขยาย **mem** ของ LPAR คือ 1024 MB
- ขนาดหน่วยความจำจริง **tmem** ของ LPAR คือ 512 MB
- เปอร์เซ็นต์ของ CPU ที่ใช้สำหรับกิจกรรม Active Memory Expansion **%xcpu.**
- ขนาดของการขาดหน่วยความจำที่ขยาย **dxm** ในหน่วยเมกะไบต์

คำสั่ง topas

พาเนลหลัก **topas** ใน LPAR ที่มี Active Memory Expansion เปิดใช้งาน

แสดงสถิติการบีบอัด หน่วยความจำโดยอัตโนมัติภายใต้ส่วนย่อย **AME**

```
Topas Monitor for host:    proc7               EVENTS/QUEUES    FILE/TTY
Mon Dec 14 16:30:50 2009   Interval:  2        Cswitch    1240  Readch  43.2M
                                               Syscall  110.8K  Writech 102.5K
CPU  User%  Kern%  Wait%  Idle%  Physc   Entc  Reads     12594  Rawin      0
ALL   49.1   50.7    0.0    0.3   1.00  249.7  Writes      515  Ttyout   388
                                               Forks       218  Igets      0
Network  KBPS   I-Pack  O-Pack   KB-In  KB-Out  Execs       218  Namei    5898
Total     1.2      7.5     1.0     0.9     0.3  Runqueue    1.0  Dirblk      0
                                               Waitqueue   0.0
Disk    Busy%     KBPS     TPS KB-Read KB-Writ                   MEMORY
Total     0.0      0.0     0.0     0.0     0.0  PAGING           Real,MB   1024
                                               Faults    53184  % Comp     85
FileSystem        KBPS     TPS KB-Read KB-Writ  Steals        0  % Noncomp   0
Total             75.4K   21.1K  75.3K  95.4    PgspIn        0  % Client    0
                                               PgspOut       0
WLM-Class (Active)     CPU%    Mem%  Blk-I/O%   PageIn        0  PAGING SPACE
System                    0      61         0  PageOut       0  Size,MB    512
Default                   0       4         0  Sios          0  % Used      1
                                                                % Free     99
Name            PID CPU% PgSp Class            AME
inetd        364682  3.5  0.5 wpar1            TMEM,MB     512  WPAR Activ   1
xmtopas      622740  0.4  0.7 wpar1            CMEM,MB     114  WPAR Total   1
topas        413712  0.1  1.5 System           EF[T/A] 2.0/1.5  Press: "h"-help
random       204934  0.1  0.1 System           CI:5.5  CO:0.0          "q"-quit
```

ใน เอาต์พุตข้างบน มีการแสดงสถิติการบีบอัดหน่วยความจำ ต่อไปนี้

- ขนาดหน่วยความจำจริง **TMEM,MB** ของ LPAR คือ 512 MB
- ขนาดพูลที่บีบอัด **CMEM,MB** ของ LPAR คือ 114 MB
- **EF[T/A]** – ปัจจัยการขยายเป้าหมายคือ 2.0 และปัจจัยการขยายที่บรรลุ ผลคือ 1.5
- อัตราของการบีบอัด **co** และการยกเลิกบีบอัด **ci** ต่อ วินาทีคือ 0.0 และ 5.5 หน้าตามลำดับ

คำสั่ง svmon

เครื่องมือ **svmon** สามารถ แสดงมุมมองรายละเอียดการใช้ AME บน LPAR

```
# svmon -G -O summary=ame,pgsz=on,unit=MB
Unit: MB
--------------------------------------------------------------------------------
               size       inuse        free   pin     virtual  available mmode
memory      1024.00      607.54      144.11   306.29  559.75    136.61   Ded-E
 ucomprsd         -      387.55           -
 comprsd          -      219.98           -
pg space     512.00        5.08

               work        pers        clnt       other
pin          213.34           0           0        28.9
in use       534.12           0        9.42
 ucomprsd    314.13
 comprsd     219.98

PageSize   PoolSize       inuse        pgsp         pin     virtual   ucomprsd
s    4 KB         -      543.54        5.02      242.27      560.59     323.55
L   16 MB         4           0           0        64.0           0          0
--------------------------------------------------------------------------------
True Memory: 512.00

              CurSz  %Cur       TgtSz        %Tgt       MaxSz   %Max   CRatio
ucomprsd     405.93  79.28      168.38       32.89      -          -    -
comprsd      106.07  20.72      343.62       67.11      159.59   31.17  2.51

                txf         cxf         dxf         dxm
AME            2.00        1.46        0.54      274.21
```

ใน เอาต์พุตข้างบน มีการแสดงสถิติการบีบอัดหน่วยความจำ ต่อไปนี้:

- โหมดหน่วยความจำ **mmode** ของ LPAR คือ Active Memory Sharing ที่ปิดใช้งานและ AME ที่เปิดใช้งาน
- จากจำนวนที่ใช้ทั้งหมด 607.54 MB **memory_inuse** ประกอบด้วยหน้าที่ไม่บีบอัด **ucomprsd_inuse** จำนวน 387.55 MB และหน้าที่บีบอัด **comprsd_inuse** จำนวน ส่วนที่เหลือ 219.98 MB
- จากจำนวนหน้าที่ทำงานที่ใช้ทั้งหมด 534.12 MB **inuse_work** ประกอบด้วยหน้าที่ไม่บีบอัด **ucomprsd_work** จำนวน 314.13 MB และ หน้าที่บีบอัด **comprsd_work** จำนวน 219.98 MB
- จากจำนวนหน้าที่ใช้ทั้งหมด 543.54 MB **4KB_inuse** ในพูล 4K-PageSize ประกอบด้วยหน้าที่ไม่บีบอัด **4KB_ucomprsd** จำนวน 323.55 MB
- ขนาดหน่วยความจำที่ขยาย **memory_size** ของ LPAR คือ 1024 MB
- ขนาดหน่วยความจำจริง **True Memory** ของ LPAR คือ 512 MB

- ขนาดปัจจุบันของพูลที่ไม่บีบอัด **ucomprsd_CurSz** คือ 405.93 MB (79.28% ของขนาดหน่วยความจำจริงทั้งหมดของ LPAR)
- ขนาดปัจจุบันของพูลที่บีบอัด **comprsd_CurSz** คือ 106.07 MB (20.72% ของขนาดหน่วยความจำจริงทั้งหมดของ LPAR)
- ขนาดเป้าหมายของพูลที่บีบอัด **comprsd_TgtSz** ซึ่งต้องใช้ เพื่อบรรลุผลตามปัจจัยการขยายหน่วยความจำเป้าหมาย **txf** 2.00 คือ 343.62 MB (67.11% ของขนาดหน่วยความจำจริงทั้งหมดของ LPAR)
- ขนาดของพูลที่ไม่บีบอัด **ucomprsd_TgtSz** ในกรณีนั้น กลายเป็น 168.38 MB (32.89% ของขนาดหน่วยความจำจริงทั้งหมดของ LPAR)
- ขนาดสูงสุดของพูลที่บีบอัด **comprsd_MaxSz** คือ 159.59 MB (31.17% ของขนาดหน่วยความจำจริงทั้งหมดของ LPAR)
- อัตราส่วนการบีบอัดปัจจุบัน **CRatio** คือ 2.51 และปัจจัยการขยาย ปัจจุบัน **cxf** ที่บรรลุผลคือ 1.46
- จำนวนของการขาดหน่วยความจำที่ขยาย **dxm** คือ 274.21 MB และปัจจัยการขยายที่ขาดไป **dxf** คือ 0.54

อ็อพชัน **–O summary=longame** แสดงสรุปของรายละเอียด การบีบอัดหน่วยความจำดังนี้:

```
# svmon -G -O summary=longame,unit=MB
Unit: MB
                                  Active Memory Expansion
--------------------------------------------------------------
 Size     Inuse     Free   DXMSz  UCMInuse  CMInuse   TMSz    TMFr
1024.00  607.91  142.82  274.96   388.56    219.35   512.00  17.4

CPSz  CPFr  txf  cxf   CR
106.07 18.7 2.00 1.46   2.50
```

ในเอาต์พุต มีการแสดงสถิติการบีบอัดหน่วยความจำ ต่อไปนี้:

- จากขนาดหน่วยความจำที่ขยายทั้งหมด **Size** จำนวน 1024.00 MB 607.91 MB ถูกใช้อยู่ **Inuse** และ 142.82 MB ว่าง **Free** ขนาด หน่วยความจำที่ขยายซึ่งขาดไป **DXMSz** คือ 274.96 MB
- จากหน่วยความจำที่ใช้อยู่ทั้งหมด **Inuse** จำนวน 607.91 MB ประกอบด้วย หน้าที่ไม่บีบอัด **UCMInuse** จำนวน 388.56 MB และหน้าที่บีบอัด **CMInuse** จำนวนส่วนที่เหลือ 219.35 MB
- จากขนาดหน่วยความจำจริง **TMSz** จำนวน 512.00 MB เพียง 17.4 MB ของหน่วยความจำจริง **TMFr** เท่านั้นที่มีอยู่
- จากขนาดพูลที่บีบอัด **CPSz** จำนวน 106.07 MB เพียง 18.7 MB ของหน่วยความจำว่าง **CPFr** เท่านั้นมีอยู่ในพูล ที่บีบอัด
- ในขณะที่ปัจจัยการขยายเป้าหมาย **txf** คือ 2.00 ปัจจัยการขยาย ปัจจุบัน **cxf** ที่บรรลุผลคือ 1.46
- อัตราส่วนการบีบอัด (CR) คือ 2.50

การปรับแอ็พพลิเคชัน

ก่อนที่ใช้ความพยายามเป็นอย่างสูงในการปรับปรุงผลการทำงานของโปรแกรม ให้ใช้เทคนิคที่อยู่ในส่วนนี้เพื่อช่วยกำหนดผลการทำงานที่สามารถปรับปรุงได้ และค้นหาพื้นที่ของโปรแกรมที่การออปติไมซ์และการปรับ จะมีผลประโยชน์มากที่สุด

โดยทั่วไป การประมวลผลที่เหมาะสมที่สุดจะเกี่ยวข้องกับขั้นตอนหลายขั้นตอน:

- การปรับบางส่วนจะเกี่ยวข้องกับการเปลี่ยนซอร์สโค้ด ตัวอย่างเช่น การเรียงลำดับคำสั่งและนิพจน์ เทคนิคนี้รู้จักกันในนาม ของ *การปรับด้วยมือ*

- สำหรับโปรแกรม FORTRAN และ C การออปติไมซ์ตัวประมวลผลก่อนจะพร้อมใช้งาน เพื่อปรับและแปลงสภาพซอร์สโค้ดก่อนที่จะคอมไพล์ เอาต์พุตของตัวประมวลผลก่อนเหล่านี้ คือซอร์สโค้ด FORTRAN หรือ C ที่ถูกออปติไมซ์แล้ว
- คอมไพลเลอร์ FORTRAN หรือ C++ จะแปลซอร์สโค้ดให้เป็น ภาษาระดับกลาง
- ตัวสร้างโค้ดจะแปลโค้ดระดับกลางให้เป็นภาษาเครื่อง ตัวสร้างโค้ดสามารถออปติไมซ์โค้ดที่สามารถเรียกทำงานได้ในที่สุด เพื่อเพิ่มความเร็ว ซึ่งขึ้นอยู่กับอ็อพชันคอมไพลเลอร์ที่เลือกไว้ คุณสามารถเพิ่มจำนวนของการออปติไมซ์ ที่ดำเนินการในขั้นตอนนี้ได้ด้วยการปรับด้วยมือหรือการประมวลผลล่วงหน้าเป็นอันดับแรก

ความเร็วที่เพิ่มขึ้นเป็นผลกระทบจากปัจจัยสองอย่างคือ:
- จำนวนของการออปติไมซ์ที่ใช้กับส่วนของโปรแกรมแต่ละส่วน
- ความถี่ของการใช้สำหรับส่วนเหล่านั้นของโปรแกรม ณ รันไทม์

การเพิ่มความเร็วของรูทีนเดี่ยวอาจเพิ่มความเร็วของโปรแกรม ถ้ารูทีนดำเนินการกับการทำงานหลัก หรืออีกนัยหนึ่ง การดำเนินการจะไม่ปรับปรุงผลการทำงานทั้งหมด ถ้ารูทีนถูกเรียก และใช้เวลาไม่นาน โปรดระลึกไว้เสมอ เมื่อประเมินผลเทคนิคของผลการทำงานและข้อมูล เพื่อให้คุณสนใจเทคนิคที่มีค่าในงาน ของคุณ

สำหรับการกล่าวถึงเทคนิคเหล่านี้ โปรดดู *คู่มือการใช้ประโยชน์และการปรับ สำหรับ XL Fortran, XL C และ XL C++* และ โปรดดู “การออกแบบและการนำโปรแกรมไปปฏิบัติซึ่งมีประสิทธิภาพ” ในหน้า 96 สำหรับคำแนะนำเพิ่มเติม

เทคนิคการใช้ประโยชน์จากคอมไพลเลอร์

มีหลายเทคนิคสำหรับการออปติไมซ์คอมไพลเลอร์

พื้นที่หลักสามส่วนของการปรับซอร์สโค้ดมีดังต่อไปนี้:
- เทคนิคโปรแกรมมิงที่ใช้ประโยชน์ของการออปติไมซ์คอมไพลเลอร์ และสถาปัตยกรรมของระบบ
- BLAS ไลบรารีของ Basic Linear Algebra Subroutines ถ้าคุณมีโปรแกรมที่มีจำนวนมาก รูทีนย่อยเหล่านี้สามารถจัดเตรียมการปรับปรุงผลการทำงาน ที่สามารถนำมาพิจารณาได้ ส่วนขยายของ BLAS คือ ESSL นั่นคือ Engineering Scientific Subroutine Library นอกจากเซ็ตย่อยของไลบรารี BLAS แล้ว ESSL ยังสอดแทรกรูทีนเชิงคณิตศาสตร์สำหรับผลการทำงานสูงสำหรับคุณสมบัติทางเคมี ทางวิศวกรรม และทางฟิสิกส์ Parallel ESSL (PESSL) จะมีอยู่สำหรับเครื่อง SMP
- อ็อพชันคอมไพลเลอร์และการใช้ตัวประมวลผลก่อน เช่น KAP และ VAST จะมีอยู่จากผู้ขายในกลุ่มที่สาม

นอกจากเทคนิคการปรับซอร์สโค้ดเหล่านี้ โปรแกรม **fdpr** จะจัดโครงสร้างโค้ดอ็อบเจ็กต์ใหม่ โปรแกรม **fdpr** จะถูกกล่าวถึงใน “การสร้างโปรแกรมที่ดำเนินการได้ขึ้นใหม่ด้วยโปรแกรม fdpr” ในหน้า 130

การคอมไพล์ด้วยการทำให้เหมาะสมที่สุด

หากต้องการสร้างโปรแกรมที่มีผลการทำงานที่ดี ขั้นตอนแรกให้ใช้ประโยชน์จากคุณลักษณะกรทำให้เหมาะสมที่สุดที่ build คอมไพลเลอร์

การคอมไพล์ด้วยการทำให้เหมาะสมที่สุดสามารถเพิ่มความเร็วที่มาจากการปรับโปรแกรมของคุณ และสามารถลบความต้องการในการดำเนินการกับชนิดของการปรับบางอย่าง

ข้อแนะนำ

ให้ปฏิบัติตามแนวทางเหล่านี้สำหรับการทำให้เหมาะสมที่สุด:

- ใช้ **-O2** หรือ **-O3 -qstrict** สำหรับ FORTRAN ในระดับที่ใช้งานจริง โปรแกรม C หรือ C++ ที่คุณคอมไพล์ สำหรับโปรแกรม High Performance FORTRAN (HPF) ห้ามไม่ให้ใช้อ็อพชัน **-qstrict**
- ใช้อ็อพชัน **-qhot** สำหรับโปรแกรมที่มี hot spot เป็นลูป หรือภาษาอาร์เรย์ ให้ใช้อ็อพชัน **-qhot** เสมอสำหรับโปรแกรม HPF
- ใช้อ็อพชัน **-qipa** ที่อยู่ใกล้กับส่วนท้ายของวงรอบการพัฒนา หากเวลาการคอมไพล์ไม่ได้เป็นข้อพิจารณาหลัก

อ็อพชัน **-qipa** จะเรียกใช้หรือปรับแต่งคลาสของการทำให้เหมาะสม ซึ่งรู้จักกันในนามของ *การวิเคราะห์ระหว่างกระบวนการ* อ็อพชัน **-qipa** มีอ็อพชันย่อยจำนวนมาก ที่แสดงรายละเอียดอยู่ในคู่มือคอมไพลเลอร์ ซึ่งสามารถนำมาใช้ได้สองวิธี คือ:

- เมธอดแรกคือ การคอมไพล์ด้วยอ็อพชัน **-qipa** ในระหว่างขั้นตอนการคอมไพล์และขั้นตอนการลิงก์ ในระหว่างการคอมไพล์ คอมไพลเลอร์จะเก็บข้อมูลการวิเคราะห์ระหว่างกระบวนการ ในไฟล์ `.o` ในระหว่างการลิงก์ อ็อพชัน **-qipa** อาจจะเป็นสาเหตุทำให้เสร็จสิ้นการคอมไพล์อีกครั้งของแอ็พพลิเคชันทั้งหมด
- เมธอดที่สองคือ การคอมไพล์โปรแกรมสำหรับการทำโปรไฟล์ด้วยอ็อพชัน **-p/-pg** (พร้อมกับหรือไม่ระบุ **-qipa**) และรันโปรแกรมบนชุดของข้อมูลตัวอย่าง ข้อมูลผลลัพธ์ สามารถป้อนลงในการคอมไพล์ลำดับถัดมาด้วย **-qipa** เพื่อให้คอมไพลเลอร์ให้ความสนใจการทำให้เหมาะสมที่สุดในหน่วยวินาทีของโปรแกรม ซึ่งถูกใช้บ่อย

การใช้ **-O4** จะเทียบเท่ากับการใช้ **-O3 -qipa** พร้อมกับสร้างสถาปัตยกรรมแบบอัตโนมัติและอ็อพชันการปรับที่เกี่ยวข้องกับแพล็ตฟอร์ม การใช้แฟล็ก **-O5** จะคล้ายคลึงกับ **-O4** ยกเว้น **-qipa**= level = 2

คุณจะได้รับผลประโยชน์ต่อไปนี้ เมื่อคุณใช้คอมไพลเลอร์การทำให้เหมาะสมที่สุด:

การแบรนช์ที่ทำให้เหมาะสมที่สุด
การจัดเรียงโค้ดโปรแกรมเพื่อทำให้ลดตรรกะของการแบรนช์ และเพื่อรวมบล็อกของโค้ดแบบฟิสิคัลเข้าด้วยกัน

การเคลื่อนที่ของโค้ด
ถ้าตัวแปรที่ถูกใช้ในการคำนวณภายในลูปไม่ถูกเลือกไว้ภายในลูป การคำนวณสามารถดำเนินการภายนอกลูป และส่งผลให้ใช้ ภายในลูป

การกำจัดนิพจน์ย่อยทั่วไป
ในนิพจน์ทั่วไป ค่าเดียวกันนี้จะถูกนำมาคำนวณใหม่ในนิพจน์ ลำดับถัดมา การทำซ้ำนิพจน์สามารถกำจัดได้โดยการใช้ ค่าก่อนหน้านี้

การกระจายค่าคงที่
ค่าคงที่ที่ใช้ในนิพจน์จะถูกนำมารวมกัน และนิพจน์ใหม่จะถูกสร้างขึ้น การแปลงโดยนัยบางอย่างระหว่างชนิดตัวเลขและชนิดเลขทศนิยามจะถูกกระทำ

การกำจัดโค้ดที่หยุดนิ่ง
กำจัดโค้ดที่ไม่สามารถเข้าถึงหรือที่มีผลโดยไม่ถูกนำมาใช้ตามลำดับ

การกำจัดหน่วยเก็บที่หยุดนิ่ง
กำจัดหน่วยเก็บเมื่อค่าที่เก็บไม่ได้ถูกอ้างอิงอีกครั้ง ตัวอย่างเช่น ถ้าหน่วยเก็บทั้งสองหน่วยในตำแหน่งเดียวกันไม่มีการโหลดที่สอดแทรก หน่วยเก็บแรกจะไม่มีความจำเป็นและจะถูกลบทิ้ง

การจัดสรรเรจิสเตอร์แบบโกลบอล
จัดสรรตัวแปรและนิพจน์เพื่อให้พร้อมใช้งานกับเรจิสเตอร์ของฮาร์ดแวร์ โดยใช้อัลกอริธึม"การระบายสีกราฟ"

Inlining
แทนที่การเรียกฟังก์ชันด้วยโค้ดโปรแกรมที่แท้จริง

การกำหนดตารางเวลาคำสั่ง
เรียงลำดับคำสั่งใหม่เพื่อลดเวลาในการประมวลผล

การวิเคราะห์ระหว่างกระบวนการ
ไม่ครอบคลุมถึงความสัมพันธ์ระหว่างการเรียกฟังก์ชัน และกำจัดการโหลด หน่วยเก็บ และการคำนวณที่ไม่สามารถกำจัดได้ด้วยการทำให้เหมาะสมที่สุด

การลอยตัวของโค้ด IF ที่ไม่เปลี่ยนแปลง (ไม่มีการสับเปลี่ยน)
ลบโค้ดแบรนช์ที่ไม่เปลี่ยนแปลงออกจากลูปเพื่อทำให้มีโอกาสในการ ทำให้เหมาะสมที่สุด

ผลป้อนกลับการผลักดันให้ใช้โปรไฟล์
มีผลมาจากการประมวลผลตัวอย่างโปรแกรมที่ถูกใช้เพื่อปรับปรุงการทำให้เหมาะสมที่สุด ที่อยู่ใกล้กับแบรนช์ที่มีเงื่อนไข และในส่วนของโค้ดที่ประมวลผลบ่อย

การเชื่อมโยงความสัมพันธ์ใหม่
การจัดเรียงลำดับของการคำนวณในนิพจน์ตัวห้อยอาร์เรย์ การสร้างผู้สมัครสำหรับการกำจัดนิพจน์ทั่วไป

การเคลื่อนที่ของหน่วยเก็บ
ย้ายคำสั่งหน่วยเก็บออกจากลูป

การลดความซับซ้อน
แทนที่คำสั่งที่มีประสิทธิภาพน้อยด้วยคำสั่งที่มีประสิทธิภาพมากกว่า ตัวอย่างเช่น ในตัวห้อยอาร์เรย์ การเพิ่มคำสั่งจะแทนที่คำสั่งจำนวนมาก

การกำหนดตัวเลขให้กับค่า
เกี่ยวข้องกับการกระจายค่าคงที่ การกำจัดนิพจน์ และการล้มเลิกคำสั่งทั้งหลาย ในคำสั่งเดี่ยว

เมื่อต้องคอมไพล์โดยไม่มีการทำให้เหมาะสมที่สุด

ห้ามใช้อ็อพชัน **-O** สำหรับโปรแกรมที่คุณตั้งใจจะดีบักด้วยดีบักเกอร์สัญลักษณ์ โดยไม่พิจารณาว่า คุณใช้อ็อพชัน **-g** อย่างไรก็ตาม เนื่องจากการทำให้เหมาะสมที่สุดเป็นสิ่งที่สำคัญสำหรับโปรแกรม HPF ให้ใช้ **-O3 -qhot** สำหรับโปรแกรมเหล่านั้นในระหว่างการดีบัก

ตัวออปติไมเซอร์จะจัดเรียงคำสั่งในภาษาแอสเซมเบลอร์ และทำให้ยากต่อการแม็พคำสั่งแต่ละคำสั่งในบรรทัดของซอร์สโค้ด ถ้าคุณคอมไพล์ด้วยอ็อพชัน **-g** การจัดเรียงใหม่อาจกำหนดลักษณะที่ปรากฏ ซึ่งคำสั่งในระดับซอร์สจะถูกเรียกใช้งานในลำดับที่ผิด เมื่อคุณใช้ดีบักเกอร์สัญลักษณ์

ถ้าโปรแกรมของคุณสร้างผลลัพธ์ที่ไม่ถูกต้อง เมื่อโปรแกรมนั้นถูกคอมไพล์ด้วยอ็อพชัน **-O** ใดๆ ให้ตรวจสอบโปรแกรมสำหรับตัวแปรแฝงในการอ้างอิงถึงโพรซีเดอร์

การคอมไพล์สำหรับแพล็ตฟอร์มของฮาร์ดแวร์ที่ระบุเฉพาะ

มีหลายๆ สิ่งที่คุณควรพิจารณาก่อนที่จะรวบรวม แพล็ตฟอร์มฮาร์ดแวร์ที่ระบุเฉพาะ

ระบบสามารถใช้ชนิดต่างๆ ของตัวประมวลผลได้ โดยการใช้อ็อพชัน **-qarch** และ **-qtune** คุณสามารถออปติไมซ์โปรแกรมสำหรับคำสั่งพิเศษ และความแรงของตัวประมวลผลเหล่านี้

ข้อแนะนำ

ให้ปฏิบัติตามแนวทางเหล่านี้เพื่อรวบรวม แพล็ตฟอร์มฮาร์ดแวร์ที่ระบุเฉพาะ:

- ถ้าโปรแกรมของคุณรันเฉพาะบนระบบเดี่ยว หรือบนกลุ่มของระบบที่มีชนิดของตัวประมวลผลที่เหมือนกัน ให้ใช้อ็อพชัน **-qarch** เพื่อระบุชนิดของตัวประมวลผล
- ถ้าโปรแกรมของคุณจะรันอยู่บนระบบที่มีชนิดของตัวประมวลผลที่ต่างกัน และคุณสามารถระบุชนิดของตัวประมวลผลหนึ่งชนิดเป็นชนิดที่สำคัญที่สุด ให้ใช้ค่าติดตั้ง **-qarch** และ **-qtune** ที่เหมาะสม ผู้ใช้ XL FORTRAN และ XL HPF สามารถใช้คำสั่ง **xxlf** และ **xxlhpf** เพื่อเลือกค่าติดตั้งเหล่านี้แบบโต้ตอบ
- ถ้าโปรแกรมของคุณมีแนวโน้มที่จะรันอยู่บนช่วงเต็มของการนำตัวประมวลผลไปใช้งาน และไม่มีแนวโน้มสำหรับการใช้ชนิดหลักของตัวประมวลผลหนึ่งชนิด ห้ามใช้ **-qarch** หรือ **-qtune**

การคอมไพล์ผลการทำงานแบบอิงดัชนี

คุณสามารถเปลี่ยนอ็อพชันการอิงดัชนีที่เป็นค่าดีฟอลต์ เพื่อปรับปรุงผลการทำงานของโปรแกรมที่ใช้การอิงดัชนี

บางอ็อพชันสามารถกระทบกับโครงสร้างแบบอิงดัชนีมาตรฐาน การใช้อ็อพชันเหล่านี้สามารถเปลี่ยนผลลัพธ์ของการคำนวณได้ แต่ในหลายๆ กรณี ผลลัพธ์คือการเพิ่มขึ้นในความแม่นยำ

ข้อแนะนำ

ให้ปฏิบัติตามแนวทางเหล่านี้:

- สำหรับความแม่นยำของโปรแกรมบนแพล็ตฟอร์ม POWER family และ POWER2 คุณสามารถปรับปรุงผลการทำงานได้ขณะที่สงวนความถูกต้องไว้โดยใช้อ็อพชันแบบอิงดัชนีเหล่านี้:

  ```
  -qfloat=fltint:rsqrt:hssngl
  ```

  ถ้าความแม่นยำของโปรแกรมเดี่ยว ไม่ได้ใช้หน่วยความจำ (ตัวอย่างเช่น ถ้าไม่มีการเข้าถึงข้อมูลที่มากกว่า พื้นที่แคช) คุณสามารถขอรับผลการทำงานที่เท่ากันหรือมากกว่า และความแม่นยำที่มากกว่าได้โดยใช้:

  ```
  -qfloat=fltint:rsqrt -qautodbl=dblpad4
  ```

  สำหรับโปรแกรม ที่ไม่มีตัวแปรที่แม่นยำ ให้ใช้ **-qfloat=rsqrt:fltint** เท่านั้น หมายเหตุ **-O3** ที่ไม่ระบุ **-qstrict** จะตั้งค่า **-qfloat=rsqrt:fltint** โดยอัตโนมัติ
- โปรแกรมที่มีความแม่นยำจะมีประสิทธิภาพมากกว่าโปรแกรมที่มีความแม่นยำเป็นสองเท่า ดังนั้น การพัฒนาค่า REAL ที่เป็นค่าดีฟอลต์ไปเป็น REAL(8) สามารถลดผลการทำงานลงได้ ใช้อ็อพชันย่อย **-qfloat** ต่อไปนี้:

การระบุขนาดแคช

ถ้าโปรแกรมของคุณจัดทำขึ้นเพื่อรันเฉพาะบนเครื่องหรือการตั้งค่าคอนฟิก เดียว คุณสามารถช่วยคอมไพเลอร์ปรับโปรแกรมของคุณเป็นโครงร่างหน่วยความจำ ของเครื่องนั้นได้โดยใช้อ็อพชัน FORTRAN **-qcache**

คุณยังต้องระบุอ็อพชัน **-qhot** สำหรับ **-qcache** เพื่อ ให้มีผลบังคับใช้ อ็อพชัน **-qhot** ใช้ข้อมูล **-qcache** เพื่อ กำหนดการใช้ประโยชน์การจัดการหน่วยความจำที่เหมาะสม

แคชมีอยู่ สามชนิดคือ: ข้อมูล คำสั่ง และรวม โดยทั่วไป โมเดลแบ่งออก เป็นสองหมวดหมู่คือ: กลุ่มที่มีทั้งแคชข้อมูลและแคชคำสั่ง และกลุ่ม ที่มีแคชรวมระหว่างข้อมูล/คำสั่งแคชเดียว อ็อพชันย่อย TYPE อนุญาตให้คุณระบุ ชนิดของแคชที่อ็อพชัน **-qcache** ใช้อ้างอิง

อ็อพชัน **-qcache** ยังสามารถ ใช้เพื่อระบุขนาดและความเชื่อมโยงระหว่างชุดของแคชระดับ 2 ของโมเดล และ Translation Lookaside Buffer (TLB) ซึ่งเป็นตารางที่ใช้ระบุตำแหน่ง หน้าของหน่วยความจำที่อ้างอิงล่าสุด ในกรณีส่วนใหญ่ คุณไม่จำเป็นต้อง ระบุรายการ **-qcache** สำหรับ TLB ยกเว้นว่าระบบของคุณใช้พื้นที่ว่างข้อมูล มากกว่า 512 KB

อาจมีกรณีที่ค่าติดตั้งที่ต่ำกว่าสำหรับแอ็ตทริบิวต์ SIZE ให้ประสิทธิภาพที่พัฒนาขึ้น ทั้งนี้ขึ้นอยู่กับโหลดระบบ ในขณะที่รัน

การขยายการเรียกโพรซีเดอร์ inline

Inlining จะเกี่ยวข้องกับการทำสำเนาที่อ้างอิงโพรซีเดอร์ในโค้ด ที่โพรซีเดอร์เหล่านั้นถูกอ้างถึง สิ่งนี้จะกำจัดการเรียกใช้สำหรับรูทีน inlined และเปิดใช้งานตัวออปติไมซ์เพื่อดำเนินการ optimization อื่นๆ ในรูทีน inlined

สำหรับ FORTRAN และโปรแกรม C คุณสามารถระบุอ็อพชัน **-Q** (พร้อมกับ **-O2** หรือ **-O3**) เพื่อให้โพรซีเดอร์ inlined ภายในจุดที่อ้างอิง

Inlining จะปรับปรุงผลการทำงานในโปรแกรมบางตัว ขณะที่ลดระดับผลการทำงานในโปรแกรมตัวอื่น โปรแกรมที่มี inlining อาจทำงานช้าลง เนื่องจากขนาดของโค้ดที่มีขนาดใหญ่ ส่งผลทำให้มีแคชจำนวนมากที่หายไปและเกิดข้อบกพร่องของเพจ หรือเนื่องจากมีเรจิสเตอร์ที่ไม่เพียงพอในการพักตัวแปรโลคัลทั้งหมด ในรูทีนที่รวมกันบางชุด

ถ้าคุณใช้อ็อพชัน**-Q** ให้ตรวจสอบผลการทำงานของเวอร์ชันของโปรแกรมของคุณที่คอมไพล์ด้วย **-O3** และ **-Q** กับที่คอมไพล์ด้วย **-O3** เท่านั้น ผลการทำงานของโปรแกรมที่คอมไพล์ด้วย **-Q** จะปรับปรุงอย่างรวดเร็ว เสื่อมสภาพอย่างรวดเร็ว หรือเปลี่ยนแปลงเพียงเล็กน้อย หรือไม่ทั้งหมด

คอมไพลเลอร์จะพิจารณาถึงโพรซีเดอร์ inline ตามขนาดของ โพรซีเดอร์ คุณอาจสามารถปรับปรุงผลการทำงานของแอ็พพลิเคชันของคุณ โดยใช้เงื่อนไขอื่นๆ สำหรับ inlining สำหรับโพรซีเดอร์ที่ไม่ได้อ้างอิงในการประมวลผล (ตัวอย่างเช่น การจัดการข้อผิดพลาดและโพรซีเดอร์การดีบัก) ให้ปิดใช้งาน inlining ตามที่เลือกไว้โดยใช้อ็อพชัน **-Q-names** สำหรับโพรซีเดอร์ที่อ้างอิงภายใน hot spot ให้ระบุอ็อพชัน **-Q+names** เพื่อมั่นใจว่า โพรซีเดอร์เหล่านั้นยังคง inline เสมอ

ใช้การลิงก์แบบไดนามิกและการลิงก์แบบสแตติกเมื่อใด

ระบบปฏิบัติการมีฟังก์ชันสำหรับการสร้างและการใช้ ไลบรารีแบบแบ่งใช้ที่ลิงก์แบบไดนามิก ด้วยการลิงก์แบบไดนามิก สัญลักษณ์ภายนอก ที่อ้างอิงในรหัสผู้ใช้และกำหนดในไลบรารีแบบแบ่งใช้จะมีการแก้ไขโดยผู้โหลด ณ เวลาโหลด เมื่อคุณคอมไพล์โปรแกรมที่ใช้ไลบรารีแบบแบ่งใช้ ไลบรารีนั้นจะมีการลิงก์แบบไดนามิกกับโปรแกรมของคุณโดยค่าดีฟอลต์

แนวคิดเบื้องหลังไลบรารีแบบแบ่งใช้คือ การมีสำเนาของรูทีนที่ใช้ทั่วไปเพียงชุด เดียวเท่านั้น และเพื่อเก็บรักษาสำเนาทั่วไปนี้ในเซกเมนต์ไลบรารีแบบแบ่งใช้เฉพาะ รูทีนทั่วไปเหล่านี้สามารถลดขนาดโปรแกรมที่ดำเนินการได้ลงเป็นอย่างมาก ดังนั้นจึงประหยัดพื้นที่ว่างดิสก์

คุณสามารถลดขนาดโปรแกรมของคุณโดยใช้การลิงก์แบบไดนามิก แต่โดยปกติ ต้องแลกกับประสิทธิภาพที่ลดลง รหัสไลบรารีแบบแบ่งใช้ไม่มีอยู่ใน รูปภาพที่ดำเนินการได้บนดิสก์ แต่มีการเก็บไว้ในไฟล์ไลบรารี แยกต่างหาก รหัสที่แบ่งใช้ถูกโหลดเข้าในหน่วยความจำหนึ่งครั้งในเซกเมนต์ไลบรารีแบบ แบ่งใช้ และแบ่งใช้โดยกระบวนการทั้งหมดที่อ้างอิง ดังนั้น ไลบรารีที่ลิงก์แบบไดนามิกจึงลดจำนวน ของหน่วยเก็บเสมือนที่ใช้โดยโปรแกรมของคุณ ถ้าแอ็พพลิเคชันที่รันพร้อมกัน หลายรายการ (หรือสำเนาของแอ็พพลิเคชันเดียวกัน) ใช้โพรซีเดอร์ที่จัดเตรียมไว้ในไลบรารีแบบแบ่งใช้ นอกจากนี้ ยังลดจำนวนของพื้นที่ว่างดิสก์ ที่ต้องการสำหรับโปรแกรมของคุณ ถ้าแอ็พพลิเคชันอื่นหลายรายการซึ่งจัดเก็บ บนระบบที่กำหนดแบ่งใช้ไลบรารี ข้อดีอื่นของไลบรารีแบบแบ่งใช้ มีดังนี้:

- เวลาโหลดอาจลดลงเนื่องจากรหัสไลบรารีแบบแบ่งใช้อาจมีอยู่แล้วใน หน่วยความจำ

- ประสิทธิภาพรันไทม์อาจพัฒนาขึ้นเนื่องจากระบบปฏิบัติการมีโอกาสน้อยกว่าที่จะ page out รหัสไลบรารีแบบแบ่งใช้ที่กำลังใช้อยู่โดยหลายแอ็พพลิเคชัน หรือสำเนาของแอ็พพลิเคชัน เมื่อเปรียบเทียบกับรหัสที่มีการใช้โดย แอ็พพลิเคชันเดียว ผลคือเกิด page faults น้อยลง
- รูทีนไม่ได้ยึดแบบสแตติกกับแอ็พพลิเคชัน แต่ยึดแบบไดนามิก เมื่อโหลดแอ็พพลิเคชัน ลักษณะนี้ทำให้แอ็พพลิเคชันได้รับการเปลี่ยนแปลงในไลบรารีแบบ แบ่งใช้โดยอัตโนมัติ โดยไม่ต้องรีคอมไพล์หรือยึดใหม่

ข้อเสียของการลิงก์แบบไดนามิกมีดังต่อไปนี้:

- จากมุมมองด้านประสิทธิภาพ มี "glue code" ที่ต้องการในโปรแกรมที่ ดำเนินการได้เพื่อเข้าถึงเซกเมนต์แบบแบ่งใช้ มีต้นทุนในการอ้างอิงรูทีนไลบรารี แบบแบ่งใช้ประมาณแปดรอบเครื่องต่อ การอ้างอิง โดยปกติแล้ว โปรแกรมที่ใช้ไลบรารีแบบแบ่งใช้ช้ากว่าโปรแกรม ที่ใช้ไลบรารีที่ลิงก์แบบสแตติก
- ผลกระทบย่อยคือการลดลงใน "ความเป็นโลคัลของการอ้างอิง" คุณอาจ สนใจในเพียงบางรูทีนในไลบรารีเท่านั้น และรูทีนเหล่านี้กระจัดกระจายใน บริเวณกว้างในพื้นที่ว่างที่อยู่เสมือนของไลบรารี ดังนั้น จำนวนหน้าทั้งหมดที่คุณต้องผ่านเพื่อเข้าถึงรูทีนทั้งหมดของคุณจึงสูงกว่า ถ้ารูทีนย่อยเหล่านี้ถูกยึดไว้ในโปรแกรมที่ดำเนินการได้ของคุณโดยตรง อย่างมาก ผลกระทบหนึ่งของสถานการณ์นี้คือ ถ้าคุณเป็นผู้ใช้เพียงรายเดียว ของรูทีนเหล่านี้ คุณจะพบ page faults มากกว่ากรณีที่นำรูทีน ทั้งหมดเข้าในหน่วยความจำจริง นอกจากนี้ เนื่องจากผ่านหน้ามากขึ้น จึงมีโอกาสเกิด instruction translation lookaside buffer (TLB) miss มากขึ้น
- เมื่อโปรแกรมอ้างอิงโพรซีเดอร์จำนวนที่จำกัดในไลบรารี แต่ละหน้าของไลบรารีที่มีโพรซีเดอร์ที่อ้างอิงต้องถูก paged into หน่วยความจำทีละหน้า ถ้าโพรซีเดอร์เล็กพอ การลิงก์แบบสแตติกอาจรวมโพรซีเดอร์ หลายรายการที่ลิงก์ซึ่งอยู่ในหน้าไลบรารีที่แตกต่างกันเข้าในหน้าเดียว การลิงก์แบบไดนามิกอาจเพิ่มการเพจ ส่งผลให้ประสิทธิภาพ ลดลง
- โปรแกรมที่ลิงก์แบบไดนามิกต้องอาศัยไลบรารีที่เข้ากันได้ ถ้าไลบรารีเปลี่ยนแปลง (ตัวอย่างเช่น คอมไพเลอร์รีลีสใหม่อาจเปลี่ยน ไลบรารี) แอ็พพลิเคชันอาจต้องทำงานใหม่เพื่อให้เข้ากันได้กับ เวอร์ชันใหม่ของไลบรารี ถ้าไลบรารีถูกลบออกจากระบบ โปรแกรมที่ใช้ไลบรารีนั้นจะ ไม่ทำงานอีกต่อไป

ในโปรแกรมที่ลิงก์แบบสแตติก รหัสทั้งหมดมีอยู่ในโมดูลที่ดำเนินการได้เพียง โมดูลเดียว การอ้างอิงไลบรารีมีประสิทธิภาพมากกว่าเนื่องจากไลบรารีโพรซีเดอร์ มีการลิงก์แบบสแตติกเข้าในโปรแกรม การลิงก์แบบสแตติกเพิ่มขนาดไฟล์ของโปรแกรมของคุณ และอาจเพิ่มขนาดรหัสในหน่วยความจำถ้าแอ็พพลิเคชันอื่น หรือสำเนาอื่นของแอ็พพลิเคชันของคุณ กำลังรันบนระบบ

คำสั่ง **cc** มีค่าดีฟอลต์เป็นอ็อพชันไลบรารีแบบแบ่งใช้ เมื่อต้องการแทนที่ค่าดีฟอลต์ เมื่อคุณคอมไพล์โปรแกรมของคุณเพื่อสร้างไฟล์ อ็อบเจ็กต์ที่ลิงก์แบบสแตติก ให้ใช้อ็อพชัน **-bnso** ดังนี้:

```
cc xxx.c -o xxx.noshr -O -bnso -bI:/lib/syscalls.exp
```

อ็อพชันนี้บังคับให้ตัวลิงก์วางไลบรารีโพรซีเดอร์ที่โปรแกรมของคุณอ้างอิง เข้าในไฟล์อ็อบเจ็กต์ของโปรแกรม ไฟล์ `/lib/syscalls.exp` มีชื่อ ของรูทีนระบบที่ต้องนำเข้าไปในโปรแกรมของคุณจาก ระบบ ต้องระบุไฟล์นี้สำหรับการลิงก์แบบสแตติก รูทีนที่ไฟล์ระบุชื่อ มีการนำเข้าโดยอัตโนมัติโดย `libc.a` สำหรับการลิงก์แบบไดนามิก คุณจึงไม่ต้องระบุไฟล์นี้ในระหว่างการลิงก์แบบไดนามิก หากต้องการรายละเอียดเพิ่มเติมเกี่ยวกับอ็อพชันเหล่านี้ให้ดู “การใช้ของคำสั่ง ld อย่างมีประสิทธิภาพ” ในหน้า 453 และคำสั่ง **ld**

การพิจารณาถึงไลบรารีที่ไม่ได้แบ่งใช้ซึ่งช่วยในเรื่องของผลการทำงาน:

หนึ่งเมธอดของการพิจารณาว่า แอ็พพลิเคชันของคุณคำนึงถึงวิธีการที่ไลบรารีที่แบ่งใช้คอมไพล์โปรแกรมเรียกทำงานอีกครั้งโดยใช้อ็อพชันไม่แบ่งใช้

ถ้าผลการทำงานมีแนวโน้มที่ดีขึ้น คุณอาจต้องการพิจารณาการแลกเปลี่ยนผลประโยชน์อื่นๆ ของไลบรารีที่แบ่งใช้ เพื่อให้ได้รับผลการทำงานที่ดีขึ้น อย่างไรก็ตาม ให้ตรวจสอบเพื่อวัดผลการทำงานในสภาวะแวดล้อมที่น่าเชื่อถือ โปรแกรมที่โยงกับการไม่แบ่งใช้อาจรันได้เร็วกว่า เนื่องจากเป็นอินสแตนซ์เดียวในเครื่องที่โหลดอยู่ โปรแกรมเดียวกันเมื่อใช้โดยจำนวนของผู้ใช้อย่างพร้อมเพียงกัน อาจเพิ่มการใช้หน่วยความจำที่เกิดขึ้นจริงให้เพียงพอ เพื่อลดเวิร์กโหลดทั้งหมด

ไลบรารีแบบแบ่งใช้ที่โหลดไว้แล้ว

ตัวแปรสภาพแวดล้อม *LDR_PRELOAD* และ *LDR_PRELOAD64* ทำให้กระบวนการสามารถโหลดไลบรารีแบบแบ่งใช้ไว้ล่วงหน้าได้ ตัวแปรสภาพแวดล้อม *LDR_PRELOAD* ใช้สำหรับกระบวนการ 32 บิต และตัวแปรสภาพแวดล้อม *LDR_PRELOAD64* ใช้สำหรับกระบวนการ 64 บิต

ในระหว่างการแก้ไขสัญลักษณ์ไลบรารีที่โหลดไว้แล้วซึ่งแสดงรายการอยู่ใน ตัวแปรสภาพแวดล้อมจะถูกค้นหาก่อนสำหรับทุกสัญลักษณ์ที่นำเข้า และเฉพาะถ้า ไม่พบสัญลักษณ์ ระบบจะใช้การค้นหาปกติ การใช้สัญลักษณ์ จากไลบรารีที่โหลดไว้แล้วใช้สำหรับทั้งการลิงก์ดีฟอลต์และการลิงก์แบบรันไทม์ของ AIX การแก้ไขสัญลักษณ์ที่เลื่อนออกไปไม่มีการเปลี่ยนแปลง

หากต้องการข้อมูลเพิ่มเติมเกี่ยวกับตัวแปรสภาพแวดล้อมเหล่านี้ ให้ดู “พารามิเตอร์ที่สามารถปรับแต่งได้เบ็ดเตล็ด” ในหน้า 473

การระบุลำดับลิงก์เพื่อลดการเพจสำหรับโปรแกรมขนาดใหญ่

ในระหว่างระยะการลิงก์ของการคอมไพล์โปรแกรม ตัวลิงก์จะเปลี่ยนตำแหน่ง หน่วยโปรแกรมเพื่อพยายามปรับปรุงความเป็นโลคัลของการอ้างอิง

ตัวอย่างเช่น ถ้าโพรซีเดอร์อ้างอิงโพรซีเดอร์อื่น ผู้ลิงก์อาจทำให้โพรซีเดอร์ อยู่ติดกันในโหลดโมดูล เพื่อให้ทั้งสองโพรซีเดอร์อยู่ในหน้าของหน่วยความจำ เสมือนเดียวกัน ซึ่งช่วยลดโอเวอร์เฮดการเพจ เมื่อโพรซีเดอร์แรกมีการอ้างอิงเป็นครั้งแรกและหน้าที่มีโพรซีเดอร์นั้นถูกนำเข้า ไปในหน่วยความจำจริง โพรซีเดอร์ที่สองจะพร้อมใช้งานได้โดยไม่มีโอเวอร์เฮด การเพจเพิ่มเติม

ในโปรแกรมขนาดใหญ่มากที่การเพจเกิดขึ้นทันที สำหรับหน้าของรหัสของโปรแกรมของคุณ คุณอาจเลือกใช้ลำดับลิงก์ บนตัวลิงก์ได้ คุณสามารถทำเช่นนี้ได้โดยการจัดเรียงส่วนการควบคุมในลำดับ ที่คุณต้องการลิงก์ และใช้อ็อพชัน **–bnoobjreorder** เพื่อป้องกัน ไม่ให้จัดลำดับตัวลิงก์ใหม่ ส่วนการควบคุมหรือ CSECT คือหน่วยที่เปลี่ยนได้ที่เล็กที่สุด ของรหัสหรือข้อมูลใน XCOFF อ็อบเจ็กต์โมดูล หากต้องการรายละเอียดเพิ่มเติม ให้ดู *Files Reference*

อย่างไรก็ตาม มีความเสี่ยงหลายอย่างในการระบุลำดับลิงก์ หลังการจัดลำดับลิงก์ใหม่ ควรมีการทดสอบประสิทธิภาพเพื่อให้มั่นใจว่าลำดับลิงก์ของคุณให้ผลลัพธ์ ที่ดีขึ้นสำหรับโปรแกรม เมื่อเปรียบเทียบกับลำดับลิงก์ที่ ตัวลิงก์เลือก พิจารณาจุดต่อไปนี้ก่อนตัดสินใจ สร้างลำดับลิงก์ของคุณเอง:

- คุณต้องกำหนดลำดับลิงก์สำหรับ CSECTs ทั้งหมดในโปรแกรมของคุณ CSECTs ต้องมีการนำเสนอไปยังตัวลิงก์ในลำดับซึ่งคุณต้องการ ลิงก์ ในโปรแกรมขนาดใหญ่ การจัดลำดับดังกล่าวต้องใช้ความพยายามสูงและเสี่ยง ต่อการเกิดข้อผิดพลาด
- ประโยชน์ด้านประสิทธิภาพที่เห็นในระหว่างการพัฒนาโปรแกรมอาจกลายเป็น การสูญเสียประสิทธิภาพในเวลาต่อมาได้ เนื่องจากการเปลี่ยนขนาดรหัสอาจทำให้ CSECTs ที่เคยถูกจัดวางไว้ด้วยกันก่อนหน้านี้ในหน้าหนึ่งจะถูกแยกไปอยู่คนละหน้า
- การจัดลำดับใหม่สามารถเปลี่ยนความถี่ของการปะทะระหว่างสายงานในแคชคำสั่ง ในการดำเนินการกับแคชคำสั่งหรือแคชรวมระหว่างข้อมูลและคำสั่ง ที่มีความเชื่อมโยงระหว่างชุดแบบสองทาง สายงานของรหัสโปรแกรมใดๆ สามารถ จัดเก็บไว้ในสายงานของแคชเพียงหนึ่งหรือสองสายงานเท่านั้น ถ้าโพรซีเดอร์อิสระขนาดสั้นตั้งแต่สามรายการขึ้นไป มีคลาส

cache-congruence เดียวกัน การชนกันของแคชคำสั่งอาจทำให้ ประสิทธิภาพลดลงได้ การจัดลำดับใหม่อาจทำให้เกิดการปะทะระหว่างสายงานแคชซึ่งไม่เคย เกิดขึ้นมาก่อน และยังสามารถตัดการปะทะระหว่างสายงานแคชที่เกิดขึ้นเมื่อไม่ได้ระบุ **-bnoobjreorder** ด้วย

ถ้าคุณพยายามปรับลำดับลิงก์ของโปรแกรมของคุณ ควรทดสอบ ประสิทธิภาพเสมอบนระบบที่หน่วยเก็บจริงทั้งหมดและการใช้ประโยชน์หน่วย ความจำโดยระบบอื่นคล้ายกับสภาพแวดล้อมการทำงานที่คาดไว้ ลำดับลิงก์ ที่ทำงานได้บนระบบที่ไม่มีเสียงดังซึ่งมีภารกิจที่รันเพียงเล็กน้อยอาจก่อให้เกิด ความวุ่นวายของหน้าบนระบบที่ยุ่งมากขึ้น

การเรียกไลบรารี BLAS และ ESSL

Basic Linear Algebra Subroutines (BLAS) จัดเตรียมระดับของผลการทำงานที่สูง สำหรับสมการพีชคณิตเชิงเส้นในการดำเนินการแบบ matrix-matrix, matrix-vector และ vector-vector Engineering and Scientific Subroutine Library (ESSL) จะมีชุดของรูทีนย่อยที่ครอบคลุมมากกว่า ซึ่งเป็นรูทีนย่อยทั้งหมดที่ถูกปรับสำหรับตระกูล POWER processor-based สถาปัตยกรรม POWER2 และ PowerPC

รูทีนย่อย BLAS และ ESSL สามารถช่วยคุณประหยัดเวลาในความพยายาม ในการปรับการคำนวณทางคณิตศาสตร์ และยังคงจัดเตรียมผลการทำงาน ที่ดีกว่าที่ได้รับโดยการปรับด้วยมือ หรือการทำให้เหมาะสมที่สุดของ การคำนวณทางคณิตศาสตร์ที่เขียนโค้ดขึ้น คุณสามารถเรียกฟังก์ชันได้จากทั้งไลบรารีจาก FORTRAN, C และโปรแกรม C++

ไลบรารี BLAS คือคอลเล็กชันของ Basic Linear Algebra Subroutines ที่ได้ถูกปรับไว้สำหรับสถาปัตยกรรม เซ็ตย่อย BLAS จะถูกจัดเตรียมไว้พร้อมกับระบบปฏิบัติการ (`/lib/libblas.a`)

ผู้ใช้ควรใช้ไลบรารีนี้ สำหรับการดำเนินการกับแมทริกซ์และเวกเตอร์ของตนเอง เนื่องจากการดำเนินการเหล่านี้จะถูกปรับระดับที่ผู้ใช้บรรลุเป้าหมายของตนเอง

รูทีน BLAS ถูกออกแบบมาเพื่อเรียกจากโปรแกรม FORTRAN แต่สามารถใช้ได้กับโปรแกรม C การดูแลอาจถูกนำมาใช้ เนื่องจากความแตกต่างของภาษา เมื่ออ้างถึงแมทริกซ์ ตัวอย่างเช่น FORTRAN จะเก็บอาร์เรย์ในคอลัมน์การเรียงลำดับหลัก ขณะที่ C ใช้แถวในการเก็บการเรียงลำดับหลัก

หากต้องการสอดแทรกไลบรารี BLAS ซึ่งมีอยู่ใน `/lib/libblas.a` ให้ใช้อ็อพชัน **-lblas** บนคำสั่งของคอมไพลเลอร์ (**xlf -O prog.f -lblas**) ถ้าการเรียก BLAS จากโปรแกรม C ยังสอดแทรกอ็อพชัน **-lxlf** ไว้สำหรับไลบรารี FORTRAN (**cc -O prog.c -lblas -lxlf**)

ESSL คือไลบรารีระดับสูงเพิ่มเติมที่สอดแทรกความหลากหลายของฟังก์ชันเชิงคณิตศาสตร์ ที่ถูกใช้ในพื้นที่เกี่ยวกับวิศวกรรม เคมี และฟิสิกส์

ประโยชน์ในการใช้รูทีนย่อย BLAS หรือ ESSL มีดังต่อไปนี้:

- การเรียกรูทีนย่อย BLAS และ ESSL จะง่ายต่อโค้ดเมื่อเปรียบเทียบกับการดำเนินการที่ แทนที่
- รูทีนย่อย BLAS และ ESSL สามารถเคลื่อนย้ายได้ระหว่างแพล็ตฟอร์มที่แตกต่างกัน ชื่อของรูทีนย่อย และลำดับการเรียกจะเป็นมาตรฐาน
- โค้ด BLAS จะดำเนินการได้ดีบนทุกแพล็ตฟอร์ม การโค้ดภายในของรูทีน จะระบุเฉพาะแพล็ตฟอร์ม ดังนั้น โค้ดจะผูกกับคุณสมบัติของผลการทำงานของ สถาปัตยกรรม

สำหรับโปรแกรมตัวอย่าง เก้าบรรทัดต่อไปนี้คือโค้ด FORTRAN:

```
do l=1,control
do j=1,control
        xmult=0.d0
        do k=1,control
                xmult=xmult+a(i,k)*a(k,j)
        end do
        b(i,j)=xmult
end do
end do
```

ซึ่งจะแทนที่ด้วยบรรทัดต่อไปนี้ของ of FORTRAN ที่เรียกรูทีน BLAS:

```
call dgemm (`n',`n',control,control,control,1,d0,a, control,a,1control,1.d0,b,control)
```

การรปรับปรุง ผลการทำงานต่อไปนี้จะถูกเฝ้าดู:

มิติอาร์เรย์	เวลาที่ใช้ไปของ MULT	เวลาที่ใช้ไปของ BLAS	อัตราส่วน
101 x 101	.1200	.0500	2.40
201 x 201	.8900	.3700	2.41
301 x 301	16.4400	1.2300	13.37
401 x 401	65.3500	2.8700	22.77
501 x 501	170.4700	5.4100	31.51

ตัวอย่างนี้สาธิตวิธีที่โปรแกรมใช้การดำเนินการคูณแมทริกซ์ซึ่งดีกว่าการใช้รูทีน BLAS ในระดับ 3 สำหรับผลการทำงานที่ปรับปรุงแล้ว หมายเหตุ การปรับปรุงจะเพิ่มขนาดของอาร์เรย์

ข้อคิดเห็นโดยตรงต่อโปรไฟล์ (Profile Directed Feedback)

PDF คือคอมไพเลอร์อ็อพชันเพื่อทำการใช้ประโยชน์ระดับโพรซีเดอร์เพิ่ม เติม เช่น การจัดสรรทะเบียนคำสั่ง การจัดตารางเวลาคำสั่ง และการจัดเรียง บล็อกพื้นฐานใหม่

เมื่อต้องการใช้ PDF ให้ทำดังต่อไปนี้:

1. คอมไพล์ซอร์สไฟล์ในโปรแกรมด้วย **-qpdf1** (ฟังก์ชัน **main( )** ต้อง ถูกคอมไพล์ด้วย) ต้องใช้อ็อพชัน **-lpdf** ในระหว่างขั้นตอนลิงก์ และยังต้องใช้อ็อพชันการคอมไพล์อื่นทั้งหมดที่ใช้ในระหว่างขั้นตอนที่ 3 ด้วย
2. รันโปรแกรมตามชุดข้อมูลปกติ โปรแกรม บันทึกข้อมูลโปรไฟล์เมื่อจบการทำงานและเข้าไปในไฟล์ที่ชื่อว่า __BLOCKS ในไดเร็กทอรีซึ่งระบุโดยตัวแปรสภาพแวดล้อม *PDFDIR* หรือ ในไดเร็กทอรีที่กำลังทำงานในปัจจุบันถ้าไม่มีการตั้งค่าตัวแปรนั้น คุณสามารถรันโปรแกรมหลายครั้งด้วยชุดข้อมูลที่แตกต่างอื่น และข้อมูลโปรไฟล์ จะมีการสะสมไว้เพื่อแสดงจำนวนที่ถูกต้องของ branches ที่ใช้และบล็อกของโค้ดที่ดำเนินการ สิ่งสำคัญคือการใช้ข้อมูลที่เป็นตัวแทนของ ข้อมูลที่ใช้ในระหว่างการรันโปรแกรมที่เสร็จสิ้นแล้วตามปกติ
3. รีคอมไพล์โปรแกรมโดยใช้คอมไพเลอร์อ็อพชันเดียวกับที่ใช้ในขั้นตอนที่ 1 แต่เปลี่ยน **-qpdf1** เป็น **-qpdf2** โปรดจำไว้ว่า **-L** และ **-l** เป็น อ็อพชันตัวลิงก์ และคุณสามารถเปลี่ยนอ็อพชันเหล่านี้ได้ ณ จุดนี้ โดยเฉพาะ การละเว้นอ็อพชัน **-lpdf** ในการคอมไพล์ครั้งที่สองนี้ ข้อมูลโปรไฟล์ที่สะสมไว้จะมีการนำมาใช้ในการปรับการใช้ประโยชน์สูงสุดโดยละเอียด โปรแกรมที่ได้ไม่มี โอเวอร์เฮดการจัดทำโปรไฟล์และรันด้วยความเร็วสูงสุด

มีคำสั่งสองรายการสำหรับการจัดการไดเร็กทอรี *PDFDIR* ดังนี้:

resetpdf *pathname*

ล้างข้อมูลการจัดทำโปรไฟล์ทั้งหมด (แต่ไม่ได้ลบไฟล์ข้อมูล) ออกจากไดเร็กทอรี pathname ถ้าไม่ได้ระบุ pathname จะลบข้อมูลออกจากไดเร็กทอรี *PDFDIR* หรือถ้าไม่ได้ตั้งค่า *PDFDIR* จะลบออกจากไดเร็กทอรีปัจจุบัน เมื่อคุณทำการเปลี่ยนแปลงในแอ็พพลิเคชันและรีคอมไพล์บางไฟล์ ข้อมูลการจัดทำ โปรไฟล์สำหรับไฟล์เหล่านี้จะถูกรีเซ็ตโดยอัตโนมัติ รันคำสั่ง **resetpdf** เพื่อรีเซ็ต ข้อมูลการจัดทำโปรไฟล์สำหรับทั้งแอ็พพลิเคชัน หลังจากทำการเปลี่ยนแปลงที่สำคัญ ซึ่งอาจส่งผลกระทบต่อจำนวนการดำเนินการสำหรับส่วนประกอบของโปรแกรม ที่ไม่ได้รีคอมไพล์

cleanpdf *pathname*

ลบข้อมูลการจัดทำโปรไฟล์ทั้งหมดออกจากไดเร็กทอรี pathname หรือ *PDFDIR* หรือ ไดเร็กทอรีปัจจุบัน การลบข้อมูลโปรไฟล์ช่วยลดรันไทม์โอเวอร์เฮด ถ้าคุณเปลี่ยนโปรแกรม แล้วกลับไปยังกระบวนการ PDF อีกครั้งหนึ่ง รันโปรแกรมนี้ หลังจากการคอมไพล์ด้วย **-qpdf2**

คำสั่ง fdpr

คำสั่ง **fdpr** สามารถจัดเรียงโค้ดใหม่ภายในโปรแกรมเรียกทำงาน ที่คอมไพล์แล้วเพื่อปรับปรุงผลการทำงานของแบรนช์ ย้ายโค้ดที่ใช้แล้วออกจากโปรแกรมที่ hot spot และทำการออปติไมซ์โกลบอลอื่นๆ

คำสั่งนี้จะทำงานได้ดีที่สุดสำหรับโปรแกรมที่มีขนาดใหญ่ด้วยเงื่อนไขการทดสอบจำนวนมาก หรือโปรแกรมที่มีโครงสร้างพร้อมกับโพรซีเดอร์ที่วางกระจัดกระจายจำนวนมาก คำสั่ง **fdpr** ได้กล่าวถึงใน “การสร้างโปรแกรมที่ดำเนินการได้ขึ้นใหม่ด้วยโปรแกรม fdpr” ในหน้า 130

การใช้ตัวประมวลผลล่วงหน้าให้เกิดประโยชน์สูงสุดสำหรับ FORTRAN และ C

การทดสอบประสิทธิภาพบ่งชี้การพัฒนาขึ้นในช่วง 8 ถึง 18 เปอร์เซ็นต์โดยเฉลี่ย เมื่อชุดของโปรแกรมคอมไพล์เข้ากับตัวประมวลผลล่วงหน้า เมื่อเปรียบเทียบกับการคอมไพล์เข้ากับอ็อพชันการใช้ประโยชน์เดียวกันของเวอร์ชัน ที่ไม่มีการประมวลผลล่วงหน้า

ตัวประมวลผลล่วงหน้า KAP และ VAST สำหรับ FORTRAN คอมไพเลอร์สามารถสร้าง FORTRAN ซอร์สโค้ดขึ้นใหม่เพื่อให้ใช้ POWER family, POWER2, และรีซอร์สหน่วยการประมวลผลและลำดับชั้นหน่วยความจำ PowerPC ได้ดียิ่งขึ้น และยังมีเวอร์ชันของตัวประมวลผลล่วงหน้า KAP สำหรับการสร้างโค้ดของโปรแกรม C ขึ้นใหม่ด้วย ตัวประมวลผลล่วงหน้า ทำการใช้การจัดการหน่วยความจำ การแปลงพีชคณิต, inlining, การวิเคราะห์ระหว่างโพรซีเดอร์ และการใช้ประโยชน์อื่นๆ เพื่อพัฒนาประสิทธิภาพ ของ FORTRAN หรือ C แอ็พพลิเคชัน

ตัวประมวลผลล่วงหน้า KAP และ VAST พยายามแปลงขั้นตอนวิธีระดับซอร์ส เป็นขั้นตอนวิธีที่สามารถได้รับประโยชน์จากความสามารถในการใช้ให้เกิดประโยชน์สูงสุด ของคอมไพเลอร์ได้อย่างเต็มที่ ตัวประมวลผลล่วงหน้ายังสร้างรายการที่ระบุการแปลงที่ดำเนินการแล้ว และพื้นที่ของโค้ดที่ป้องกันไม่ให้ทำ การแปลง ตัวประมวลผลล่วงหน้าวิเคราะห์ซอร์สโค้ด และทำการแปลงที่สามารถพัฒนาประสิทธิภาพของโปรแกรม

การแปลงใดๆ ที่ทำโดยตัวประมวลผลล่วงหน้ายังสามารถทำได้โดยใช้ การปรับด้วยมือ ข้อดีของการใช้ตัวประมวลผลล่วงหน้าแทนการปรับด้วยมือ มีดังนี้:

- ในหลายกรณี ตัวประมวลล่วงหน้าให้โปรแกรมที่สามารถทำงานได้มีประสิทธิภาพ เทียบเท่าหรือดีกว่าการปรับด้วยมือ โดยไม่ต้องทำให้โปรแกรมเมอร์ต้องเสีย เวลานาน หากคุณใช้ตัวประมวลผลล่วงหน้า คุณอาจไม่ต้องทำความเข้าใจ กับสถาปัตยกรรมหรือเทคนิคการปรับที่อธิบายไว้ในที่อื่นใน คู่มือนี้ครบถ้วนทั้งหมด

- สำหรับบางโปรแกรม คุณสามารถได้รับโค้ดที่มีประโยชน์มาก โดยเพียงแต่ เลือกอ็อพชันตัวประมวลผลล่วงหน้าบนบรรทัดคำสั่งที่เหมาะสม และเพิ่มทิศทาง เล็กน้อยไปยังซอร์สโค้ดของโปรแกรมของคุณ ในกรณีที่ ตัวประมวลผลล่วงหน้าไม่ได้ให้การพัฒนาที่สังเกตได้ ให้ทำงานกับรายการ ตัวประมวลผลล่วงหน้าเพื่อดูว่าพื้นที่ใดของซอร์สโค้ดที่ทำให้ไม่เกิดการใช้ประโยชน์สูงสุด
- การแปลงบางอย่างที่ทำโดยตัวประมวลผลล่วงหน้าเกี่ยวข้องกับส่วนขยายจำนวนมาก ของซอร์สโค้ด ในขณะที่ส่วนขยายเหล่านี้สามารถพัฒนาประสิทธิผลของโปรแกรม แต่การดำเนินการด้วยวิธีการปรับด้วยมือจะเพิ่มโอกาสของข้อผิดพลาด ขั้นตอนวิธี และการพิมพ์ ลดการอ่านได้ของซอร์สโค้ด และทำให้การดูแลรักษาโปรแกรมยากขึ้น
- ตัวประมวลผลล่วงหน้าสามารถสร้างโค้ดที่มีการปรับสำหรับการตั้งค่าคอนฟิก สถาปัตยกรรมเฉพาะ แม้แต่โค้ดที่ไม่มีอยู่บนระบบ POWER family, POWER2, และ PowerPC คุณสามารถเก็บรักษาซอร์สโค้ดเวอร์ชันเดียว และจัดทำเวอร์ชันที่แปลง ซึ่งจะมีการปรับสำหรับโมเดล POWER family, POWER2, และ PowerPC แบบอื่น หรือสำหรับเครื่องที่มีลักษณะแคชและตัวประมวลผลอื่น
- บ่อยครั้งที่ตัวประมวลผลล่วงหน้าสามารถพัฒนาบนโค้ดแบบปรับด้วยมือ แม้ว่าสามารถปรับโปรแกรม ของคุณด้วยมือได้ดีในระดับประสิทธิผลเดียวกันกับตัวประมวลผลล่วงหน้า แต่การแปลงที่ซับซ้อนมากขึ้นบางอย่างอาจทำให้เกิดข้อผิดพลาดการโค้ดได้ เมื่อพยายามทำด้วยมือ

เทคนิคเกี่ยวกับ Code-optimization

การลดระดับจากการใช้หน่วยความที่ไม่มีประสิทธิภาพจะมีค่ามากกว่า การใช้แคชที่ไม่มีประสิทธิภาพ เนื่องจากความแตกต่างในเรื่องของความเร็วระหว่าง หน่วยความจำและดิสก์จะมีค่าที่สูงกว่าความเร็วระหว่างแคชและหน่วยความจำ

เทคนิค Code-optimization ประกอบด้วยสิ่งต่อไปนี้:

- ในการลดปริมาณของโค้ดชุดการทำงานของโปรแกรม ให้แพ็กโค้ดที่เรียกใช้บ่อยๆ เข้าด้วยกัน ขณะที่แยกโค้ดที่ใช้ไม่บ่อยออก หรืออีกนัยหนึ่ง ห้ามวางบล็อกที่มีขนาดยาวของโค้ดการจัดการข้อผิดพลาดไว้ในบรรทัดและโหลดโมดูลที่เรียกบ่อยที่อยู่ถัดจากตัวเรียก
- ในการลดจำนวนข้อมูลชุดการทำงาน ให้สนใจถึงการรวมข้อมูลที่ใช้บ่อย และหลีกเลี่ยงการอ้างอิงเพจที่ไม่จำเป็น การดำเนินการนี้สามารถบรรลุเป้าหมายได้โดยใช้รูทีนย่อย **malloc( )** แทนการใช้รูทีนย่อย **calloc( )** ให้กำหนดค่าเริ่มต้นให้กับโครงสร้างข้อมูลในทันทีก่อนที่โครงสร้างข้อมูลนั้นจะถูกนำมาใช้ และมั่นใจว่าคุณได้ล้างข้อมูลและไม่ยอมรับพื้นที่ที่ถูกจัดสรร เมื่อไม่มีความต้องการอีกต่อไป
- ในการลดจำนวนหน่วยเก็บที่ตรึงไว้ ให้จัดทำแพ็กเกจที่ตรึงโค้ดไว้ในโมดูลการโหลดที่แยกจากกัน ตรวจสอบให้แน่ใจว่ามีความจำเป็นที่ต้องใช้โค้ดที่ตรึงไว้ โครงสร้างของระบบบางระบบ (เช่น พูล mbuf) จะถูกตรึงไว้ในหน่วยความจำ ห้ามเพิ่มโครงสร้างเหล่านี้โดยไม่มีกฎเกณฑ์
- เทคนิคในแบบเรียลไทม์สามารถนำมาใช้ได้ เช่น รูทีนย่อย **plock( )** เพื่อตรึงโค้ดในหน่วยความจำ และระดับความสำคัญที่ตรึงไว้ด้วยรูทีนย่อย **setpri( )**

ไฟล์แม็พ

การใช้ไฟล์แม็พคือเทคนิคการ optimization โค้ด

แอ็พพลิเคชันสามารถใช้การเรียกระบบ **shmat( )** หรือ **mmap( )** เพื่อเข้าถึงไฟล์ตามแอดเดรส แทนการใช้การเรียกระบบการอ่านและเขียน เนื่องจากมีการใช้ที่เชื่อมโยงกับการเรียกของระบบ การเรียกเพียงเล็กน้อยจะดีกว่า การเรียก **shmat( )** หรือ **mmap( )** สามารถปรับผลการทำงานได้มากสุด 50 ครั้งเมื่อเปรียบเทียบกับการเรียกระบบ **read( )** หรือ **write( )** แบบดั้งเดิม หากต้องการใช้รูทีนย่อย **shmat( )** ไฟล์จะถูกเปิด และ file descriptor (*fd*) จะถูกส่งคืน หากการเรียกระบบการอ่านหรือเขียนที่

ถูกใช้ การเรียก **shmat( )** จะส่งแอดเดรสของไฟล์แม็พ ตั้งค่าองค์ประกอบ ที่เท่ากับแอดเดรสต่อมาในไฟล์ แทนการใช้การเรียกระบบการอ่านจำนวนมาก ทำการอ่านจากไฟล์ไปยังแมทริกซ์

การเรียก **mmap( )** แสดงการแม็พหน่วยความจำระหว่างขอบเขตเซ็กเมนต์ ผู้ใช้สามารถมีได้มากกว่า 10 พื้นที่แม็พลงในหน่วยความจำ ฟังก์ชัน **mmap( )** จะจัดเตรียมการปกป้องในระดับของเพจสำหรับพื้นที่หน่วยความจำ เพจแต่ละเพจ สามารถมีการเขียนหรือการอ่าน หรือไม่สามารถเข้าถึงชุดของสิทธิที่ไม่สามารถเข้าถึง การเรียก **mmap( )** อนุญาตให้การแม็พของหนึ่งเพจของไฟล์

การเรียก **shmat( )** ยังอนุญาตให้การแม็พของเซ็กเมนต์ที่มากกว่าหนึ่ง เมื่อไฟล์ที่ถูกแม็พมีขนาดใหญ่กว่าเซ็กเมนต์

ต่อไปนี้คือตัวอย่างโปรแกรมที่อ่านจากไฟล์การใช้คำสั่งอ่าน:

```
fd = open("myfile", O_RDONLY);
for (i=0;i<cols;i++) {
        for (j=0;j<rows;j++) {
                read(fd,&n,sizeof(char));
                *p++ = n;
        }
}
```

การใช้รูทีนย่อย **shmat( )** จะมีผลลัพธ์ที่เหมือนกันซึ่งบรรลุเป้าหมายได้โดย ไม่มีคำสั่งอ่าน:

```
fd = open("myfile", O_RDONLY);
nptr = (signed char *) shmat(fd,0,SHM_MAP | SHM_RDONLY);
for (i=0;i<cols;i++) {
        for (j=0;j<rows;j++) {
                *p++ = *nptr++;
        }
}
```

ข้อเสียเปรียบในการใช้ไฟล์แม็พคือการเขียน คุณลักษณะของระบบแบบ write-behind ซึ่งเป็นการเขียนเพจที่แก้ไขไปยังไฟล์โดยใช้บล็อกลำดับ ไม่ได้ใช้เมื่อแอ็พพลิเคชันใช้รูทีนย่อย **shmat( )** หรือ **mmap( )** เพจที่แก้ไข สามารถเก็บรวบรวมอยู่ในหน่วยความจำ และจะถูกเขียโดยการสุ่ม เมื่อ Virtual Memory Manager (VMM) ต้องการพื้นที่ สถานะการณ์นี้จะส่งผลต่อการเขียนขนาดเล็กลงในดิสก์ จะทำให้การใช้ CPU และดิสก์ไม่มีประสิทธิภาพ

การมอนิเตอร์ผลการทำงานของ Java

มีเมธอดจำนวนมากที่พร้อมใช้งานสำหรับการแยกคอขวด และผลการทำงานสำหรับการปรับในแอ็พพลิเคชัน Java™

Java คือภาษาการเขียนโปรแกรมที่ มุ่งเน้นอ็อบเจ็กต์ ซึ่งพัฒนาโดย Oracle Java เป็นโมเดลที่มาหลัง C++ และออกแบบมาให้มีแพล็ตฟอร์มขนาดเล็ก แบบง่าย และสามารถเคลื่อนย้ายได้ และระบบปฏิบัติการที่ระดับซอร์สและที่ระดับไบนารี โปรแกรม Java ที่ประกอบด้วยแอ็พเพล็ตและแอ็พพลิเคชัน ซึ่งสามารถรันบนเครื่องใดๆ ที่ติดตั้ง Java Virtual Machine, JVM

ข้อได้เปรียบของ Java

Java มีข้อได้เปรียบที่สำคัญเหนือภาษาอื่นๆ และสภาวะแวดล้อมที่เหมาะสมสำหรับภารกิจโปรแกรมมิงใดๆ

ข้อได้เปรียบของ Java มีดังต่อไปนี้:

- Java ง่ายต่อการเรียนรู้

Java ถูกออกแบบมาเพื่อให้ง่ายต่อการใช้งาน และง่ายต่อการเขียน คอมไพล์ ดีบัก และศึกษาได้ง่ายกว่าภาษาโปรแกรมอื่นๆ

- Java คือภาษาในเชิงอ็อบเจ็กต์
  ซึ่งอนุญาตให้คุณสร้างมอดูลาร์โปรแกรม และโค้ดที่สามารถนำกลับมาใช้ได้ใหม่
- Java คือแพล็ตฟอร์มอิสระ
  หนึ่งในข้อได้เปรียบที่สำคัญของ Java คือ ความสามารถในการย้ายจากระบบคอมพิวเตอร์หนึ่งไปยังระบบอื่น ความสามารถ ในการรันโปรแกรมเดียวกันได้บนระบบที่ต่างกันเป็นข้อได้เปรียบที่สำคัญสำหรับซอฟต์แวร์แบบเวิล์ดไวด์เว็บ และ Java ประสบความสำเร็จกับจุดนี้โดยทำแพล็ตฟอร์มอิสระ ทั้งในระดับซอร์สและระดับไบนารี

เนื่องจากสภาพกันผิดพลาดของจาวา ความง่ายต่อการใช้ ความสามารถในการใช้ระหว่างแพล็ตฟอร์ม และคุณลักษณะการรักษาความปลอดภัย จาวาจึงกลายเป็นภาษาของตัวเลือกสำหรับการจัดเตรียมโซลูชันอินเตอร์เน็ต ได้อย่างกว้างขวาง

คำแนะนำเกี่ยวกับผลการทำงาน Java

ผลการทำงานของ Java บน AIX สามารถได้รับการปรับปรุงในหลายวิธี

- ใช้ฟังก์ชัน **StringBuffer** แทนการต่อข้อมูลสตริง เมื่อจัดการกับสตริงที่เกินปกติเพื่อหลีกเลี่ยงการสร้างอ็อบเจ็กต์ ที่ไม่จำเป็นซึ่งต้องใช้การเก็บรวบรวมขยะ
- หลีกเลี่ยงการเขียนลงคอนโซล Java เพื่อลดต้นทุนของการจัดการกับสตริง การจัดรูปแบบข้อความ และเอาต์พุต
- หลีกเลี่ยงต้นทุนของการสร้างอ็อบเจ็กต์ และการจัดการโดยใช้ชนิดพื้นฐาน สำหรับตัวแปรเมื่อจำเป็น
- อ็อบเจ็กต์ที่มีการใช้แคชบ่อยควรลดจำนวนการเก็บรวบรวมขยะเท่าที่ต้องการ และหลีกเลี่ยงความต้องการในการสร้างอ็อบเจ็กต์อีกครั้ง
- การดำเนินการกับกลุ่มท้องถิ่นเพื่อลดจำนวนของการเรียก Java Native Interface (JNI) หากเป็นไปได้
- ใช้เมธอดที่ซิงโครไนซ์เท่านั้นเมื่อต้องการจำกัดมัลติทาสกิงใน JVM และระบบปฏิบัติการ
- หลีกเลี่ยงการใช้ตัวเก็บรวบรวมขยะเว้นเสียแต่จำเป็นต้องใช้ ถ้าคุณต้องเรียกใช้ตัวเก็บรวมรวมขยะ ให้ทำในช่วงเวลาที่ไม่ได้ใช้งานหรือเฟสที่ไม่สำคัญ
- ใช้ชนิด **int** แทนชนิด **long** เมื่อเป็นไปได้ เนื่องจากการดำเนินการแบบ 32 บิตจะถูกเรียกใช้งานได้เร็วกว่าการดำเนินการแบบ 64 บิต
- วิธีการประกาศอาจเป็นวิธีท้ายสุดเมื่อเป็นไปได้ เมธอดสุดท้ายจะถูกจัดการได้โดย JVM
- ใช้คีย์เวิร์ด **static final** เมื่อสร้างค่าคงที่ เพื่อลดจำนวนครั้งที่ที่ตัวแปรต้องการกำหนดค่าเริ่มต้น
- หลีกเลี่ยงการอ้างอิง "casts" และ "instanceof" เนื่องจาก casting ใน Java จะทำที่รันไทม์
- หลีกเลี่ยงการใช้เวกเตอร์เท่าที่เป็นไปได้เมื่ออาร์เรย์มีเพียงพอ
- เพิ่มหรือลบรายการจากส่วนท้ายของเวกเตอร์
- หลีกเลี่ยงการจัดสรรอ็อบเจ็กต์ภายในลูป
- ใช้บัฟเฟอร์ I/O และปรับขนาดบัฟเฟอร์
- ใช้การเชื่อมต่อพูลและคำสั่งที่จัดเตรียมการแคชไว้สำหรับการเข้าถึงฐานข้อมูล
- ใช้การเชื่อมต่อพูลกับฐานข้อมูลนำการเชื่อมต่อกลับมาใช้แทน การปิดและเปิดการเชื่อมต่อ
- ขยายและย่อให้เล็กสุดสำหรับการสร้างเธรดและการทำลายวงรอบ
- ลด contention ของรีซอร์สที่แบ่งใช้

- ลดการสร้างอ็อบเจ็กต์ที่มีอายุสั้น
- หลีกเลี่ยงการเรียกแบบรีโมต
- ใช้การเรียกกลับเพื่อหลีกเลี่ยงการบล็อกการเรียกแบบรีโมต
- หลีกเลี่ยงการสร้างอ็อบเจ็กต์ที่ใช้สำหรับเข้าถึงเมธอด
- ทำให้เมธอดซิงโครไนซ์ซึ่งอยู่นอกลูป
- เก็บสตริงและข้อมูลอักขระเป็น Unicode ในฐานข้อมูล
- การเรียงลำดับ CLASSPATH ใหม่เพื่อใช้ไลบรารีที่ใช้บ่อย

เครื่องมือในการมอนิเตอร์ Java

มีเครื่องมือบางชุดที่คุณสามารถใช้เพื่อมอนิเตอร์และระบุตัวยับยั้งผลการทำงานในแอ็พพลิเคชัน Java ของคุณ

vmstat แสดงข้อมูลเกี่ยวกับรีซอร์สของระบบที่หลากหลาย ซึ่งจะรายงานข้อมูลสถิติบนเธรดของเคอร์เนล ในคิวของการรัน พร้อมกับในคิวของการรอ การใช้หน่วยความจำ พื้นที่การเพจ ดิสก์ I/O อินเตอร์รัปต์ การเรียกของระบบ context switch และกิจกรรมของ CPU

iostat รายงานข้อมูลดิสก์ I/O โดยละเอียด

topas รายงาน CPU เน็ตเวิร์ก ดิสก์ I/O Workload Manager และกิจกรรมการประมวลผล

tprof ทำโปรไฟล์แอ็พพลิเคชันเพื่อหาตำแหน่งรูทีนหรือเมธอดที่เป็นที่นิยม ซึ่งสามารถนำมาพิจารณาปัญหาเกี่ยวกับผลการทำงาน

ps -mo *THREAD*
แสดงการประมวลผล CPU หรือเธรดที่โยงอยู่

Java profilers [-Xrunhprof, Xrunjpa64]
พิจารณารูทีนหรือเมธอดที่ใช้อย่างหนัก

java -verbose:gc
ตรวจสอบผลกระทบของการเก็บรวบรวมขยะบนแอ็พพลิเคชันของคุณ ซึ่งจะรายงานเวลาที่ใช้ไปทั้งหมด ในการเก็บรวบรวมขยะ เวลาเฉลี่ยต่อการเก็บรวบรวมขยะ หน่วยความจำโดยเฉลี่ยที่สะสมต่อการเก็บรวบรวมขยะ และอ็อบเจ็กต์โดยเฉลี่ยที่สะสม ต่อการเก็บรวบรวมขยะ

การปรับ Java สำหรับ AIX

AIX มีชุดของพารามิเตอร์ที่แนะนำให้ใช้สำหรับสภาวะแวดล้อม Java ของคุณ

AIXTHREAD_SCOPE=S
ค่าดีฟอลต์สำหรับตัวแปรนี้ คือ **P** ซึ่งหมายถึง process-wide contention scope (M:N) ค่า **S** หมายถึง system-wide contention scope (1:1) สำหรับแอ็พพลิเคชัน Java ค่าดีฟอลต์ของตัวแปรนี้ คือ **S**

AIXTHREAD_MUTEX_DEBUG=OFF
รักษารายการของ mutex ที่แอ็คทีฟสำหรับการใช้โดยดีบักเกอร์

AIXTHREAD_COND_DEBUG=OFF
รักษารายการของตัวแปรเงื่อนไขสำหรับการใช้โดยดีบักเกอร์

AIXTHREAD_RWLOCK_DEBUG=OFF

รักษารายการของการล็อกที่ไม่เกิดร่วมกันแบบแอ็คทีฟ ตัวแปรเงื่อนไข และการล็อกการอ่าน-เขียน สำหรับใช้โดยดีบักเกอร์ เมื่อการล็อกถูกกำหนดค่าเริ่มต้น การล็อกจะถูกเพิ่มลงในรายการ ถ้ายังไม่มีอยู่ รายการนี้จะนำไปใช้เป็นรายการที่ลิงก์ ดังนั้นการค้นหาเพื่อนำมาพิจารณาว่า การล็อกยังคงอยู่หรือไม่ มีผลการทำงานเมื่อรายการมีขนาดใหญ่ ปัญหาจะเพิ่มขึ้นตามความเป็นจริงที่ว่า รายการจะถูกป้องกันไว้โดยการล็อก ซึ่งจะพักไว้สำหรับช่วงเวลาของการดำเนินการ ค้นหา หรือการเรียกรูทีนย่อย **pthread_mutex_init( )** ต้องรอขณะที่การค้นหาเสร็จสิ้น สำหรับผลการทำงานสูงสุด คุณควรตั้งค่าของอ็อพชันเธรด-ดีบักนี้ให้เป็น OFF ค่าดีฟอลต์คือ ON

SPINLOOPTIME=500

จำนวนครั้งที่การประมวลผลสามารถหมุนการล็อกที่ไม่ว่างก่อนที่จะบล็อก ค่านี้จะมีค่า 40 ตามค่าดีฟอลต์ ถ้าเอาต์พุตคำสั่ง **tprof** บ่งชี้การใช้ CPU สูงสำหรับรูทีน **check_lock** และถ้าการล็อก พร้อมใช้งานภายในเวลาอันสั้น คุณควรเพิ่มเวลาในการหมุนโดยตั้งค่าเป็น 500 หรือสูงกว่า

และ ค่าติดตั้งต่อไปนี้จะแนะนำให้ใช้สำหรับสภาวะแวดล้อม Java ของคุณ:

ulimit -d *unlimited*

ulimit -m *unlimited*

ulimit -n *unlimited*

ulimit -s *unlimited*

พารามิเตอร์สภาวะแวดล้อมและค่าติดตั้งสามารถนำมาใช้เพื่อปรับผลการทำงาน Java ภายในระบบปฏิบัติการ นอกจากนี้จำนวนเทคนิคสำหรับการปรับคอมโพเนนต์ของระบบ เช่น CPU หน่วยความจำ เน็ตเวิร์ก I/O และอื่นๆ สามารถเพิ่มผลการทำงาน Java ได้ หากต้องการพิจารณาว่า พารามิเตอร์สภาวะแวดล้อมอาจมีประโยชน์กับสถานการณ์ของคุณ โปรดอ้างถึงหัวข้อเฉพาะสำหรบข้อมูลเพิ่มเติม

หากต้องการขอรับผลการทำงานและความสามารถในการวัด Java ให้ใช้เวอร์ชันล่าสุดที่มีอยู่ของระบบปฏิบัติการและJava พร้อมกับคอมไพลเลอร์ Just-In-Time (JIT)

การเก็บรวบรวมขยะที่ส่งผลต่อผลการทำงานของ Java

ปัญหาเกี่ยวกับผลการทำงานโดยทั่วไปจะเชื่อมโยงกับ Java ที่เกี่ยวข้องกับกลไกการเก็บรวบรวมขยะ

ถ้าขนาดของฮีป Java มีขนาดใหญ่เกินไป ฮีปนั้นต้องอยู่ในหน่วยความจำหลัก ซึ่งเป็นสาเหตุทำให้กิจกรรมการเพจเพิ่มขึ้นและส่งผลต่อผลการทำงานของ Java

และ ฮีปขนาดใหญ่ยังสามารถใช้เวลาหลายวินาทีในการเติมให้เต็ม นั่นหมายความว่า ถ้าการเก็บรวบรวมขยะเกิดขึ้นไม่ย่อย เวลาในการหยุดพักที่เชื่อมโยงกับการเก็บรวบรวมขยะ จะเพิ่มขึ้น

หากต้องการปรับฮีปของ Java Virtual Machine (JVM) ให้ใช้คำสั่ง **java** พร้อมด้วยอ็อพชัน **-ms** or **-mx** ใช้ข้อมูลสถิติการเก็บรวบรวมขยะ เพื่อช่วยในการพิจารณาถึงค่าติดตั้งที่เหมาะสม

การวิเคราะห์ผลการทำงานด้วยตัวช่วยการติดตาม

ตัวช่วยการติดตามของระบบปฏิบัติการคือทูลการเฝ้าดูระบบ ที่มีประสิทธิภาพ

ตัวช่วยการติดตามจะดักจับลำดับการไหลของเหตุการณ์ของระบบที่ประทับเวลาไว้ จัดเตรียมระดับของรายละเอียดที่เหมาะสมกับกิจกรรมของระบบ เหตุการณ์จะถูกแสดงในลำดับเวลา และในบริบทของเหตุการณ์อื่นๆ การติดตามเป็นเครื่องมือที่มีค่าสำหรับการเฝ้าดูระบบ และการประมวลผลแอ็พพลิเคชัน ไม่เหมือนกับเครื่องมืออื่นๆ ที่จัดเตรียมเฉพาะการใช้ประโยชน์จาก CPU หรือช่วงเวลารอ I/O การติดตามจะขยายข้อมูลเพื่อวัตถุประสงค์ในการทำความเข้าใจถึงเหตุการณ์ที่กำลังเกิดขึ้น ผู้ที่มีหน้าที่รับผิดชอบ เวลาที่เหตุการณ์เข้าแทนที่ ผลกระทบที่มีต่อระบบ และสาเหตุที่เกิดผลกระทบนั้น

ระบบปฏิบัติการจะถูกติดตั้งไว้เพื่อจัดเตรียมความสามารถในการมองเห็น การประมวลผลของระบบโดยทั่วไป ผู้ใช้สามารถขยายความสามารถในการมองเห็นในแอ็พพลิเคชันของพวกเขา โดยการแทรกเหตุการณ์เพิ่มเติมและจัดเตรียมกฎการจัดรูปแบบ

การดูแลจะนำมาใช้ในการออกแบบและการนำไปปฏิบัติของตัวช่วยนี้ เพื่อสร้างการเก็บรวบรวมข้อมูลที่มีประสิทธิภาพ ดังนั้นผลการทำงานของระบบ และการไหลจะถูกปรับเปลี่ยนเพียงเล็กน้อยตามการเรียกใช้การติดตาม ด้วยเหตุนี้ ตัวช่วยการติดตามจะมีประโยชน์ในฐานะเป็นเครื่องมือช่วยวิเคราะห์ผลการทำงาน และเป็นเครื่องมือช่วยในการกำหนดปัญหา

ฟังก์ชันการติดตามโดยละเอียด

ฟังก์ชันการติดตามมีความยืดหยุ่นมากกว่าการบริการมอนิเตอร์ระบบ ดั้งเดิม ที่ประเมินและแสดงสถิติซึ่งจัดทำขึ้นโดยระบบ

ไม่สามารถคาดล่วงหน้าว่าจะต้องการสถิติใด การติดตามนำเสนอลำดับของ เหตุการณ์และอนุญาตให้ผู้ใช้เลือกข้อมูลที่ต้องการดึงไปใช้ได้ ด้วยการบริการมอนิเตอร์ดั้งเดิม การลดข้อมูล (การแปลงจากเหตุการณ์ระบบเป็น สถิติ) อาศัยเครื่องมือของระบบเป็นหลัก ตัวอย่างเช่น หลายระบบเก็บรักษาค่าต่ำสุด ค่าสูงสุด และเวลาที่ผ่านไปเฉลี่ยซึ่งสังเกตได้สำหรับ การดำเนินการของภารกิจ A และอนุญาตให้ดึงข้อมูลนี้ได้

ฟังก์ชันการติดตามไม่ได้เชื่อมโยงการลดข้อมูลกับเครื่องมือ แต่นำเสนอลำดับของเร็กคอร์ดเหตุการณ์การติดตาม (โดยปกติย่อเป็น *เหตุการณ์*) ไม่จำเป็นต้องตัดสินใจล่วงหน้าว่าจะต้องการสถิติใด การลดข้อมูล ไม่เกี่ยวข้องกับเครื่องมือ ผู้ใช้ อาจเลือกกำหนดค่าต่ำสุด ค่าสูงสุด และเวลาเฉลี่ยสำหรับภารกิจ A จากลำดับของเหตุการณ์ และยังสามารถ:

- ดึงข้อมูลเวลาเฉลี่ยสำหรับภารกิจ A เมื่อเรียกโดยกระบวนการ B
- ดึงข้อมูลเวลาเฉลี่ยสำหรับภารกิจ A เมื่อตรงตามเงื่อนไข XYZ
- คำนวณการเบี่ยงเบนมาตรฐานของเวลารันสำหรับภารกิจ A
- ตัดสินใจว่าภารกิจอื่นซึ่งรับทราบจากลำดับของเหตุการณ์ มีประโยชน์ มากกว่าที่จะสรุปหรือไม่

ความยืดหยุ่นนี้มีประโยชน์สำหรับการวิเคราะห์ปัญหาประสิทธิภาพหรือ ฟังก์ชัน

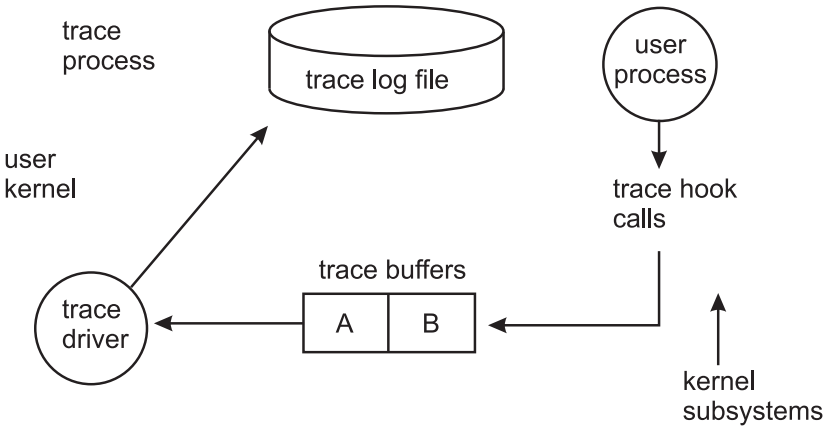
นอกเหนือจากการให้ข้อมูลรายละเอียดเกี่ยวกับกิจกรรมระบบแล้ว ฟังก์ชันการ ติดตามช่วยให้สามารถทดลองแอ็พพลิเคชัน โปรแกรมและรวบรวมเหตุการณ์การ ติดตามเพิ่มเติมจากเหตุการณ์ระบบ ไฟล์การติดตามมีเร็กคอร์ดที่สมบูรณ์ ของแอ็พพลิเคชันและกิจกรรมระบบในลำดับที่ถูกต้อง และมีเวลาประทับที่แม่นยำ

การนำฟังก์ชันการติดตามไปใช้

trace hook คือเหตุการณ์เฉพาะที่จะถูกมอนิเตอร์ หมายเลขเฉพาะที่กำหนดให้กับเหตุการณ์นั้นเรียกว่า *hook ID* คำสั่ง **trace** มอนิเตอร์ hooks เหล่านี้

คำสั่ง **trace** สร้างสถิติเกี่ยวกับกระบวนการผู้ใช้ และระบบย่อยเคอร์นัล ข้อมูลไบนารีมีการบันทึกลงใน บัฟเฟอร์อื่นสองราย การในหน่วยความจำ จากนั้นกระบวนการ **trace** โอนย้ายข้อมูลไปยังไฟล์บันทึกการติดตามบนดิสก์ ไฟล์ นี้ใหญ่ขึ้นอย่างรวดเร็วมาก โปรแกรม **trace** รันเป็น กระบวนการซึ่งอาจมอนิเตอร์ได้โดยคำสั่ง **ps** คำสั่ง **trace** ทำหน้าที่เป็น daemon คล้ายกับ การลงบัญชี

รูปภาพต่อไปนี้แสดงการนำไปใช้ของฟังก์ชัน การติดตาม



รูปที่ 25. การนำไปใช้ของฟังก์ชันการติดตาม. ภาพสาธิตนี้ แสดงกระบวนการติดตาม ในกระบวนการนี้ กระบวนการผู้ใช้ (ระบบย่อยเคอร์นัล) ส่งการเรียก trace hook ไปยังบัฟเฟอร์การติดตามที่มีชื่อว่า A และ B จากบัฟเฟอร์ การเรียกเดินทางผ่านไดรเวอร์การติดตาม และไปยัง ไฟล์บันทึกการติดตามของเคอร์นัลผู้ใช้

การมอนิเตอร์ฟังก์ชันใช้รีซอร์สระบบ สิ่งที่ดีที่สุดคือ โอเวอร์เฮด ควรจะต่ำเพียงพอที่จะไม่กระทบต่อการดำเนินการของระบบมากนัก เมื่อโปรแกรม **trace** ใช้งานอยู่ CPU โอเวอร์เฮด มีค่าน้อยกว่า 2 เปอร์เซ็นต์ เมื่อข้อมูลการติดตามเต็มบัฟเฟอร์และต้องถูก บันทึกลงในไฟล์บันทึก ต้องการ CPU เพิ่มเติมสำหรับไฟล์ I/O โดยปกติ ค่านี้น้อยกว่า 5 เปอร์เซ็นต์ เนื่องจาก โปรแกรม **trace** เรียกร้องสิทธิและ pins พื้นที่ว่างบัฟเฟอร์ ถ้าสภาพแวดล้อมมีหน่วยความจำจำกัด อาจเกิดปัญหาได้ ระวังว่าไฟล์บันทึกการติดตามและรายงาน อาจขยายใหญ่มาก

การจำกัดจำนวนของข้อมูลการติดตามที่เก็บรวบรวม

ตัวช่วยการติดตามจะสร้างวอลุ่มของข้อมูลที่มีขนาดใหญ่ ข้อมูลนี้ ไม่สามารถดักจับได้สำหรับระยะเวลาที่ขยายเพิ่มโดยไม่มีโอเวอร์โฟลว์อุปกรณ์หน่วยเก็บ

มีสองวิธีในการใช้ตัวช่วยการติดตามอย่างมีประสิทธิภาพ:

- ตัวช่วยการติดตามสามารถเปิดปละปิดใช้งานด้วยวิธีต่างๆ เพื่อดักจับกิจกรรมของระบบ ซึ่งเป็นไปได้ในการดักจับด้วยวิธีนี้ ซึ่งใช้เวลาวินาทีถึงนาทีของกิจกรรมของระบบ สำหรับการติดประกาศการประมวลผล นี่คือเวลาที่เพียงพอต่อการแสดงคุณสมบัติของ การทำรายการแอ็พพลิเคชันหลักและส่วนที่น่าสนใจของภารกิจที่ยาวนาน
- ตัวช่วยการติดตามสามารถปรับแต่งเพื่อสั่งออกคำสั่งสตรีมเหตุการณ์ในเอาต์พุต มาตรฐาน ซึ่งอนุญาตให้ประมวลผลแบบเรียลไทม์เพื่อเชื่อมต่อกับสตรีมเหตุการณ์ และจัดเตรียมการลดทอนข้อมูลตามเหตุการณ์ที่ได้บันทึกไว้ ด้วยวิธีการสร้างความสามารถในการมอนิเตอร์ในระยะยาว ตรรกะส่วนขยายสำหรับเครื่องมือพิเศษคือ การออกคำสั่ง data stream ไปยังอุปกรณ์ช่วยเสริมที่สามารถเก็บ จำนวนของข้อมูลขนาดใหญ่ หรือจัดเตรียมการลดทอนข้อมูลแบบไดนามิก อย่างใดอย่างหนึ่ง เทคนิคนี้จะถูกใช้โดย ทูลด้านประสิทธิภาพการทำงาน **tprof, pprof, netpmon** และ **filemon**

การเริ่มต้นและการควบคุมการติดตาม

ฟังก์ชันการติดตามมีโหมดการใช้งานที่แตกต่างกันสามโหมดดังนี้:

โหมดคำสั่งย่อย

การติดตามเริ่มต้นขึ้นด้วยคำสั่ง shell (**trace**) และดำเนินการโต้ตอบกับผู้ใช้ผ่านทางคำสั่งย่อย เวิร์กโหลดที่กำลังติดตามต้องมาจาก กระบวนการอื่น เนื่องจากกระบวนการ shell ดั้งเดิมถูกใช้งานอยู่

โหมดคำสั่ง

การติดตามเริ่มต้นขึ้นด้วยคำสั่ง shell (**trace –a**) ที่มีแฟล็กซึ่งระบุว่าฟังก์ชันการติดตามจะรันแบบอะซิงโครนัส กระบวนการ shell ดั้งเดิมว่างและสามารถรันคำสั่งธรรมดาร่วมกับ คำสั่งควบคุมการติดตามได้

โหมดแอ็พพลิเคชันที่ควบคุม

การติดตามเริ่มต้นขึ้นด้วยรูทีนย่อย **trcstart( )** และควบคุมโดย การเรียกรูทีนย่อย เช่น **trcon( )** และ **trcoff( )** จากแอ็พพลิเคชัน โปรแกรม

การจัดรูปแบบข้อมูลการติดตาม

วัตถุประสงค์โดยทั่วไปของตัวช่วยรายงานการติดตามจะถูกจัดเตรียมไว้ด้วยคำสั่ง **trcrpt**

ตัวช่วยรายงานจะจัดเตรียมการลดทอนข้อมูลเพียงเล็กน้อย แต่จะแปลงกระแสเหตุการณ์แบบไบนารีไปเป็นการแสดงรายการ ASCII ที่สามารถอ่านได้ ข้อมูลสามารถแตกออกด้วยเครื่องอ่าน หรือเครื่องมือที่สามารถพัฒนาเพื่อลดข้อมูลในอนาคตได้

ตัวช่วยรายงาน จะแสดงข้อความและข้อมูลสำหรับแต่ละเหตุการณ์ตามกฎที่ได้จัดเตรียมไว้ในฟอร์แม็ตไฟล์การติดตาม ฟอร์แม็ตไฟล์การติดตามที่เป็นค่าดีฟอลต์คือ `/etc/trcfmt` ซึ่งจะมี stanza สำหรับ ID แต่ละตัว stanza สำหรับเหตุการณ์จะจัดเตรียมตัวช่วยรายงาน พร้อมกับกฎการจัดรูปแบบสำหรับเหตุการณ์นั้น เทคนิคนี้อนุญาตให้ผู้ใช้ เพิ่มเหตุการณ์ของตนเองให้กับโปรแกรม และแทรก stanza ของเหตุการณ์ที่สอดคล้องกันในฟอร์แม็ตไฟล์เพื่อระบุวิธีที่ควรจัดรูปแบบเหตุการณ์ใหม่

การดูข้อมูลการติดตาม

เมื่อจัดรูปแบบข้อมูลการติดตาม โดยปกติ ข้อมูลทั้งหมดสำหรับเหตุการณ์ที่ กำหนดจะวางอยู่บนบรรทัดเดียว

บรรทัดเพิ่มเติมอาจมีข้อมูลคำอธิบาย ขึ้นอยู่กับฟิลด์ที่รวมไว้ บรรทัดที่จัดรูปแบบอาจมีมากกว่า 80 อักขระ ที่ดีที่สุดคือการดูรายงานบนอุปกรณ์เอาต์พุตที่สนับสนุน 132 คอลัมน์

ตัวอย่างการใช้ตัวช่วยการติดตาม

การติดตามโดยทั่วไปเกี่ยวข้องกับการขอรับ การจัดรูปแบบ การกรอง การอ่านไฟล์การติดตาม

หมายเหตุ: ตัวอย่างนี้จะมีความหมายเป็นอย่างมาก หากไฟล์อินพุตยังไม่ได้แคชในหน่วยความจำระบบ เลือกไฟล์ใดๆ ที่เป็นไฟล์ต้นฉบับ ซึ่งมีขนาด 50 KB และยังไม่ได้ใช้งาน

การได้รับไฟล์การติดตามตัวอย่าง

ข้อมูลการติดตามสะสมเพิ่มขึ้นอย่างรวดเร็วมาก วงเล็บชุดข้อมูลให้ใกล้เคียงกับ พื้นที่ที่สนใจให้มากที่สุดเท่าที่จะเป็นไปได้ เทคนิคอย่างหนึ่งในการทำเช่นนี้คือ การออกใช้หลายคำสั่งบนบรรทัดคำสั่งเดียวกัน

ตัวอย่างเช่น:

```
# trace -a -k "20e,20f" -o trc_raw ; cp ../bin/track /tmp/junk ; trcstop
```

ตรวจจับ การดำเนินการของคำสั่ง **cp** เราใช้คุณลักษณะสองอย่าง ของคำสั่ง **trace** อ็อพชัน **-k "20e,20f"** หยุดชุดของเหตุการณ์จากฟังก์ชัน **lockl( )** และ **unlockl( )** การเรียกเหล่านี้มีจำนวนมากบนระบบแบบตัวประมวลผลเดียว แต่ไม่ใช่บนระบบ SMP และเพิ่มวอลุ่มในรายงานโดยไม่ให้ข้อมูลเพิ่มเติมแก่เรา อ็อพชัน **-o trc_raw** ทำให้มีการบันทึกไฟล์เอาต์พุตการติดตามแบบ raw ลงในโลคัล ไดเร็กทอรีของเรา

การจัดรูปแบบตัวอย่างการติดตาม

ใช้คำสั่ง **trcrpt** เพื่อจัดรูปแบบรายงานการติดตาม

```
# trcrpt -O "exec=on,pid=on" trc_raw > cp.rpt
```

คำสั่งนี้ จะรายงานทั้งชื่อที่ได้ผ่านการรับรองของไฟล์ที่รันอยู่ และ ID การประมวลผล ที่กำหนดให้กับการประมวลผลนั้น

ไฟล์รายงานจะแสดงให้เห็นว่า เพจ VMM จำนวนมากกำหนดและลบออกจาเหตุการณ์ในการติดตาม เช่นเดียวกับลำดับต่อไปนี้ :

```
1B1 ksh            8526          0.003109888       0.162816
         VMM page delete:      V.S=0000.150E ppage=1F7F
             working_storage delete_in_progress process_private computational

1B0 ksh            8526          0.003141376       0.031488
         VMM page assign:      V.S=0000.2F33 ppage=1F7F
             working_storage delete_in_progress process_private computational
```

เราไม่สนใจ ในระดับของรายละเอียดกิจกรรม VMM ในขณะนี้ ดังนั้น เราจะจัดรูปแบบการติดตามใหม่ดังนี้:

```
# trcrpt -k "1b0,1b1" -O "exec=on,pid=on" trc_raw > cp.rpt2
```

อ็อพชัน **-k "1b0,1b1"** จะยับยั้งเหตุการณ์ VMM ที่ไม่ต้องการในเอาต์พุตที่จัดรูปแบบแล้ว ซึ่งจะช่วยเราไม่ต้องติดตามเวิร์กโหลดใหม่อีกครั้งสำหรับการยับยั้งเหตุการณ์ที่ไม่ต้องการ เราสามารถมีฟังก์ชัน **-k** ที่นำมาใช้ได้ของคำสั่ง **trcrpt** แทนคำสั่ง **trace** เพื่อยับยั้งเหตุการณ์ **lockl( )** และ **unlockl( )** ถ้าเราเชื่อว่า เราอาจต้องการค้นหากิจกรรมการล็อกในบางจุด ถ้าเราสนใจเฉพาะชุดของเหตุการณ์ขนาดเล็ก เราสามารถระบุ **-d** *"hookid1,hookid2"* เพื่อสร้างรายงานพร้อมกับเหตุการณ์เหล่านี้ได้เท่านั้น เนื่องจาก ID ของ hook คือคอลัมน์ซ้ายสุดของรายงาน คุณสามารถคอมไพล์รายการของ hook ได้อย่างรวดเร็วเพื่อสอดแทรกหรือแยกออก รายการที่ครอบคลุมของ ID ของ hook การติดตามที่นิยามอยู่ในไฟล์ `/usr/include/sys/trchkid.h`

การอ่านรายงานการติดตาม

ส่วนหัวของรายงานการติดตามบอกคุณว่าการติดตามเกิดขึ้นเมื่อไรและที่ใด ตลอดจนคำสั่งที่ใช้เพื่อจัดทำรายงาน

ข้อมูลต่อไปนี้เป็นตัวอย่างส่วนหัว:

```
Thu Oct 28 13:34:05 1999
System: AIX texmex Node: 4
Machine: 000691854C00
Internet Protocol Address: 09359BBB 9.53.155.187
Buffering: Kernel Heap

trace -a -k 20e,20f -o trc_raw
```

เนื้อหาของรายงานถ้าแสดงขึ้นในแบบอักษรที่เล็กมากพอ มีลักษณะคล้าย ข้อมูลต่อไปนี้:

```
ID  PROCESS NAME  PID     ELAPSED_SEC    DELTA_MSEC  APPL  SYSCALL KERNEL  INTERRUPT
101 ksh           8526    0.005833472    0.107008          kfork LR = D0040AF8
101 ksh           7214    0.012820224    0.031744          execve LR = 10015390
134 cp            7214    0.014451456    0.030464          exec:  cmd=cp ../bin/track /tmp/junk pid=7214 tid=24713
```

ใน cp.rpt2 คุณสามารถเห็นข้อมูลต่อไปนี้:

- **fork( ), exec( )**, และกิจกรรม page fault ของกระบวนการ **cp**
- การเปิดไฟล์อินพุตสำหรับการอ่านและการสร้างไฟล์ /tmp/junk
- การเรียกระบบ **read( ) / write( )** ที่สืบเนื่องเพื่อจัดทำ สำเนา
- กระบวนการ **cp** จะถูกบล็อกในขณะที่รอให้ I/O สมบูรณ์ และมีการจัดส่งกระบวนการ **wait**
- วิธีการแปลการร้องขอโลจิคัลวอลุ่มเป็นการร้องขอฟิสิคัลวอลุ่ม
- ไฟล์จะถูกแม็พแทนการบัฟเฟอร์ในบัฟเฟอร์เคอร์นัลดั้งเดิม และการเข้าถึงการอ่านส่งผลให้เกิด page faults ที่ต้องแก้ไขโดย Virtual Memory Manager
- Virtual Memory Manager รับทราบการเข้าถึงตามลำดับและเริ่ม prefetch หน้าไฟล์
- ขนาดของ prefetch เริ่มใหญ่ขึ้นเมื่อการเข้าถึงตามลำดับเกิดขึ้นอย่างต่อเนื่อง
- เมื่อเป็นไปได้ ไดรเวอร์อุปกรณ์ดิสก์จะรวมการร้องขอหลายไฟล์เข้าเป็น การร้องขอ I/O เดียวในไดร์ฟ

เอาต์พุตการติดตามอาจดูมีข้อมูลมากมายในตอนแรก นี่เป็นตัวอย่างที่ดี ในการใช้เป็นเครื่องมือช่วยเรียนรู้ ถ้าคุณเข้าใจกิจกรรมที่อธิบายไว้ แสดงว่าคุณพร้อมแล้วที่จะ สามารถใช้ฟังก์ชันการติดตามเพื่อวิเคราะห์ปัญหาประสิทธิภาพ ระบบได้

การกรองรายงานการติดตาม

รายละเอียดแบบเต็มของข้อมูลการติดตามอาจไม่จำเป็นต้องมี คุณสามารถเลือกเหตุการณ์เฉพาะ ที่น่าสนใจและต้องการให้แสดง

ตัวอย่างเช่น ในบางครั้งจะมีข้อได้เปรียบในการค้นหาจำนวนครั้ง ที่เกิดเหตุการณ์ที่แน่นอน หากต้องการตอบคำถาม "จำนวนการเปิดที่เกิดขึ้น ในตัวอย่างสำเนาคือเท่าใด?" อันดับแรกให้ค้นหา ID เหตุการณ์สำหรับการเรียกของระบบ **open( )** ซึ่งสามารถทำได้ดังนี้:

```
# trcrpt -j | grep -i open
```

คุณควรสามารถมองเห็น ID เหตุการณ์ 15b ซึ่งเป็นเหตุการณ์ OPEN SYSTEM CALL ในตอนนี้ การประมวลผลข้อมูลจากตัวอย่างสำเนามีดังนี้:

```
# trcrpt -d 15b -O "exec=on" trc_raw
```

รายงาน จะถูกเขียนลงในเอาต์พุตมาตรฐาน และคุณสามารถพิจารณาจำนวนของรูทีนย่อย **open( )** ที่เกิดขึ้น ถ้าคุณต้องการดูเฉพาะรูทีนย่อย **open( )** ซึ่งถูกดำเนินการด้วยการประมวลผล **cp** ให้รันคำสั่งรายงานอีกครั้ง โดยใช้คำสั่งต่อไปนี้:

```
# trcrpt -d 15b -p cp -O "exec=on" trc_raw
```

การเริ่มต้นและการควบคุมการติดตามจากบรรทัดคำสั่ง

ฟังก์ชันการติดตามที่มีการตั้งค่าคอนฟิกและชุดข้อมูลที่เป็นอ็อพชัน เริ่มต้นขึ้นโดยคำสั่ง **trace** ไวยากรณ์โดยรายละเอียดซึ่งมีการอธิบายไว้ใน *Commands Reference, Volume 5*

หลังจากตั้งค่าคอนฟิกการติดตามโดยใช้คำสั่ง **trace** แล้ว มีตัวควบคุมเพื่อเปิดและปิดการรวบรวมข้อมูลและเพื่อหยุดฟังก์ชันการติดตาม คุณสามารถเรียกใช้ตัวควบคุมผ่านทาง: คำสั่งย่อย คำสั่ง และรูทีนย่อย อินเตอร์เฟสรูทีนย่อยมีการอธิบายไว้ใน “การเริ่มต้นและการควบคุมการติดตามจากโปรแกรม” ในหน้า 438

การควบคุมการติดตามในโหมดคำสั่งย่อย

ถ้ารูทีน **trace** ถูกปรับแต่งไว้โดยไม่ระบุอ็อพชัน **–a** รูทีนนั้นจะรันในโหมดคำสั่งย่อย

ขณะที่รันรูทีน trace ในโหมดคำสั่งย่อยแทนการรันในพร้อมต์ shell ปกติ พร้อมต์ของ ″–>″ จะแสดงขึ้น ในโหมดนี้ คำสั่งย่อยต่อไปนี้ จะถูกจดจำไว้:

trcon เริ่มต้นหรือกลับสู่การเก็บรวบรวมข้อมูลเหตุการณ์

trcoff หยุดเก็บรวบรวมข้อมูลเหตุการณ์ชั่วคราว

q หรือ quit
หยุดการเก็บรวบรวมข้อมูลเหตุการณ์และยกเลิกรูทีน **trace**

!command
รันคำสั่ง shell ที่ระบุ

? แสดงคำสั่งที่พร้อมใช้งาน

ตัวอย่างเช่น:

```
# trace -f -m "Trace of events during mycmd"
-> !mycmd
-> q
#
```

การควบคุมการติดตามด้วยคำสั่ง

มีคำสั่งจำนวนมากที่สามารถนำมาใช้เพื่อควบคุมรูทีน **trace**

ถ้ารูทีน **trace** ถูกปรับแต่งเพื่อรันแบบอะซิงโครนัส (**trace –a**) การติดตามสามารถควบคุมได้ด้วยคำสั่ง ต่อไปนี้:

trcon เริ่มต้นหรือกลับสู่การเก็บรวบรวมข้อมูลเหตุการณ์

trcoff หยุดเก็บรวบรวมข้อมูลเหตุการณ์ชั่วคราว

trcstop หยุดการเก็บรวบรวมข้อมูลเหตุการณ์และยกเลิกรูทีน **trace**

ตัวอย่างเช่น:

```
# trace -a -n -L 2000000 -T 1000000 -d -o trace.out
# trcon
# cp /a20kfile /b
# trcstop
```

ด้วยการระบุอ็อพชัน **–d** (เลื่อนการติดตามจนกว่าจะป้อนคำสั่งย่อย **trcon**) คุณสามารถจำกัดจำนวนของการติดตามที่ทำบนคำสั่ง **trace** ด้วยตัวของคำสั่งเอง ถ้าไม่ได้ระบุอ็อพชัน **–d** ไว้ การติดตามจะเริ่มต้นขึ้นโดยทันที และสามารถบันทึกเหตุการณ์สำหรับการกำหนดค่าเริ่มต้นของคำสั่ง **trace** ด้วยบัฟเฟอร์หน่วยความจำของตนเอง โดยทั่วไป เราต้องการติดตามทุกสิ่ง แต่คำสั่ง **trace** จะติดตามตัวคำสั่งเอง

ตามค่าดีฟอลต์ ขนาดบัฟเฟอร์เคอร์เนล (อ็อพชัน **-T**) สามารถมีค่าครึ่งหนึ่ง ของขนาดบัฟเฟอร์บันทึกการทำงาน (อ็อพชัน **-L**) ถ้าคุณใช้แฟล็ก **-f** ขนาดของบัฟเฟอร์อาจเหมือนกันได้

อ็อพชัน **-n** จะมีประโยชน์ถ้ามีส่วนขยายเคอร์เนลการเรียกของระบบ ที่จำเป็นต้องติดตาม

การเริ่มต้นและการควบคุมการติดตามจากโปรแกรม

ฟังก์ชันการติดตามสามารถเริ่มต้นจากโปรแกรมได้โดยใช้การเรียก รูทีนย่อย รูทีนย่อยนั้นคือ **trcstart( )** และอยู่ในไลบรารี `librts.a`

ไวยากรณ์ของรูทีนย่อย **trcstart( )** มีดังนี้:

```
int trcstart(char *args)
```

โดยที่ *args* คือรายการอ็อพชันที่คุณจะต้องระบุสำหรับคำสั่ง **trace** โดยค่าดีฟอลต์ การติดตามระบบ (แชนเนล 0) มีการเริ่มต้นขึ้น ถ้าคุณต้องการเริ่มต้น การติดตามทั่วไป ให้รวมอ็อพชัน **-g** ในสตริง *args* เมื่อเสร็จเรียบร้อยแล้ว รูทีนย่อย **trcstart( )** จะส่งคืนแชนเนล ID สำหรับการติดตามทั่วไป สามารถใช้แชนเนล ID นี้เพื่อบันทึกแชนเนลทั่วไป ส่วนตัวได้

เมื่อคอมไพล์โปรแกรมโดยใช้รูทีนย่อยนี้ ต้องมีการร้องขอลิงก์ไปยังไลบรารี `librts.a` (ใช้ **-lrts** เป็นคอมไพล์อ็อพชัน)

การควบคุมการติดตามด้วยการเรียกรูทีนย่อย trace

การควบคุมรูทีน **trace** จะพร้อมใช้งานเป็นรูทีนย่อยจากไลบรารี `librts.a`

รูทีนย่อยจะส่งคืนค่าศูนย์สำหรับการดำเนินการที่เสร็จสิ้นอย่างสำเร็จผล รูทีนย่อย คือ:

int trcon( )
: เริ่มต้นหรือกลับสู่การทำงานการรวบรวมข้อมูลการติดตาม

int trcoff( )
: หยุดทำการเก็บรวบรวมข้อมูลการติดตามชั่วคราว

int trcstop( )
: หยุดการเก็บรวบรวมข้อมูลการติดตามและยกเลิกรูทีน **trace**

การใช้คำสั่ง trcrpt เพื่อจัดรูปแบบรายงาน

ฟังก์ชันรายงานการติดตามอ่านไฟล์บันทึกการติดตาม จัดรูปแบบ รายการการติดตาม และเขียนรายงาน

คำสั่ง **trcrpt** แสดงข้อความและข้อมูลสำหรับแต่ละเหตุการณ์ ตามกฎที่ระบุในไฟล์รูปแบบการติดตาม (`/etc/trcfmt`) Stanzas ในไฟล์รูปแบบแสดงกฎการจัดรูปแบบสำหรับเหตุการณ์หรือ hooks ผู้ใช้ที่เพิ่ม hooks ในโปรแกรมสามารถแทรก stanzas เหตุการณ์ที่ตรงกันในไฟล์รูปแบบ เพื่อพิมพ์ข้อมูลการติดตาม (โปรดดู “การเพิ่มเหตุการณ์ติดตามใหม่” ในหน้า 440)

ฟังก์ชัน **trcrpt** ไม่ได้จัดทำรายงานสรุปใดๆ แต่คุณสามารถใช้คำสั่ง **awk** เพื่อสร้างสรุปแบบง่าย ผ่านทางการประมวลผลเพิ่มเติมของเอาต์พุต **trcrpt**

ไวยากรณ์โดยละเอียดของคำสั่ง **trcrpt** มีการอธิบายไว้ใน *Commands Reference, Volume 5*

การจัดรูปแบบรายงานบนระบบเดียวกัน

คำสั่ง **trcrpt** จัดรูปแบบรายงานของข้อมูลการติดตามเหตุการณ์ ที่มีอยู่ในไฟล์บันทึกการติดตาม

คุณสามารถระบุเหตุการณ์ที่จะสอดแทรก (หรือละเว้น) ในรายงาน และกำหนดการนำเสนอเอาต์พุตด้วยคำสั่งนี้

คุณสามารถใช้ System Management Interface Tool (SMIT) เพื่อรันคำสั่ง **trcrpt** โดยพิมพ์วิธีลัด SMIT:

```
# smitty trcrpt
```

หากต้องการสร้าง รายงานการติดตามลงในไฟล์ `newfile` ให้พิมพ์:

```
# trcrpt -o newfile
```

การจัดรูปแบบรายงานบนระบบที่แตกต่างกัน

บ่อยครั้งที่มีความต้องการในการรันคำสั่ง **trcrpt** บนระบบอื่นที่ไม่ใช่ที่เก็บการติดตาม

มีหลายเหตุผลสำหรับเหตุการณ์นี้ เช่น:

- ระบบที่กำลังติดตามอาจไม่พร้อมใช้งานเพื่อให้คุณรันคำสั่ง **trcrpt** และการติดตามอาจถูกเก็บรวบรวมโดยผู้ดูแลระบบหรือใครบางคนที่ไซต์รีโมต
- ระบบที่กำลังติดตามไม่ว่างเพื่อให้คุณรันคำสั่ง **trcrpt**
- ระบบที่กำลังติดตามไม่มีพื้นที่ระบบไฟล์เพียงพอ เพื่อให้เหมาะสมกับไฟล์ **trcrpt** ที่มีขนาดใหญ่มาก

คุณสามารถรันคำสั่ง **trace** บนระบบได้ และรันคำสั่ง **trcrpt** บนไฟล์การติดตามนั้นบนระบบอื่นได้ เพื่อให้การดำเนินการนี้ทำงานได้อย่างถูกต้อง เอาต์พุตของคำสั่ง **trcnm** อาจต้องการจากระบบที่การติดตามรันอยู่ รันคำสั่ง **trcnm** และเปลี่ยนทิศทางเอาต์พุตลงในไฟล์ได้ดังนี้:

```
# trcnm > trace.nm
```

ถ้า คุณต้องการใช้ไฟล์การติดตามสำหรับทูลด้านประสิทธิภาพการทำงานอื่นๆ เช่น **tprof**, **pprof**, **netpmon** และ **filemon** ให้รันคำสั่ง **gennames** *Gennames_File*

ไฟล์นั้น จะถูกใช้พร้อมกับแฟล็ก **-n** ของคำสั่ง **trcrpt** ดังต่อไปนี้:

```
# trcrpt -n trace.nm -o newfile
```

ถ้าไม่ได้ระบุ **-n** ไว้ คำสั่ง **trcrpt** จะสร้างตารางสัญลักษณ์ จากระบบที่คำสั่ง **trcrpt** รันอยู่

นอกจากนี้ สำเนาของไฟล์ `/etc/trcfmt` จากระบบที่กำลังติดตาม อาจมีประโยชน์ เนื่องจากระบบนั้นอาจมี stanza รูปแบบการติดตามที่แตกต่าง หรือมากกว่าระบบที่คำสั่ง **trcrpt** กำลังรันอยู่ คำสั่ง **trcrpt** สามารถใช้แฟล็ก **-t** เพื่อระบุไฟล์รูปแบบการติดตาม (ตามค่าดีฟอลต์ใช้ไฟล์ `/etc/trcfmt` จากระบบที่คำสั่ง **trcrpt** กำลังรัน) ตัวอย่างเช่น:

```
# trcrpt -n trace.nm -t trcfmt_file -o newfile
```

การจัดรูปแบบรายงานจากการติดตามเอาต์พุต -C

ถ้าการติดตามรันด้วยแฟล็ก **-C** ไฟล์เอาต์พุตการติดตามตั้งแต่หนึ่งไฟล์ขึ้นไป จะถูกสร้างขึ้น

ตัวอย่างเช่น ถ้าระบุชื่อไฟล์การติดตามเป็น `trace.out` และระบุ **-C all** ไว้บน SMP แบบ 4 ทิศทาง ดังนั้นไฟล์ `trace.out`, `trace.out-1`, `trace.out-2`, `trace.out-3` และ `trace.out-4` จะถูกสร้างขึ้น เมื่อคุณรันคำสั่ง **trcrpt** ให้ระบุ **trcrpt -C all** และ `trace.out` ให้เป็นชื่อไฟล์ และไฟล์ทั้งหมดจะถูกอ่านดังต่อไปนี้:

```
# trcrpt -C all -r trace.out > trace.tr
```

ไฟล์ `trace.tr` นี้สามารถนำมาใช้เป็นอินพุตสำหรับคำสั่งอื่นๆ (ซึ่งจะสอดแทรกข้อมูลการติดตามจาก CPU แต่ละตัว) เหตุผลสำหรับแฟล็ก **-C** เกี่ยวกับการติดตามคือ การติดตามสามารถติดตามกิจกรรมของ CPU แต่ละตัวบนระบบเหล่านั้นได้ ซึ่งมี CPU จำนวนมาก (ตัวอย่างเช่น มากกว่า 12) เหตุผลอื่นๆ คือ ขนาดของบัฟเฟอร์สำหรับบัฟเฟอร์การติดตาม จะมีค่าต่อ CPU เมื่อคุณใช้แฟล็ก **-C all**

การเพิ่มเหตุการณ์ติดตามใหม่

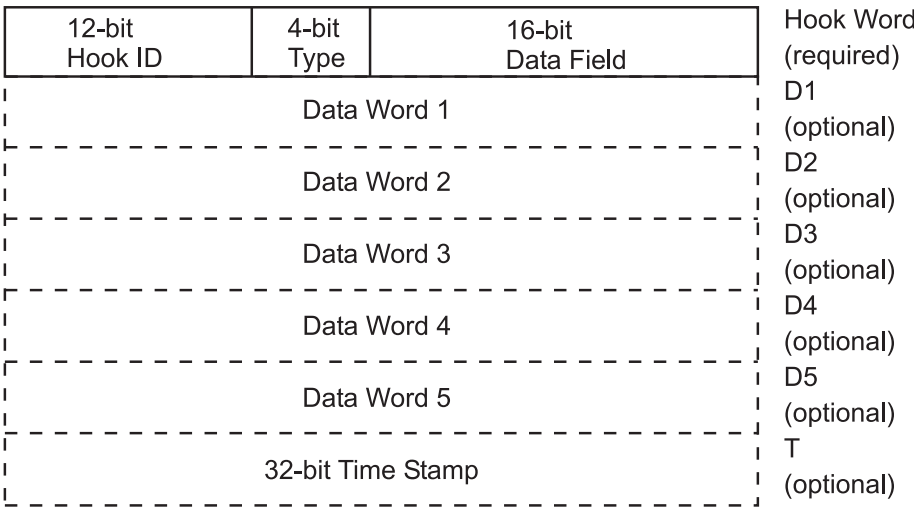
ระบบปฏิบัติการจะถูกจัดส่งเครื่องมือมาพร้อมกับคีย์เหตุการณ์ ผู้ใช้ต้องการเพียงแค่ เรียกทำงานการติดตามเท่านั้น เพื่อดักจับการไหลของเหตุการณ์จาก ระบบปฏิบัติการ ผู้พัฒนาแอ็พพลิเคชันอาจต้องการเครื่องมือสำหรับโค้ดแอ็พพลิเคชันของตนเอง ในระหว่างการพัฒนาสำหรับวัตถุประสงค์ในการปรับ ซึ่งจะจัดเตรียมข้อมูลโดยละเอียดเกี่ยวกับ วิธีการที่แอ็พพลิเคชันจะโต้ตอบกับระบบ

หากต้องการเพิ่มเหตุการณ์ติดตามใหม่ คุณต้องออกแบบการติดตามเร็กคอร์ด ที่สร้างโดยโปรแกรมของคุณในตามระเบียบอินเตอร์เฟส จากนั้น คุณจะเพิ่มแมโคร trace-hook ให้กับโปรแกรมที่ตำแหน่งที่เหมาะสม การติดตามสามารถใช้ผ่านวิธีมาตรฐานของการเรียกใช้ และการควบคุมการติดตามใดๆ ได้ (คำสั่ง คำสั่งย่อย หรือการเรียกรูทีนย่อย) หากต้องการใช้โปรแกรม **trcrpt** เพื่อจัดรูปแบบการติดตามของคุณ ให้เพิ่ม stanza ที่อธิบายถึงเร็กคอร์ดการติดตามใหม่แต่ละเร็กคอร์ด แซะการจัดรูปแบบข้อกำหนดให้กับไฟล์รูปแบบการติดตาม

รูปแบบที่เป็นไปได้ของเร็กคอร์ดเหตุการณ์การติดตาม

เหตุการณ์ประกอบด้วยคำ hook, คำข้อมูลที่เป็นอ็อพชัน และเวลาประทับ

ดังแสดงในรูปภาพต่อไปนี้ มีการกำหนดชนิดสี่บิตสำหรับแต่ละรูปแบบ ที่เร็กคอร์ดเหตุการณ์สามารถใช้ได้ ฟิลด์ชนิดสร้างขึ้นโดยรูทีนการบันทึก เพื่อให้ฟังก์ชันรายงานสามารถข้ามจากเหตุการณ์หนึ่งไปยังอีกเหตุการณ์หนึ่งได้เสมอ ในขณะประมวลผลข้อมูล แม้ว่ากฎการจัดรูปแบบในไฟล์รูปแบบการติดตามจะ ไม่ถูกต้องหรือขาดไปสำหรับเหตุการณ์นั้น



รูปที่ 26. รูปแบบของเร็กคอร์ดเหตุการณ์การติดตาม. ภาพสาธิตนี้เป็นตาราง ที่มี 7 แถว เซลล์ในแถวแรกมีป้ายชื่อว่า 12-bit hook ID, 4-bit Type และ 16-bit Data Field อีก 6 แถวถัดไป มีป้ายชื่อว่า Data Word 1 ถึง Data Word 5 และแถวสุดท้ายมีป้ายชื่อว่า 32-bit Time Stamp ส่วนหัวแถวของแถวที่ 1 คือ Hook Word (บังคับ) อีก 5 แถวถัดไปมีป้ายชื่อว่า D1 (เป็นอ็อพชัน), D2 (เป็นอ็อพชัน), D3 (เป็นอ็อพชัน), D4 (เป็นอ็อพชัน), และ D5 (เป็นอ็อพชัน) แถวสุดท้ายมีป้ายชื่อว่า T (บังคับ)

เร็กคอร์ดเหตุการณ์ควรสั้นที่สุดเท่าที่เป็นไปได้ เหตุการณ์ระบบจำนวนมาก ใช้เฉพาะคำ hook และเวลาประทับเท่านั้น รูปแบบยาวช่วยให้ผู้ใช้สามารถบันทึก ข้อมูลที่มีความยาวผันแปรได้ ในรูปแบบยาวนี้ ฟิลด์ข้อมูล 16 บิต ของคำ hook ถูกแปลงเป็นฟิลด์ความยาวที่อธิบาย ความยาวของเร็กคอร์ดเหตุการณ์

แชนเนลการติดตาม

ฟังก์ชันการติดตามสามารถสนับสนุนแชนเนลของกิจกรรม trace-hook ได้มากถึงแปดแชนเนลพร้อมกัน ซึ่งมีหมายเลข 0-7

แชนเนล 0 ใช้สำหรับเหตุการณ์ระบบเสมอ แต่เหตุการณ์แอ็พพลิเคชันสามารถ ใช้แชนเนลนี้ได้เช่นกัน อีกเจ็ดแชนเนลที่เรียกว่าแชนเนลทั่วไป สามารถใช้สำหรับ การติดตามกิจกรรมแอ็พพลิเคชันโปรแกรม

เมื่อเริ่มต้นการติดตาม มีการใช้แชนเนล 0 โดยค่าดีฟอลต์ คำสั่ง **trace -n** *channel_number* เริ่มต้นการติดตามที่แชนเนลทั่วไป การใช้แชนเนลทั่วไปมีข้อจำกัดบางอย่างดังนี้:

- อินเตอร์เฟสไปยังแชนเนลทั่วไปใช้เวลา CPU มากกว่าอินเตอร์เฟสไปยัง แชนเนล 0 เนื่องจากต้องแยกความแตกต่างระหว่างแชนเนล และเนื่องจาก แชนเนลทั่วไปบันทึกเร็กคอร์ดที่มีความยาวผันแปร
- เหตุการณ์ที่บันทึกบนแชนเนล 0 และบนแชนเนลทั่วไปสามารถเชื่อมโยงโดย เวลาประทับเท่านั้น ไม่ใช่โดยลำดับ ดังนั้นจึงอาจมีสถานการณ์ซึ่งไม่สามารถกำหนด ได้ว่าเหตุการณ์ใดเกิดขึ้นก่อน

แมโครสำหรับการทำบันทึกเหตุการณ์การติดตาม

แมโครที่บันทึกชนิดของเหตุการณ์การบันทึกที่เป็นไปได้จะถูกกำหนดอยู่ในไฟล์ `/usr/include/sys/trcmacros.h`

ID เหตุการณ์จะถูกกำหนดในไฟล์ `/usr/include/sys/trchkid.h` สอดแทรกสองไฟล์นี้ในโปรแกรมใดๆ ที่กำลังบันทึกเหตุการณ์การติดตาม

แมโครที่บันทึกเหตุการณ์บนช่องสัญญาณ 0 ด้วยการประทับเวลาจะเป็นดังนี้:

```
TRCHKLOT(hw)
TRCHKL1T(hw,D1)
TRCHKL2T(hw,D1,D2)
TRCHKL3T(hw,D1,D2,D3)
TRCHKL4T(hw,D1,D2,D3,D4)
TRCHKL5T(hw,D1,D2,D3,D4,D5)
```

ในเวอร์ชันก่อนหน้าของ AIX ให้ใช้แมโครต่อไปนี้เพื่อบันทึกเหตุการณ์บนช่องสัญญาณ 0 โดยไม่มีการประทับ เวลา:

```
TRCHKL0(hw)
TRCHKL1(hw,D1)
TRCHKL2(hw,D1,D2)
TRCHKL3(hw,D1,D2,D3)
TRCHKL4(hw,D1,D2,D3,D4)
TRCHKL5(hw,D1,D2,D3,D4,D5)
```

เหตุการณ์การติดตามทั้งหมดจะถูกประทับเวลาโดยไม่คำนึงถึงแมโครที่ใช้

ฟิลด์ชนิดของเร็กคอร์ดเหตุการณ์ที่ติดตามจะตั้งค่าที่สอดคล้องกับแมโครที่ใช้โดยไม่พิจารณาถึงค่าเหล่านั้นที่มีขนาดสี่บิตใน พารามิเตอร์ `hw`

เฉพาะเหตุการณ์การบันทึกแมโครทั้งสองเหตุการณ์ในหนึ่งช่องสัญญาณทั่วไป (1–7) ซึ่งจะเป็นดังนี้ :

```
TRCGEN(ch,hw,D1,len,buf)
TRCGENT(ch,hw,D1,len,buf)
```

การบันทึกแมโครเหล่านี้ในสตรีมเหตุการณ์ที่ระบุโดยพารามิเตอร์ช่องสัญญาณ (`ch`) hook word (`hw`) data word (`D1`) และ `len` ไบต์จากเซ็กเมนต์ข้อมูลของผู้ใช้ที่เริ่มต้นที่ตำแหน่งที่ระบุโดย `buf`

การใช้ของ IDs เหตุการณ์

ID เหตุการณ์ในเร็กคอร์ดการติดตามระบุว่าเร็กคอร์ดเป็นสมาชิกของ คลาสเฉพาะของเร็กคอร์ด ID เหตุการณ์คือข้อมูลพื้น ฐานซึ่งกลไก การติดตามบันทึกหรือละเว้น trace hooks และเป็นข้อมูลพื้นฐานซึ่งคำสั่ง **trcrpt** รวมหรือแยกเร็กคอร์ดการติด ตามในรายงานที่จัดรูปแบบ

ก่อนหน้า AIX 6.1 และบนแอ็พพลิเคชัน 32 บิตที่กำลังรันบน AIX 6.1 และสูงกว่า IDs เหตุการณ์เป็น 12 บิต (สามเลขฐานสิบ หก) สำหรับ 4096 IDs ที่เป็นไปได้ IDs เหตุการณ์ที่สำรองและจัดส่งพร้อมกับรหัสมีการกำหนดให้อย่างถาวร เพื่อหลีกเลี่ยง การทำซ้ำ เพื่อให้ผู้ใช้สามารถกำหนดเหตุการณ์ในสภาพแวดล้อมของตน หรือในระหว่างการพัฒนาได้ ช่วงของ IDs เหตุการณ์ ตั้งแต่ 0x010 ถึง hex 0x0FF จึงถูกสำรองไว้สำหรับการใช้งานชั่วคราว ผู้ใช้สามารถใช้ IDs ในช่วงนี้ได้อย่างอิสระ ในสภาพแวด ล้อมของตนเอง (นั่นคือ ภายในชุดของระบบซึ่งผู้ใช้จัดเตรียมเพื่อให้ แน่ใจว่าไม่มีการใช้ ID เหตุการณ์เดียวกันอย่างคลุม เครือ)

ในแอ็พพลิเคชัน 64 บิตและเคอร์นัลรูทีนที่กำลังรัน AIX 6.1 และ สูงกว่า คุณสามารถใช้ IDs เหตุการณ์ 16 บิต (สี่เลขฐานสิบ หก) สำหรับ 65536 IDs ที่เป็นไปได้ IDs เหตุการณ์ที่น้อยกว่า 0x1000 ต้องมีตำแหน่งสำคัญน้อยที่สุดของ 0 (ในรูปแบบของ "0x0hh0") เพื่อให้ผู้ใช้สามารถกำหนดเหตุการณ์ในสภาพแวดล้อมของตน หรือในระหว่างการพัฒนาได้ ช่วงของ IDs เหตุ การณ์ตั้งแต่ 0x0100 ถึง 0x0FF0 จึงถูกสำรองไว้สำหรับการใช้งานชั่วคราว

หมายเหตุ: สิ่งสำคัญคือผู้ใช้ที่ใช้ช่วงเหตุการณ์นี้ต้องไม่ปล่อยให้รหัสออกจาก สภาพแวดล้อมของพวกเขา ถ้าคุณจัดส่งโค้ด เครื่องมือที่มี hook IDs ชั่วคราวไปยังสภาพแวดล้อมซึ่งคุณไม่ได้ควบคุมการใช้ของ IDs คุณ เสี่ยงในการปะทะกับโปรแกรมอื่น ที่ใช้ IDs เดียวกันอยู่แล้วในสภาพแวดล้อมนั้น

IDs เหตุการณ์ควรมีการสงวนไว้เนื่องจากมีอยู่น้อยมาก แต่สามารถขยายได้โดยใช้ฟิลด์ข้อมูล 16 บิต ซึ่งส่งผลให้มีเหตุการณ์ที่แยกแยะได้ที่อาจเป็นไปได้จำนวน 65536 เหตุการณ์สำหรับทุก hook ID ที่เป็นทางการ เหตุการณ์เดียวสำหรับการมี ID เฉพาะคือ ID เป็นระดับซึ่งมีการรวบรวมและการกรองรายงาน ในฟังก์ชันการติดตาม

เหตุการณ์ที่ผู้ใช้เพิ่มสามารถจัดรูปแบบได้โดยใช้คำสั่ง **trcrpt** ถ้ามี stanza สำหรับเหตุการณ์อยู่ในไฟล์รูปแบบการติดตามที่ระบุ ไฟล์รูปแบบการติดตามเป็นไฟล์ ASCII ที่แก้ไขได้ (โปรดดู “ไวยากรณ์สำหรับ stanzas ในไฟล์รูปแบบการติดตาม” ในหน้า 444)

ตัวอย่างของการโค้ดและการจัดรูปแบบเหตุการณ์

เหตุการณ์การติดตามสามารถนำมาใช้ในเวลาที่ประมวลผลลูปของโปรแกรม

```
#include <sys/trcctl.h>
#include <sys/trcmacros.h>
#include <sys/trchkid.h>
char *ctl_file = "/dev/systrctl";
int ctlfd;
int i;
main()
{
  printf("configuring trace collection \n");
  if (trcstart("-ad")){
    perror("trcstart");
    exit(1);
  }

  printf("opening the trace device  \n");
  if((ctlfd = open(ctl_file,0))<0){
    perror(ctl_file);
    exit(1);
  }

  printf("turning  trace on \n");
  if(ioctl(ctlfd,TRCON,0)){
    perror("TRCON");
    exit(1);
  }

  for(i=1;i<11;i++){
    TRCHKL1T(HKWD_USER1,i);

    /* The code being measured goes here. The interval */
    /* between occurrences of HKWD_USER1 in the trace  */
    /* file is the total time for one iteration. */
  }

  printf("turning trace off\n");
  if(ioctl(ctlfd,TRCSTOP,0)){
    perror("TRCOFF");
    exit(1);
  }

  printf("stopping the trace daemon \n");
  if (trcstop(0)){
```

```
    perror("trcstop");
    exit(1);
  }

  exit(0);
}
```

เมื่อคุณคอมไพล์โปรแกรมตัวอย่าง คุณต้องลิงก์ไปยังไลบรารี `librts.a` ได้ดังนี้:

```
# xlc -O3 sample.c -o sample -l rts
```

HKWD_USER1 คือ ID ของเหตุการณ์ที่มีเลขฐานสิบหก 010 (คุณสามารถตรวจสอบค่านี้ได้โดยตรวจสอบไฟล์ `/usr/include/sys/trchkid.h`) ตัวช่วยรายงาน ไม่ได้จัดรูปแบบเหตุการณ์ HKWD_USER1 เว้นเสียแต่กฎจะถูกจัดเตรียมไว้ในไฟล์รูปแบบการติดตาม ตัวอย่างต่อไปนี้ของ stanza สำหรับ HKWD_USER1 สามารถนำมาใช้เป็น:

```
# User event HKWD_USER1 Formatting Rules Stanza
# An example that will format the event usage of the sample program
010 1.0 L=APPL "USER EVENT - HKWD_USER1" O2.0      \n \
               "The # of loop iterations =" U4     \n \
               "The elapsed time of the last loop = " \
                endtimer(0x010,0x010) starttimer(0x010,0x010)
```

เมื่อคุณป้อนตัวอย่าง stanza ห้ามแก้ไขไฟล์รูปแบบต้นฉบับ `/etc/trcfmt` แต่ทำสำเนาและเก็บไว้ในไดเร็กทอรีของตนเอง (สมมุติว่าคุณตั้งชื่อ **mytrcfmt**) เมื่อคุณรันโปรแกรมตัวอย่าง ข้อมูลเหตุการณ์ดิบจะถูกดักจับในไฟล์บันทึกการทำงานที่เป็นค่าดีฟอลต์ เนื่องจากไม่มีไฟล์บันทึกการทำงานที่ระบุไว้ในรูทีนย่อย **trcstart( )** คุณสามารถกรองรายงานเอาต์พุต เพื่อขอรับเหตุการณ์ของคุณ หากต้องการทำสิ่งนี้ ให้รันคำสั่ง **trcrpt** ดังต่อไปนี้:

```
# trcrpt -d 010 -t mytrcfmt -O "exec=on" > sample.rpt
```

คุณสามารถเรียกไฟล์ `sample.rpt` เพื่อดูผลลัพธ์ได้

ไวยากรณ์สำหรับ stanzas ในไฟล์รูปแบบการติดตาม

ไฟล์รูปแบบการติดตามมีกฎสำหรับการนำเสนอและการแสดงผล ของข้อมูลที่คาดไว้สำหรับแต่ละ ID เหตุการณ์ ซึ่งช่วยให้สามารถจัดรูปแบบรายงานใหม่ได้โดยไม่ต้องเปลี่ยนฟังก์ชันรายงาน

กฎสำหรับเหตุการณ์ใหม่จะถูกเพิ่มลงในไฟล์รูปแบบ ไวยากรณ์ของกฎ ช่วยให้การนำเสนอข้อมูลมีความยืดหยุ่น

Stanza รูปแบบการติดตามสามารถยาวได้มากตามความต้องการเพื่ออธิบายกฎ สำหรับเหตุการณ์เฉพาะ Stanza สามารถต่อไปยังบรรทัดถัดไปโดยการจบ บรรทัดปัจจุบันด้วยอักขระ '\' ฟิลด์มีการอธิบาย อยู่ใน *Files Reference*

ข้อคิดเห็นในไฟล์ `/etc/trcfmt` อธิบายรูปแบบอื่นๆ และแมโครที่เป็นไปได้ และอธิบายวิธีการที่ผู้ใช้สามารถกำหนดแมโครเพิ่มเติม

การรายงานปัญหาประสิทธิภาพ

ถ้าคุณเชื่อว่าคุณพบปัญหาประสิทธิภาพที่อาจเกิดขึ้นได้ใน ระบบปฏิบัติการ คุณสามารถใช้เครื่องมือและโพรซีเดอร์สำหรับการรายงาน ปัญหาและจัดส่งข้อมูลการวิเคราะห์ปัญหาได้ เครื่องมือเหล่านี้มีไว้เพื่อให้มั่นใจว่า คุณได้รับพร้อมต์และการตอบกลับที่ถูกต้อง โดยใช้ความพยายามและเวลาในส่วนของ คุณน้อยที่สุด

การวัดเส้นบรรทัด

ปัญหาเกี่ยวกับผลการทำงานจะรายงานโดยทันทีพร้อมกับการเปลี่ยนแปลงในระบบฮาร์ดแวร์ หรือซอฟต์แวร์ เว้นเสียแต่มีการวัดค่าเส้นบรรทัดการเปลี่ยนแปลงล่วงหน้า ซึ่งจะเปรียบเทียบผลการทำงานแบบติดประกาศการเปลี่ยนแปลง การรับรองจำนวนของปัญหา ที่เป็นไปไม่ได้

การเปลี่ยนข้อมูลต่อไปนี้อาจมีผลต่อผลการทำงาน:

- ฮาร์ดแวร์คอนฟิกูเรชัน – การเพิ่ม การลบ หรือการเปลี่ยนคอนฟิกูเรชัน เช่น วิธีการเชื่อมต่อดิสก์
- ระบบปฏิบัติการ – การติดตรั้ง หรืออัพเดตชุดไฟล์ การติดตั้ง PTF และการเปลี่ยนพารามิเตอร์
- แอ็พพลิเคชัน – การติดตั้งเวอร์ชันใหม่และโปรแกรมฟิกซ์
- แอ็พพลิเคชัน – การปรับแต่งหรือการเปลี่ยนการกำหนดตำแหน่งข้อมูล
- การปรับแอ็พพลิเคชัน
- อ็อพชันการปรับในระบบปฏิบัติการ RDBMS หรือแอ็พพลิเคชัน
- การเปลี่ยนแปลงใดๆ

อ็อพชันที่ดีที่สุดคือ การวัดสภาวะแวดล้อมก่อน และหลังการเปลี่ยนแปลงแต่ละครั้ง ทางเลือกหนึ่งคือ การรันการวัดค่าที่ช่วงเวลาปกติ (ตัวอย่างเช่น หนึ่งครั้งในหนึ่งเดือน) และบันทึกเอาต์พุต เมื่อพบปัญหา การดักจับก่อนหน้านี้สามารถนำมาใช้สำหรับการเปรียบเทียบ ซึ่งจะเก็บสะสมชุดของเอาต์พุต เพื่อสนับสนุนการวินิจฉัยปัญหาผลการทำงานที่เป็นไปได้

หากต้องการเพิ่มการวินิจฉัยผลการทำงาน ให้เก็บรวบรวมข้อมูลสำหรับระยะเวลาต่างๆ ของการทำงาน วัน สัปดาห์ หรือเดือน เมื่อผลการทำงานจะถูกส่งออก ตัวอย่างเช่น คุณอาจมีเวิร์กโหลดสูงสุดดังนี้:

- ในช่วงกลางของตอนเช้าสำหรับผู้ใช้ที่ออนไลน์
- ในระหว่าง การรันแบ็ตช์ตอนกลางคืน
- ในระหว่างการประมวลผลปลายเดือน
- ในระหว่างการโหลดข้อมูลหลัก

ใช้การวัดค่าเพื่อเก็บสะสมข้อมูลสำหรับการใช้งานเวิร์กโหลดสูงสุด เนื่องจากปัญหาเกี่ยวกับผลการทำงานอาจเป็นสาเหตุทำให้ปัญหาหนึ่งในระยะเวลาเหลานี้ และไม่ใช่ในครั้งอื่น

หมายเหตุ: การวัดค่ามีผลกระทบกับผลการทำงานของระบบ ที่ต้องการวัด
เครื่องมือเก็บสะสมข้อมูล AIXเครื่องมือการเก็บรวบรวมผลการทำงาน PMR (perfpmr) คือเมธอดที่ต้องการสำหรับการรวบรวมข้อมูล เส้นบรรทัด การเข้าถึงเครื่องมือเหล่านี้ผ่านเว็บที่ ftp://ftp.software.ibm.com/aix/tools/perftools/perfpmr ต่อไปนี้คือคำสั่งในไฟล์ README ในไดเร็กทอรีที่ตรงกับเวอร์ชัน AIX คุณจะวัดค่าที่ได้รับ ติดตั้ง และสะสมข้อมูลบนระบบของคุณ

ปัญหาประสิทธิภาพคืออะไร

เจ้าหน้าที่ฝ่ายสนับสนุนจำเป็นต้องกำหนดว่าปัญหาที่ได้รับรายงานเป็น ปัญหาเกี่ยวกับฟังก์ชันหรือปัญหาประสิทธิภาพ

เมื่อแอ็พพลิเคชัน ระบบฮาร์ดแวร์ หรือเครือข่ายไม่ได้ทำงาน อย่างถูกต้อง กรณีนี้เรียกว่า *ปัญหาเกี่ยวกับฟังก์ชัน* ตัวอย่างเช่น แอ็พพลิเคชันหรือระบบที่มีหน่วยความจำสูญเปล่ามีปัญหาเกี่ยวกับฟังก์ชัน

ในบางครั้ง ปัญหาเกี่ยวกับฟังก์ชันบางอย่างนำไปสู่ปัญหาประสิทธิภาพ ตัวอย่างเช่น เมื่อฟังก์ชันทำงานได้สำเร็จ แต่ความเร็วของฟังก์ชัน ต่ำ ในกรณีเหล่านี้ แทนที่จะปรับระบบ สิ่งที่สำคัญกว่าคือกำหนดสาเหตุหลักของ ปัญหาและแก้ไข อีกตัวอย่างหนึ่งอาจเป็น เมื่อ การสื่อสารช้าเนื่องจากเครือข่ายหรือเซิร์ฟเวอร์ชื่อ ดาวน์

คำอธิบายปัญหาประสิทธิภาพ

บุคลากรฝ่ายสนับสนุนมักได้รับรายงานปัญหาที่ระบุว่าบางคนมี ปัญหาเกี่ยวกับประสิทธิภาพบนระบบ และนำเสนอการวิเคราะห์ข้อมูลบางอย่าง ข้อมูลนี้ไม่เพียงพอสำหรับการกำหนดลักษณะของปัญหาประสิทธิภาพอย่าง ถูกต้อง ข้อมูลอาจบ่งชี้การใช้ประโยชน์ CPU 100 เปอร์เซ็นต์และคิวการรันยาว แต่อาจไม่มีข้อมูลใดเกี่ยวกับสาเหตุของปัญหาประสิทธิภาพ

ตัวอย่างเช่น ระบบอาจมีผู้ใช้ที่ล็อกอินจากรีโมตเทอร์มินัลบนเครือข่าย ที่ครอบคลุมหลายเราเตอร์ ผู้ใช้รายงานว่าระบบ ช้า ข้อมูลอาจบ่งชี้ว่า CPU มีการใช้ประโยชน์หนักมาก แต่ปัญหาที่แท้จริง อาจเป็นอักขระแสดงขึ้นบนเทอร์มินัลหลังจากล่าช้าไปนาน เนื่องจากแพ็กเก็ตสูญหายบนเครือข่าย (ซึ่งอาจเกิดจากเราเตอร์ที่ล้มเหลว หรือเครือข่ายที่โอเวอร์โหลด) สถานการณ์นี้อาจไม่มีอะไรต้องทำ กับการใช้ประโยชน์ CPU บนเครื่อง ถ้าในทางตรงกันข้าม ลูกค้าบ่นว่าชุดงานบนระบบใช้เวลารันนาน กรณีอาจเกี่ยวข้องกับการใช้ประโยชน์ CPU หรือ I/O แบนด์วิธ

พยายามให้ได้รับรายละเอียดมากที่สุดเท่าที่เป็นไปได้ก่อนที่คุณจะพยายามรวบรวม หรือวิเคราะห์ข้อมูล โดยการถามคำถามเกี่ยวกับปัญหาประสิทธิภาพ ดังต่อไปนี้:

- สามารถสาธิตปัญหาโดยการรันคำสั่งเฉพาะหรือการสร้างลำดับของเหตุการณ์ขึ้นใหม่ได้หรือไม่? (ตัวอย่างเช่น: **ls** `/slow/fs` หรือ **ping** *xxxxx*) ถ้าไม่ได้ ให้อธิบายตัวอย่างที่ซับซ้อนน้อยที่สุดของปัญหา
- ประสิทธิภาพช้าเป็นครั้งคราวหรือไม่? ระบบช้า แต่จากนั้นหายไปครู่หนึ่ง หรือไม่? เกิดขึ้นในบางเวลาของวันหรือเกิดขึ้นสัมพันธ์กับกิจกรรมเฉพาะ บางอย่างหรือไม่?
- ทุกอย่างช้าไปหมดหรือช้าเป็นบางอย่างเท่านั้น?
- ส่วนใดที่ช้า? ตัวอย่างเช่น เวลาในการกระจายอักขระ หรือเวลาที่ผ่านไป ในการทำให้ธุรกรรมเสร็จสมบูรณ์ หรือเวลาในการลงสีจอภาพ?
- ปัญหาเริ่มเกิดขึ้นเมื่อไร? สถานการณ์เป็นเหมือนเดิมตั้งแต่ ติดตั้งระบบในครั้งแรกหรือเริ่มต้นการใช้งาน? มีอะไรเปลี่ยนแปลง บนระบบก่อนที่ปัญหาเกิดขึ้น (เช่น การเพิ่มผู้ใช้เพิ่มเติม หรือการย้ายข้อมูลเพิ่มเติมไปในระบบ) หรือไม่?
- ถ้าไคลเอ็นต์/เซิร์ฟเวอร์ สามารถสาธิตปัญหาเมื่อรันเฉพาะแบบโลคัล บนเซิร์ฟเวอร์ (ปัญหาเครือข่ายและเซิร์ฟเวอร์) ได้หรือไม่?
- ถ้าเกี่ยวข้องกับเครือข่าย เซกเมนต์เครือข่ายมีการตั้งค่าคอนฟิก (รวมถึง แบนด์วิธ เช่น 10 Mb/วินาทีหรือ 9600 baud) อย่างไร? มีเราเตอร์ใดๆ ระหว่างไคลเอ็นต์และ เซิร์ฟเวอร์หรือไม่?
- แอ็พพลิเคชันผู้ขายใดกำลังรันอยู่บนระบบ และแอ็พพลิเคชันเหล่านั้น เกี่ยวข้องกับปัญหาประสิทธิภาพหรือไม่?
- ปัญหาประสิทธิภาพส่งผลกระทบต่อผู้ใช้อย่างไร?

การรายงานปัญหาประสิทธิภาพ

คุณควรรายงานปัญหาประสิทธิภาพของระบบปฏิบัติการให้ฝ่ายสนับสนุน IBM ทราบ ใช้ช่องทางการรายงาน ปัญหาซอฟต์แวร์ปกติของคุณ ถ้าคุณไม่คุ้นเคยกับช่องทางการรายงานปัญหาที่ถูกต้องของ องค์กรของคุณ ให้ตรวจสอบกับตัวแทน IBM ของคุณ

เครื่องมือการรวบรวมข้อมูล AIX Performance PMR (perfpmr) นับเป็นวิธีที่ดีที่สุดในการรวบรวมข้อมูลประสิทธิภาพ เมื่อสงสัยว่าเกิดปัญหาประสิทธิภาพ AIX เข้าถึงเครื่องมือเหล่านี้ผ่านทางเว็บที่ ftp://ftp.software.ibm.com/aix/tools/

perftools/perfpmr ปฏิบัติตามคำสั่งในไฟล์ README ในไดเร็กทอรีที่ตรงกับเวอร์ชัน AIX ซึ่งคุณจะประเมิน เพื่อให้ได้รับติดตั้ง และรวบรวมข้อมูลบนระบบของคุณ คำสั่งยัง แสดงวิธีการส่งข้อมูลไปยังฝ่ายสนับสนุน IBM สำหรับการวิเคราะห์หลังจากที่เปิด PMR แล้วด้วย

เมื่อบางคนรายงานปัญหาประสิทธิภาพ การเพียงแต่รวบรวมข้อมูลแล้ววิเคราะห์ไม่ใช่สิ่งที่เพียงพอ หากไม่ทราบ ธรรมชาติของปัญหาประสิทธิภาพ คุณอาจเสียเวลาอย่างมากในการวิเคราะห์ข้อมูล ซึ่งไม่เกี่ยวข้องกับปัญหาที่รายงาน

ก่อนที่คุณ จะรายงานปัญหาให้เจ้าหน้าที่ฝ่ายสนับสนุนทราบ ควรจัดเตรียมข้อมูลที่คุณจะถูก ถามซึ่งจะช่วยในการสืบสวนปัญหาไว้ ล่วงหน้า เจ้าหน้าที่ฝ่ายสนับสนุนในท้องถิ่นของคุณจะพยายามแก้ไขปัญหาประสิทธิภาพ ร่วมกับคุณโดยตรงอย่างรวดเร็ว

คุณสามารถช่วยให้การแก้ไขปัญหา เร็วขึ้นได้ในสามวิธีดังนี้:

1. ส่งข้อความที่เป็นลายลักษณ์อักษรอย่างชัดเจนเกี่ยวกับอินสแตนซ์เฉพาะแบบง่าย ของปัญหา แต่ต้องแน่ใจว่าแยกอาการและข้อเท็จจริงออกจากทฤษฎี แนวคิด และข้อสรุปของคุณเอง PMRs ที่รายงานว่า "ระบบช้า" ต้องใช้ การสืบสวนหลายด้านเพื่อกำหนดว่าคำว่าช้าของคุณหมายความว่าอะไร มีการประเมิน อย่างไร และอะไรคือประสิทธิภาพที่ยอมรับได้
2. ให้ข้อมูลเกี่ยวกับทุกสิ่งที่มีการเปลี่ยนแปลงบนระบบ ในช่วงหลายสัปดาห์ก่อนเกิดปัญหา การขาดข้อมูลการเปลี่ยนแปลงบางอย่างอาจเป็นอุปสรรคต่อการ สืบสวน และจะส่งผลให้การค้นหาวิธีแก้ไขล่าช้าออกไป ถ้ามี ข้อเท็จจริงครบถ้วนสมบูรณ์ ทีมงานฝ่ายสนับสนุนสามารถตัดสาเหตุที่ไม่เกี่ยวข้อง ออกได้อย่างรวดเร็ว
3. ใช้เครื่องที่ถูกต้องในการให้ข้อมูล ในไซต์ขนาดใหญ่มาก อาจเกิดการรวบรวมข้อมูลบนเครื่องที่ไม่ถูกต้องโดยไม่ได้ตั้งใจได้ง่าย ซึ่งทำให้ ยากมากต่อการสืบสวนปัญหา

เมื่อคุณรายงานปัญหา ให้แจ้งข้อมูลพื้นฐานดังต่อไปนี้:

- รายละเอียดปัญหาซึ่งสามารถใช้เพื่อค้นหาฐานข้อมูลประวัติปัญหา เพื่อดูว่าเคยมีการรายงานปัญหาที่คล้ายกันมาก่อนแล้วหรือไม่
- แง่มุมใดของการวิเคราะห์ของคุณที่ทำให้คุณสรุปว่าปัญหาเกิดขึ้นจาก ข้อบกพร่องในระบบปฏิบัติการ?
- การตั้งค่าคอนฟิกฮาร์ดแวร์และซอฟต์แวร์ซึ่งมีปัญหาเกิดขึ้นเป็น อย่างไร?
  - ปัญหาเกิดขึ้นกับระบบเดียว หรือส่งผลกระทบต่อหลาย ระบบ?
  - รุ่น ขนาดหน่วยความจำ ตลอดจนจำนวนและขนาดของดิสก์บนระบบ ที่ได้รับผลกระทบเป็นอย่างไร?
  - LAN และสื่อการสื่อสารอื่นชนิดใดที่เชื่อมต่ออยู่กับ ระบบ?
  - การตั้งค่าคอนฟิกโดยรวมรวมการตั้งค่าคอนฟิกสำหรับระบบปฏิบัติการอื่นด้วยหรือไม่?
- ลักษณะของโปรแกรมหรือเวิร์กโหลดที่ประสบปัญหา เป็นอย่างไร?
  - การวิเคราะห์ด้วยคำสั่ง **time, iostat,** และ **vmstat** บ่งชี้ว่ามีข้อจำกัด CPU หรือ I/O หรือไม่?
  - มีเวิร์กโหลดกำลังรันอยู่บนระบบที่ได้รับผลกระทบหรือไม่: เวิร์กสเตชัน เซิร์ฟเวอร์ หลายผู้ใช้ หรือชุดดังกล่าว?
- วัตถุประสงค์ด้านประสิทธิภาพที่ไม่สามารถปฏิบัติตามได้คืออะไร?
  - วัตถุประสงค์หลักคือในแง่ของเวลาการตอบกลับของคอนโซลหรือเทอร์มินัล ผลผลิต หรือการตอบกลับแบบเรียลไทม์?
  - การประเมินวัตถุประสงค์ดำเนินการบนระบบอื่นหรือไม่? ถ้าใช่ ระบบนั้นมีการตั้งค่าคอนฟิกอย่างไร?

ถ้านี่เป็นการรายงานปัญหาครั้งแรก คุณจะได้รับหมายเลข PMR สำหรับใช้ในการระบุข้อมูลเพิ่มเติมใดๆ ที่คุณให้และสำหรับการอ้างอิง ในอนาคต

รวมรายการทั้งหมดดังต่อไปนี้เมื่อรวบรวมข้อมูลสนับสนุน และข้อมูล **perfpmr** สำหรับ PMR ในครั้งแรก:

- สื่อของการทำซ้ำปัญหา
  - ถ้าเป็นไปได้ ควรจะรวมโปรแกรมหรือ shell script ที่แสดง ปัญหา
  - อย่างน้อยที่สุดต้องมีคำอธิบายโดยละเอียดของสภาพซึ่งปัญหา เกิดขึ้น
- แอ็พพลิเคชันเกิดปัญหา:
  - ถ้าแอ็พพลิเคชันเป็นหรือพึ่งพาผลิตภัณฑ์ซอฟต์แวร์ใดๆ ควรจะระบุ เวอร์ชันและรีลีสที่ถูกต้องของผลิตภัณฑ์นั้น
  - ถ้าไม่สามารถรีลีสซอร์สโค้ดของแอ็พพลิเคชันที่ผู้ใช้เขียนขึ้นได้ ควรระบุ ชุดที่ถูกต้องของคอมไพเลอร์พารามิเตอร์ที่ใช้ในการสร้างโปรแกรม ที่ดำเนินการได้

การมอนิเตอร์และปรับคำสั่งและรูทีนย่อย

ระบบได้จัดเตรียมคำสั่งที่เกี่ยวข้องกับผลการทำงาน และรูทีนย่อย

ทูลด้านประสิทธิภาพการทำงานสำหรับสภาวะแวดล้อมระบบที่ตกอยู่ในสองหมวดหมู่ทั่วๆ ไปคือ: หมวดหมู่ที่บอกให้คุณทราบถึงสิ่งที่เกิดขึ้น และหมวดหมู่ที่อนุญาตให้คุณทำบางสิ่ง เกี่ยวกับทูลนั้น ทูลบางตัวจะทำงานทั้งสองแบบ สำหรับรายละเอียดของไวยากรณ์และฟังก์ชันของคำสั่ง โปรดดู *Commands Reference*

คำสั่งที่เกี่ยวข้องกับผลการทำงานคือแพ็กเกจที่เป็นส่วนหนึ่งของชุดไฟล์ `perfagent.tools`, `bos.acct`, `bos.sysmgt.trace`, `bos.adt.samples`, `bos.perf.tools` และ `bos.perf.tune` ซึ่งจัดส่งมาพร้อมกับ ระบบปฏิบัติการพื้นฐาน

คุณสามารถกำหนดได้ว่า ทูลด้านประสิทธิภาพการทำงานทั้งหมดได้ถูกติดตั้งไว้โดยรันหนึ่งในคำสั่งต่อไปนี้:

```
# lslpp -lI perfagent.tools bos.sysmgt.trace bos.acct bos.perf.tools bos.perf.tune
```

คำสั่งการรายงานและการวิเคราะห์ประสิทธิภาพ

คำสั่งการรายงานและการวิเคราะห์ประสิทธิภาพให้ข้อมูลเกี่ยวกับประสิทธิภาพ ของส่วนของระบบหนึ่งส่วนขึ้นไป หรือข้อมูลพารามิเตอร์หนึ่งตัวขึ้นไปที่มีผลต่อ ประสิทธิภาพ

คำสั่งมีดังนี้:

คำสั่ง **ฟังก์ชัน**

alstat รายงานจำนวนข้อยกเว้นการจัดวาง

atmstat แสดงสถิติอะแด็ปเตอร์ Asynchronous Transfer Mode (ATM)

curt รายงานการใช้ประโยชน์ CPU สำหรับแต่ละเคอร์เนลเธรด

emstat รายงานจำนวนการสร้างการเลียนแบบ

entstat แสดงสถิติไดรเวอร์อุปกรณ์อีเทอร์เน็ตและอุปกรณ์

fddistat
แสดงสถิติไดรเวอร์อุปกรณ์ FDDI และอุปกรณ์

filemon ใช้ฟังก์ชันการติดตามเพื่อรายงานกิจกรรม I/O ของฟิสิคัลวอลุ่ม โลจิคัลวอลุ่ม แต่ละไฟล์ และผู้จัดการหน่วยความจำเสมือน

fileplace
แสดงการวางแบบฟิสิคัลหรือแบบโลจิคัลของบล็อกที่ประกอบขึ้นเป็นไฟล์ ภายในฟิสิคัลหรือโลจิคัลวอลุ่มซึ่งบล็อกตั้งอยู่

gprof รายงานลำดับของการควบคุมระหว่างรูทีนย่อยต่างๆ ของโปรแกรม และ จำนวนเวลา CPU ที่ใช้โดยแต่ละรูทีนย่อย

ifconfig ตั้งค่าคอนฟิกหรือแสดงพารามิเตอร์อินเตอร์เฟสเครือข่ายสำหรับเครือข่ายที่ใช้ TCP/IP

ioo ตั้งค่าพารามิเตอร์การปรับที่เกี่ยวข้องกับ I/O (พร้อมกับ **vmo** แทนที่คำสั่ง **vmtune**)

iostat แสดงข้อมูลการใช้ประโยชน์สำหรับ:

- เทอร์มินัล
- CPU
- ดิสก์
- Adapters

ipfilter แยกส่วนหัวการดำเนินงานอื่นจากไฟล์เอาต์พุต **ipreport** และแสดงข้อมูลนั้นในตาราง

ipreport
สร้างรายงานการติดตามแพ็กเก็ตจากไฟล์การติดตามแพ็กเก็ตที่ระบุ

iptrace แสดงการติดตามแพ็กเก็ตระดับอินเตอร์เฟสสำหรับอินเทอร์เน็ตโปรโตคอล

locktrace
เปิดการติดตามล็อค

lsattr แสดงแอ็ตทริบิวต์ของระบบที่มีผลกระทบต่อประสิทธิภาพ เช่น:

- ความเร็วของตัวประมวลผล
- ขนาดของแคช
- ขนาดของหน่วยความจำจริง
- จำนวนหน้าสูงสุดในบัฟเฟอร์แคชของบล็อก I/O
- จำนวนกิโลไบต์สูงสุดของหน่วยความจำที่ใช้ได้สำหรับ mbufs
- เครื่องหมาย high- และ low-water สำหรับการเพจดิสก์-I/O

lsdev แสดงอุปกรณ์ในระบบและลักษณะของอุปกรณ์นั้น

lslv แสดงข้อมูลเกี่ยวกับโลจิคัลวอลุ่ม

lsps แสดงลักษณะของพื้นที่ว่างการเพจ

lspv แสดงข้อมูลเกี่ยวกับฟิสิคัลวอลุ่มภายในกลุ่มวอลุ่ม

lsvg แสดงข้อมูลเกี่ยวกับกลุ่มวอลุ่ม

mtrace พิมพ์ multicast พาธจากซอร์สไปยังเครื่องรับ

netpmon
ใช้ฟังก์ชันการติดตามเพื่อรายงานกิจกรรมเครือข่าย รวมถึง:

- การใช้ CPU
- อัตราข้อมูล

- เวลาตอบกลับ

netstat แสดงข้อมูลการตั้งค่าคอนฟิกหลายอย่างและสถิติเกี่ยวกับกิจกรรม การสื่อสาร เช่น:

- สถานะปัจจุบันของพูล mbuf
- ตารางการเราต์
- สถิติรวมเกี่ยวกับกิจกรรมเครือข่าย

nfso แสดง (หรือเปลี่ยน) ค่าของอ็อพชัน NFS

nfsstat แสดงสถิติเกี่ยวกับกิจกรรมเซิร์ฟเวอร์และไคลเอ็นต์ Network File System (NFS) และ Remote Procedure Call (RPC)

no แสดง (หรือเปลี่ยน) ค่าของอ็อพชันเครือข่าย เช่น:

- ขนาดของซ็อกเก็ตบัฟเฟอร์การส่งและการรับดีฟอลต์
- จำนวนหน่วยความจำรวมสูงสุดที่ใช้ในพูล mbuf และคลัสเตอร์

pdt_config
เริ่มต้น หยุด หรือเปลี่ยนพารามิเตอร์ของเครื่องมือการวิเคราะห์ ประสิทธิภาพ

pdt_report
สร้างรายงาน PDT ตามข้อมูลประวัติปัจจุบัน

pprof รายงานการใช้ CPU ของเคอร์นัลเธรดทั้งหมดในรอบระยะเวลาหนึ่ง

prof แสดงข้อมูลโปรไฟล์ของไฟล์อ็อบเจ็กต์

ps แสดงสถิติและข้อมูลสถานะเกี่ยวกับกระบวนการในระบบ เช่น:

- ID กระบวนการ
- กิจกรรม I/O
- การใช้ประโยชน์ CPU

sar แสดงสถิติเกี่ยวกับกิจกรรมระบบปฏิบัติการ เช่น:

- การเข้าถึงไดเร็กทอรี
- การเรียกระบบการอ่านและการบันทึก
- Forks และ execs
- กิจกรรมการเพจ

schedo ตั้งค่าพารามิเตอร์การปรับสำหรับตัวจัดตารางเวลา CPU (แทนที่คำสั่ง **schedtune**)

smitty แสดง (หรือเปลี่ยน) พารามิเตอร์การจัดการระบบ

splat ล็อคเครื่องมือการวิเคราะห์การชว่งชิง

svmon รายงานสถานะของหน่วยความจำที่ระดับระบบ กระบวนการ และระดับเซกเมนต์

tcpdump
พิมพ์ส่วนหัวแพ็กเก็ต

time, timex
พิมพ์เวลาที่ผ่านไปและเวลา CPU ที่ใช้โดยการดำเนินการของคำสั่ง

topas รายงานสถิติระบบโลคัลที่เลือก

tokstat แสดงสถิติไดรเวอร์อุปกรณ์ Token-Ring และอุปกรณ์

tprof ใช้ฟังก์ชันการติดตามเพื่อรายงานการใช้ CPU ของการบริการเคอร์นัล รูทีนย่อยไลบรารีโมดูลแอ็พพลิเคชัน โปรแกรม และแต่ละบรรทัดของ ซอร์สโค้ดในแอ็พพลิเคชันโปรแกรม

trace, trcrpt
บันทึกไฟล์ที่บันทึกลำดับที่แน่นอนของกิจกรรมภายใน ระบบ

traceroute
พิมพ์เราต์ที่ IP แพ็กเก็ตใช้เดินทางไปยังโฮสต์เครือข่าย

vmo ตั้งค่าพารามิเตอร์การปรับที่เกี่ยวข้องกับ VMM (พร้อมกับ **ioo** แทนที่คำสั่ง **vmtune**)

vmstat แสดงข้อมูล VMM เช่น:

- จำนวนกระบวนการที่จัดส่งได้หรือกำลังรออยู่
- ขนาดเฟรมหน้า รายการที่ว่าง
- กิจกรรม page-fault
- การใช้ประโยชน์ CPU

คำสั่งการปรับประสิทธิภาพ

AIX สนับสนุน คำสั่งหลายรายการที่ช่วยให้คุณสามารถเปลี่ยนส่วนที่เกี่ยวข้องกับประสิทธิภาพ ของระบบหนึ่งส่วนขึ้นไป

คำสั่ง ฟังก์ชัน

bindprocessor
ยึดหรือยกเลิกการยึดเคอร์นัลเธรดของกระบวนการไว้กับตัวประมวลผล

caccelstat
รายงานสถิติที่เกี่ยวข้องกับตัวเร่งที่สอดคล้องกันทั้งระบบ หรือสำหรับตัวเร่งและกระบวนการแต่ละส่วน

chdev เปลี่ยนลักษณะของอุปกรณ์

chlv เปลี่ยนเฉพาะลักษณะของโลจิคัลวอลุ่ม

chps เปลี่ยนแอ็ตทริบิวต์ของพื้นที่ว่างการเพจ

fdpr ยูทิลิตีการปรับประสิทธิภาพสำหรับการพัฒนาเวลาการดำเนินการและการใช้ ประโยชน์หน่วยความจำจริงของแอ็พพลิเคชันโปรแกรมระดับผู้ใช้

ifconfig ตั้งค่าคอนฟิกหรือแสดงพารามิเตอร์อินเตอร์เฟสเครือข่ายสำหรับเครือข่ายที่ใช้ TCP/IP

ioo ตั้งค่าพารามิเตอร์การปรับที่เกี่ยวข้องกับ I/O (พร้อมกับ **vmo** แทนที่คำสั่ง **vmtune**)

migratepv
ย้ายฟิสิคัลพาร์ติชันที่จัดสรรจากฟิสิคัลวอลุ่มหนึ่งไปยังฟิสิคัลวอลุ่มอื่น หนึ่งวอลุ่มขึ้นไป

mkps เพิ่มพื้นที่ว่างการเพจเพิ่มเติมลงในระบบ

nfso ตั้งค่าคอนฟิกตัวแปรเครือข่าย Network File System (NFS)

nice รันคำสั่งที่ระดับความสำคัญต่ำกว่าหรือสูงกว่า

no ตั้งค่าคอนฟิกแอ็ตทริบิวต์เครือข่าย

renice เปลี่ยนค่า nice ของกระบวนการที่กำลังรันอยู่

reorgvg
จัดระเบียบการจัดสรรฟิสิคัลพาร์ติชันใหม่สำหรับกลุ่มวอลุ่ม

rmss จำลองระบบที่มีขนาดหน่วยความจำต่างๆ สำหรับการทดสอบประสิทธิภาพ ของแอ็พพลิเคชัน

schedo ตั้งค่าพารามิเตอร์การปรับสำหรับตัวจัดตารางเวลา CPU (แทนที่คำสั่งเริ่มต้น **schedtune**)

smitty เปลี่ยน (หรือแสดง) พารามิเตอร์การจัดการระบบ

tuncheck
ตรวจสอบไฟล์ stanza ที่มีค่าพารามิเตอร์การปรับ

tundefault
รีเซ็ตพารามิเตอร์ที่ปรับได้ทั้งหมดเป็นค่าดีฟอลต์

tunrestore
เรียกคืนค่าพารามิเตอร์ที่ปรับได้ทั้งหมดจากไฟล์ stanza

tunsave
บันทึกค่าพารามิเตอร์ที่ปรับได้ทั้งหมดในไฟล์ stanza

vmo ตั้งค่าพารามิเตอร์การปรับที่เกี่ยวข้องกับ VMM (พร้อมกับ **ioo** แทนที่คำสั่ง **vmtune**)

ข้อมูลที่เกี่ยวข้อง:

คำสั่ง caccelstat

รูทีนย่อยที่เกี่ยวข้องกับประสิทธิภาพ

AIX สนับสนุนรูทีนย่อยหลายรายการ ซึ่งสามารถใช้ในการมอนิเตอร์และปรับประสิทธิภาพได้

bindprocessor( )
ยึดเคอร์นัลเธรดไว้กับตัวประมวลผล

getpri( )
กำหนดระดับความสำคัญการจัดตารางเวลาของกระบวนการที่กำลังรันอยู่

getpriority( )
กำหนดค่า nice ของกระบวนการที่กำลังรันอยู่

getrusage( )
ดึงข้อมูลเกี่ยวกับการใช้รีซอร์สระบบ

nice( ) เพิ่มค่า nice ของกระบวนการปัจจุบัน

psdanger( )
ดึงข้อมูลเกี่ยวกับการใช้พื้นที่ว่างการเพจ

setpri( )
เปลี่ยนระดับความสำคัญของกระบวนการที่กำลังรันเป็นระดับความสำคัญคงที่

setpriority( )
ตั้งค่า nice ของกระบวนการที่กำลังรันอยู่

การใช้ของคำสั่ง ld อย่างมีประสิทธิภาพ

ตัวโยง (เรียกใช้งานเป็นขั้นตอนสุดท้ายของการคอมไพล์ หรือเรียกใช้คำสั่ง **ld** โดยตรง) มีฟังก์ชันที่ไม่ได้พบในตัวเชื่อมโยง UNIX ปกติ

สถานการณ์นี้อาจส่งผลทำให้การลิงก์ยาวนานขึ้น หากกำลังของตัวโยงระบบปฏิบัติการเพิ่มเติม ไม่ได้ใช้ประโยชน์ ส่วนนี้อธิบายถึงเทคนิคบางอย่าง สำหรับการใช้ตัวโยงอย่างมีประสิทธิภาพ

ตัวอย่าง

ต่อไปนี้คือ ตัวอย่างที่แสดงให้เห็นภาพของการใช้คำสั่ง **ld** อย่างมีประสิทธิผล:

1. หากต้องการโยงไลบรารีล่วงหน้า ให้ใช้คำสั่งต่อไปนี้บนไฟล์เก็บถาวร:

   ```
   # ld -r libfoo.a -o libfooa.o
   ```

2. การคอมไฟล์และการโยงโปรแกรม FORTRAN `something.f` จะเป็นดังนี้:

   ```
   # xlf something.f libfooa.o
   ```

   หมายเหตุ ไลบรารีที่โยงไว้ก่อนหน้าจะถูกนำมาใช้เป็นไฟล์อินพุตปกติ ไม่ได้ใช้กับไวยากรณ์ identification ไลบรารี (**-lfoo**)

3. หากต้องการคอมไพล์โมดูลใหม่และโยงโปรแกรมเรียกทำงานใหม่ หลังจากที่แก้ไขจุดบกพร่องแล้ว ให้ใช้คำสั่งต่อไปนี้:

   ```
   # xlf something.f a.out
   ```

4. อย่างไรก็ตาม ถ้าการแก้ไขจุดบกพร่องมีผลทำให้การเรียกรูทีนที่แตกต่างกันในไลบรารี การโยงจะล้มเหลว สคริปต์ Korn shell ต่อไปนี้จะทดสอบโค้ดสำหรับส่งคืนความล้มเหลว และเรียกคืน:

   ```
   # !/usr/bin/ksh
   # Shell script for source file replacement bind
   #
   xlf something.f a.out
   rc=$?
   if [ "$rc" != 0 ]
   then
   echo "New function added ... using libfooa.o"
   xlf something.o libfooa.o
   fi
   ```

โปรแกรมที่ดำเนินการได้ซึ่งยึดใหม่ได้

ส่วนเอกสารที่เป็นทางการของ binder อ้างอิงถึงความสามารถของ binder ในการใช้โปรแกรมที่ดำเนินการได้ (โหลดโมดูล) เป็นอินพุต

การใช้ฟังก์ชันนี้สามารถพัฒนาประสิทธิภาพโดยรวมของระบบได้เป็นอย่างมาก ในด้านเวิร์กโหลดการพัฒนาซอฟต์แวร์ตลอดจนเวลาตอบกลับของ แต่ละคำสั่ง **ld**

ในระบบ UNIX ปกติส่วนใหญ่ คำสั่ง **ld** ใช้เป็นอินพุตชุดของไฟล์ที่มีอ็อบเจ็กต์โค้ด จากแต่ละไฟล์ `.o` หรือจากไลบรารีที่ เก็บถาวรของไฟล์ `.o` อย่างใดอย่างหนึ่งเสมอ จากนั้น คำสั่ง `ld` แก้ไขการอ้างอิงภายนอกระหว่างไฟล์เหล่านี้ และเขียนโปรแกรมที่ดำเนินการได้ ด้วยชื่อดีฟอลต์ `a.out` ไฟล์ `a.out` สามารถดำเนินการได้เท่านั้น หากพบบักในโมดูลใดโมดูลหนึ่งที่รวมอยู่ใน ไฟล์ `a.out` ซอร์สโค้ดที่บกพร่องจะถูกเปลี่ยนแปลง และรีคอมไพล์ จากนั้น ต้องทำซ้ำกระบวนการ **ld** ทั้งหมด โดยเริ่มต้นจากชุดทั้งหมดของไฟล์ `.o`

อย่างไรก็ตาม ในระบบปฏิบัติการนี้ binder สามารถยอมรับทั้งไฟล์ `.o` และ `a.out` เป็นอินพุต เนื่องจาก binder มีข้อมูล External Symbol Dictionary (ESD) และ Relocation Dictionary (RLD) ที่แก้ไขแล้วอยู่ในไฟล์ที่ดำเนินการได้ ซึ่งหมายความว่าผู้ใช้สามารถยึดโปรแกรมที่ดำเนินการได้ที่มีอยู่ใหม่ เพื่อเปลี่ยนไฟล์ `.o` ที่แก้ไขไฟล์เดียว แทนการสร้าง โปรแกรมที่ดำเนินการได้ใหม่ตั้งแต่เริ่มต้น เนื่องจากกระบวนการยึด ใช้หน่วยเก็บและรอบตัวประมวลผลอย่างเป็นสัดส่วนกับจำนวน ของไฟล์ต่างๆ ที่กำลังเข้าถึงและจำนวนของการอ้างอิงต่างๆ ถึงสัญลักษณ์ที่ต้องแก้ไข การยึดโปรแกรมที่ดำเนินการได้ใหม่เข้ากับเวอร์ชันใหม่ของโมดูลจึงเร็วกว่าการยึด จาก scratch มาก

ไลบรารีรูทีนย่อยที่ยึดไว้แล้ว

สิ่งสำคัญไม่ยิ่งหย่อนไปกว่ากันในบางสภาพแวดล้อมคือความสามารถในการ ยึดไลบรารีรูทีนย่อยทั้งไลบรารีไว้ล่วงหน้าก่อนการใช้งาน

ไลบรารีรูทีนย่อยของระบบ เช่น `libc.a` มีผลบังคับใช้ และจัดส่งมาในรูปแบบ binder-output แทนที่จะเป็นไฟล์เก็บถาวรของไฟล์ `.o` ไลบรารีนี้ช่วยประหยัดเวลาการประมวลผลของผู้ใช้ได้เป็นอย่างมาก เมื่อยึดแอ็พพลิเคชันเข้ากับไลบรารีระบบที่ต้องการ เนื่องจากต้องแก้ไขเฉพาะ ข้อมูลอ้างอิงจากแอ็พพลิเคชันไปยังรูทีนย่อยในไลบรารี เท่านั้น ข้อมูลอ้างอิงระหว่างรูทีนไลบรารีระบบเองมีการแก้ไขแล้ว ในระหว่างกระบวนการสร้างระบบ

อย่างไรก็ตาม ไลบรารีรูทีนย่อยของบริษัทอื่นจำนวนมากมีการจัดส่งแบบรูทีน ในรูปแบบไฟล์เก็บถาวรเป็นไฟล์ raw `.o` เมื่อผู้ใช้ยึด แอ็พพลิเคชันเข้ากับไลบรารีดังกล่าว ผู้ยึดต้องแก้ไขสัญลักษณ์ของทั้งไลบรารี ในแต่ละครั้งที่ยึดแอ็พพลิเคชัน การทำเช่นนี้ส่งผลให้ใช้เวลายึดนาน ในสภาพแวดล้อมที่แอ็พพลิเคชันกำลังยึดเข้ากับไลบรารีขนาดใหญ่ บนเครื่องขนาดเล็ก

ประสิทธิภาพของไลบรารีที่ยึดและที่ไม่ยึดมีความแตกต่างกันมาก โดยเฉพาะในการตั้งค่าคอนฟิกต่ำสุด

การเข้าถึงตัวจับเวลาโพรเซสเซอร์

บ่อยครั้งที่ความพยายามในการวัดช่วงเวลาที่เล็กมากจะไม่ประสบผลสำเร็จ โดยกิจกรรมพื้นหลังที่ไม่ต่อเนื่องซึ่งเป็นส่วนหนึ่งของระบบปฏิบัติการ และโดยเวลาของการประมวลผลที่ใช้โดยรูทีนเวลาระบบ หนึ่งวิธีที่แก้ปัญหานี้คือ การเข้าถึงตัวจับเวลาโพรเซสเซอร์โดยตรง เพื่อกำหนดเวลาเริ่มต้นและเวลาสิ้นสุดของช่วงเวลาการวัด รันการวัดซ้ำๆ จากนั้นกรองผลลัพธ์เพื่อลบระยะเวลา เมื่ออินเตอร์รัปต์เกิดขึ้น

สถาปัตยกรรม POWER และ POWER2 ใช้ ตัวจับเวลาตัวประมวลผลเป็นคู่ของรีจิสเตอร์วัตถุประสงค์พิเศษ สถาปัตยกรรม POWER processor-based จะกำหนดเรจิสเตอร์ขนาด 64 บิตที่เรียกว่า *TimeBase* เฉพาะภาษาแอสเซมบลีเท่านั้น ที่สามารถเข้าถึงเรจิสเตอร์เหล่านี้ได้

หมายเหตุ: เวลาที่วัดได้โดยตัวจับเวลาโพรเซสเซอร์คือ เวลาของนาฬิกาติดผนังที่แน่นอน ถ้าอินเตอร์รัปต์เกิดขึ้นระหว่างการเข้าถึงตัวจับเวลา ช่วงเวลาที่คำนวณได้จะสอดแทรกการประมวลผลของอินเตอร์รัปต์ และการประมวลผลอื่นๆ ที่เป็นไปได้ซึ่งถูกจัดส่งก่อนที่การควบคุมจะส่งคืนไปยัง โค้ดที่กำลังจับเวลา เวลาจากตัวจับเวลาโพรเซสเซอร์คือ เวลาดิบ และไม่ควรใช้ในสถานการณ์ที่ไม่ได้รับการทดสอบ อย่างสมเหตุสมผล

กลุ่มของไลบรารีซับรูทีนสามส่วนจะสร้างการเข้าถึงเรจิสเตอร์ TimeBase ที่เป็นสถาปัตยกรรมที่เป็นอิสระ รูทีนย่อยมีดังต่อไปนี้:

read_real_time( )
รูทีนย่อยนี้จะขอรับเวลาปัจจุบันจากแหล่งที่มาที่เหมาะสม และเก็บไว้เป็นค่าสองค่าแบบ 32 บิต

read_wall_time( )
รูทีนย่อยนี้ขอรับค่าเรจิสเตอร์ *TimeBase* ที่เป็นค่าดิบจากแหล่งที่มาที่เหมาะสม และเก็บไว้เป็นค่าสองค่าแบบ 32 บิต

time_base_to_time( )
รูทีนย่อยนี้ทำให้มั่นใจได้ว่า ค่าเวลาอยู่ในหน่วยวินาทีและนาโนวินาที ซึ่งดำเนินการแปลงใดๆ จากรูปแบบ TimeBase

ฟังก์ชันการรับเวลาและการแปลงเวลาจะแยกจากกันเพื่อลดการใช้เวลา ที่ได้รับมาให้น้อยที่สุด

ตัวอย่างต่อไปนี้จะแสดงวิธีที่รูทีนย่อย **read_real_time( )** และ **time_base_to_time( )** สามารถนำมาใช้เพื่อวัดเวลาที่ใช้ไปสำหรับส่วนของโค้ดที่ระบุเฉพาะ:

```
#include <stdio.h>
#include <sys/time.h>

int main(void) {
   timebasestruct_t start, finish;
   int val = 3;
   int w1, w2;
   double time;

   /* get the time before the operation begins */
   read_real_time(&start, TIMEBASE_SZ);

   /* begin code to be timed */
   printf("This is a sample line %d \n", val);
   /* end code to be timed   */

   /* get the time after the operation is complete
   read_real_time(&finish, TIMEBASE_SZ);

   /* call the conversion routines unconditionally, to ensure    */
   /* that both values are in seconds and nanoseconds regardless */
   /* of the hardware platform. */
   time_base_to_time(&start, TIMEBASE_SZ);
   time_base_to_time(&finish, TIMEBASE_SZ);

   /* subtract the starting time from the ending time */
   w1 = finish.tb_high - start.tb_high; /* probably zero */
   w2 = finish.tb_low - start.tb_low;

   /* if there was a carry from low-order to high-order during  */
   /* the measurement, we may have to undo it. */
   if (w2 < 0) {
      w1--;
      w2 += 1000000000;
   }

   /* convert the net elapsed time to floating point microseconds */
```

```
   time = ((double) w2)/1000.0;
   if (w1 > 0)
      time += ((double) w1)*1000000.0;

   printf("Time was %9.3f microseconds \n", time);
   exit(0);
}
```

หากต้องการลบการใช้งานของการเรียกและส่งคืนจากรูทีนตัวจับเวลา คุณสามารถทดสอบได้ด้วยการโยงเบนช์มาร์กที่ไม่ได้แบ่งใช้ (โปรดดู “ใช้การลิงก์แบบไดนามิกและการลิงก์แบบสแตติกเมื่อใด” ในหน้า 421)

ถ้านี่คือเบนช์มาร์กของผลการทำงานที่เกิดขึ้นจริง โค้ดจะถูกวัดแบบซ้ำๆ จำนวนของการทำซ้ำอย่างต่อเนื่องจะสะสมเวลาไว้ เวลาเฉลี่ยสำหรับการดำเนินการจะถูกคำนวณ แต่อาจสอดแทรกตัวจัดการอินเตอร์รัปต์ หรือกิจกรรมภายนอกอื่น ถ้าจำนวนของการทำซ้ำคือเวลาที่วัดไว้แต่ละครั้ง เวลานั้นจะสามารถตรวจสอบได้อย่างเป็นเหตุเป็นผล แต่การใช้งานของรูทีนการจับเวลาจะถูกสอดแทรกอยู่ในการวัดแต่ละครั้ง ซึ่งอาจต้องการใช้ทั้งเทคนิคและการเปรียบเทียบผลลัพธ์ ในกรณีใดๆ คุณอาจต้องการพิจารณาวัตถุประสงค์ของการวัดในการเลือกเมธอด

การเข้าถึงไทม์เมอร์ POWER-based-architecture-unique

สถาปัตยกรรมตัวประมวลผล POWER family และ POWER2 มีทะเบียนวัตถุประสงค์พิเศษสองรายการ (ทะเบียนด้านบนและทะเบียนด้านล่าง) ที่มีไทม์เมอร์ความละเอียดสูง

หมายเหตุ: คำอธิบายต่อไปนี้ใช้เฉพาะกับสถาปัตยกรรม POWER family และ POWER2 (และ 601 processor chip) เท่านั้น ตัวอย่างโค้ดจะทำงานอย่างถูกต้องในระบบที่ใช้ POWER แต่คำสั่งบางอย่างจะถูกจำลองขึ้น เนื่องจากวัตถุประสงค์ของการเข้าถึง ไทม์เมอร์ตัวประมวลผลคือเพื่อให้ได้ค่าเวลาที่มีความแม่นยำสูงโดยมีโอเวอร์เฮด ต่ำ การจำลองทำให้ผลลัพธ์มีประโยชน์น้อยลงมาก

ทะเบียนด้านบนของ สถาปัตยกรรมตัวประมวลผล POWER family และ POWER2 มีเวลาในหน่วยวินาที และทะเบียนด้านล่างมีการนับเศษส่วนวินาที ในหน่วยนาโนวินาที ความแม่นยำที่แท้จริงของเวลาในทะเบียนด้านล่าง ขึ้นอยู่กับความถี่ในการอัพเดต ซึ่งเป็นคุณลักษณะเฉพาะโมเดล

รูทีนแอสเซมเบลอร์เพื่อเข้าถึงเรจิเตอร์ตัวจับเวลา POWER processor-based

โมดูลภาษาแอสเซมเบลอร์ (`timer.s`) จะมีรูทีน (`rtc_upper` และ `rtc_lower`) เพื่อเข้าถึงเรจิสเตอร์ส่วนบนและส่วนล่างของตัวจับเวลา

```
          .globl  .rtc_upper
.rtc_upper: mfspr   3,4         # copy RTCU to return register
            br

          .globl  .rtc_lower
.rtc_lower: mfspr   3,5         # copy RTCL to return register
            br
```

รูทีนย่อย C เพื่อจัดหาเวลาในหน่วยวินาที

โมดูล `second.c` มีรูทีน C ที่เรียกรูทีน `timer.s` เพื่อเข้าถึงเนื้อหาเรจิสเตอร์ส่วนบนและส่วนล่าง

รูทีนจะส่งคืนค่าความเที่ยงตรงสองเท่าของเวลาในหน่วยวินาที

```
double second()
{
  int ts, tl, tu;

  ts = rtc_upper();     /* seconds                          */
  tl = rtc_lower();     /* nanoseconds                      */
  tu = rtc_upper();     /* Check for a carry from           */
  if (ts != tu)         /* the lower reg to the upper. */
    tl = rtc_lower();   /* Recover from the race condition. */
  return ( tu + (double)tl/1000000000 );
}
```

รูทีนย่อย **second( )** สามารถเรียกได้จากรูทีน C หรือรูทีน FORTRAN อย่างใดอย่างหนึ่ง

หมายเหตุ: ขึ้นอยู่กับช่วงเวลา ตั้งแต่การรีเซ็ตระบบล่าสุด โมดูล `second.c` อาจได้รับผลตอบแทนจาการผันแปรของจำนวนของความแม่นยำ เวลาที่ยาวนานตั้งแต่การรีเซ็ต จำนวนของบิตที่ใหญ่กว่าของความแม่นยำจะถูกใช้โดยส่วนของจำนวนวินาทีทั้งหมด เทคนิคที่แสดงอยู่ในส่วนแรกของภาคผนวกนี้ จะหลีกเลี่ยงปัญหานี้โดยดำเนินการลบออกเพื่อขอรับเวลาที่ใช้ไป ก่อนที่จะแปลงเป็นเลขทศนิยม

การเข้าถึงเรจิสเตอร์ตัวจับเวลาในระบบ PowerPC

สถาปัตยกรรม PowerPC จะสอดแทรกเรจิสเตอร์ *TimeBase* ขนาด 64 บิต ซึ่งจะแบ่งออกเป็นฟิลด์ด้านบนและด้านล่าง ด้านละ 32 บิต (TBU และ TBL)

เรจิสเตอร์ *TimeBase* จะเพิ่มขึ้นที่ความถี่ที่ต้องพึ่งพาการนำฮาร์ดแวร์ไปใช้งาน และการนำซอฟต์แวร์ไปใช้งาน และสามารถเปลี่ยนแปลงตามเวลา การแปลงสภาพค่าจาก TimeBase ไปเป็นวินาทีคือภารกิจที่ซับซ้อนกว่าในสถาปัตยกรรม POWER processor-based หากต้องการขอรับค่าเวลาในระบบ PowerPC ให้ใช้อินเตอร์เฟส **read_real_time( )**, **read_wall_time( )** และ **time_base_to_time( )**

ตัวอย่างรูทีนย่อย second

โปรแกรมสามารถใช้รูทีนย่อย **second( )** ได้

```
#include <stdio.h>
double second();
main()
{
      double t1,t2;

      t1 = second();
      my_favorite_function();
      t2 = second();

     printf("my_favorite_function time: %7.9f\n",t2 - t1);
      exit();
}
```

ตัวอย่าง (`main.f`) โปรแกรม FORTRAN โดยใช้รูทีนย่อย **second( )** มีดังต่อไปนี้:

```
      double precision t1
      double precision t2
```

```
      t1 = second()
      my_favorite_subroutine()
      t2 = second()
      write(6,11) (t2 - t1)
11    format(f20.12)
      end
```

หากต้องการคอมไพล์และใช้ main.c หรือ main.f อย่างใดอย่างหนึ่ง ให้ใช้คำสั่งต่อไปนี้:

```
xlc -O3 -c second.c timer.s
xlf -O3 -o mainF main.f second.o timer.o
xlc -O3 -o mainC main.c second.o timer.o
```

การพิจารณาถึงความเร็วของไมโครโพรเซสเซอร์

ส่วนนี้อธิบายถึงการประมวลผลเพื่อพิจารณาถึงความเร็วของ ไมโครโพรเซสเซอร์

เมื่อต้องการใช้ความเร็วของตัวประมวลผลในหน่วยเฮิร์ซ (Hz) ให้ป้อนคำสั่ง:

```
lsattr -E -l proc0 | grep "Processor Speed"
```

เมื่อ ใช้รีลีสก่อนหน้า ให้ใช้คำสั่ง **uname** การรันคำสั่ง **uname -m** จะสร้างเอาต์พุตในรูปแบบต่อไปนี้:

```
xxyyyyyymmss
```

โดยที่:

xx 00

yyyyyy CPU ID เฉพาะ

mm ID แบบจำลอง (จำนวนที่ใช้เพื่อพิจารณาถึงความเร็วของไมโครโพรเซสเซอร์)

ss 00 (แบบจำลองย่อย)

ด้วยการอ้างถึงค่า mm ระหว่างกันจากเอาต์พุต **uname -m** ตามตารางด้านล่าง คุณสามารถพิจารณาถึงความเร็วของตัวประมวลผลได้

Model ID	Machine Type	Processor Speed	Architecture
02	7015-930	25	Power
10	7013-530	25	Power
10	7016-730	25	Power
11	7013-540	30	Power
14	7013-540	30	Power
18	7013-53H	33	Power
1C	7013-550	41.6	Power
20	7015-930	25	Power
2E	7015-950	41	Power
30	7013-520	20	Power
31	7012-320	20	Power
34	7013-52H	25	Power
35	7012-32H	25	Power
37	7012-340	33	Power
38	7012-350	41	Power
41	7011-220	33	RSC

43	7008-M20	33	Power
43	7008-M2A	33	Power
46	7011-250	66	PowerPC
47	7011-230	45	RSC
48	7009-C10	80	PowerPC
4C		See Note 1.	
57	7012-390	67	Power2
57	7030-3BT	67	Power2
57	9076-SP2 Thin	67	Power2
58	7012-380	59	Power2
58	7030-3AT	59	Power2
59	7012-39H	67	Power2
59	9076-SP2 Thin w/L2	67	Power2
5C	7013-560	50	Power
63	7015-970	50	Power
63	7015-97B	50	Power
64	7015-980	62.5	Power
64	7015-98B	62.5	Power
66	7013-580	62.5	Power
67	7013-570	50	Power
67	7015-R10	50	Power
70	7013-590	66	Power2
70	9076-SP2 Wide	66	Power2
71	7013-58H	55	Power2
72	7013-59H	66	Power2
72	7015-R20	66	Power2
72	9076-SP2 Wide	66	Power2
75	7012-370	62	Power
75	7012-375	62	Power
75	9076-SP1 Thin	62	Power
76	7012-360	50	Power
76	7012-365	50	Power
77	7012-350	41	Power
77	7012-355	41	Power
77	7013-55L	41.6	Power
79	7013-591	77	Power2
79	9076-SP2 Wide	77	Power2
80	7015-990	71.5	Power2
81	7015-R24	71.5	Power2
89	7013-595	135	P2SC
89	9076-SP2 Wide	135	P2SC
94	7012-397	160	P2SC
94	9076-SP2 Thin	160	P2SC
A0	7013-J30	75	PowerPC
A1	7013-J40	112	PowerPC
A3	7015-R30	See Note 2.	PowerPC
A4	7015-R40	See Note 2.	PowerPC
A4	7015-R50	See Note 2.	PowerPC
A4	9076-SP2 High	See Note 2.	PowerPC
A6	7012-G30	See Note 2.	PowerPC
A7	7012-G40	See Note 2.	PowerPC
C0	7024-E20	See Note 3.	PowerPC
C0	7024-E30	See Note 3.	PowerPC
C4	7025-F30	See Note 3.	PowerPC
F0	7007-N40	50	ThinkPad

หมายเหตุ:

1. สำหรับระบบที่คำสั่ง **uname -m** สร้างเอาต์พุต ID แบบจำลองของ 4C โดยทั่วไปแล้ว มีเพียงวิธีเดียวในการพิจารณาถึงความเร็วของตัวประมวลผลของเครื่อง ที่มี ID แบบจำลองของ 4C คือ การรีบูตใน System Management Services และเลือกอ็อพชันคอนฟิกูเรชันระบบ อย่างไรก็ตาม ในบางกรณี ข้อมูลที่ได้รับจากคำสั่ง **uname -M** อาจมีประโยชน์ ดังที่แสดงอยู่ในตารางต่อไปนี้

```
uname -M           Machine Type    Processor Speed    Processor Architecture
IBM,7017-S70         7017-S70          125                RS64
IBM,7017-S7A         7017-S7A          262                RD64-II
IBM,7017-S80         7017-S80          450                RS-III
IBM,7025-F40         7025-F40          166/233            PowerPC
IBM,7025-F50         7025-F50          See Note 4.        PowerPC
IBM,7026-H10         7026-H10          166/233            PowerPC
IBM,7026-H50         7026-H50          See Note 4.        PowerPC
IBM,7026-H70         7026-H70          340                RS64-II
IBM,7042/7043 (ED)   7043-140          166/200/233/332    PowerPC
IBM,7042/7043 (ED)   7043-150          375                PowerPC
IBM,7042/7043 (ED)   7043-240          166/233            PowerPC
IBM,7043-260         7043-260          200                Power3
IBM,7248             7248-100          100                PowerPersonal
IBM,7248             7248-120          120                PowerPersonal
IBM,7248             7248-132          132                PowerPersonal
IBM,9076-270      9076-SP Silver Node  See Note 4.        PowerPC
```

2. สำหรับระบบ J-Series, R-Series และ G-Series คุณสามารถพิจารณาถึงความเร็วของตัวประมวลผลในระบบ MCA SMP ได้จากหมายเลข FRU ของการ์ดไมโครโพรเซสเซอร์ โดยใช้คำสั่งต่อไปนี้:

```
# lscfg -vl cpucard0 | grep FRU
FRU Number    Processor Type    Processor Speed
     E1D           PowerPC 601       75
     C1D           PowerPC 601       75
     C4D           PowerPC 604       112
     E4D           PowerPC 604       112
     X4D           PowerPC 604e      200
```

3. สำหรับระบบ E-series และ F-30 ให้ใช้คำสั่งต่อไปนี้เพื่อพิจารณาถึงความเร็วของ ไมโครโพรเซสเซอร์:

```
# lscfg -vp | pg
```

มองหา stanza ต่อไปนี้:

```
procF0                              CPU Card

Part Number.................093H5280
EC Level....................00E76527
Serial Number...............17700008
FRU Number..................093H2431
Displayable Message.........CPU Card
Device Specific.(PL)........
Device Specific.(ZA)........PS=166,PB=066,PCI=033,NP=001,CL=02,PBH
                                      Z=64467000,PM=2.5,L2=1024
Device Specific.(RM)........10031997 140951 VIC97276
ROS Level and ID............03071997 135048
```

ในส่วนนี้ Device Specific.(ZA) ส่วนของ PS= คือความเร็วของตัวประมวลผลที่แสดงในหน่วย MHz

4. สำหรับระบบ F-50 และ H-50 และ SP Silver Node คำสั่งต่อไปนี้สามารถนำมาใช้ เพื่อกำหนดความเร็วของตัวประมวลผลของระบบ F-50 ได้:

```
# lscfg -vp | more
```

มองหา stanza ต่อไปนี้:

```
Orca M5 CPU:
Part Number.................08L1010
EC Level....................E78405
Serial Number...............L209034579
FRU Number..................93H8945
Manufacture ID..............IBM980
Version.....................RS6K
Displayable Message.........OrcaM5 CPU DD1.3
Product Specific.(ZC).......PS=0013c9eb00,PB=0009e4f580,SB=0004f27
                                        ac0,NP=02,PF=461,PV=05,KV=01,CL=1
```

สำหรับบรรทัดที่มี `Product Specific.(ZC)` รายการ `PS=` คือความเร็วของตัวประมวลผลใน hexadecimal notation หากต้องการแปลงค่านี้ให้เป็นความเร็วที่เกิดขึ้นจริง ให้ใช้การแปลงต่อไปนี้:

```
0009E4F580 = 166 MHz
0013C9EB00 = 332 MHz
```

ค่า `PF=` บ่งชี้ถึงคอนฟิกูเรชันของตัวประมวลผล

```
251 = 1 way 166 MHz
261 = 2 way 166 MHz
451 = 1 way 332 MHz
461 = 2 way 332 MHz
```

การสนับสนุนภาษาประจำชาติ: โลแคลและความเร็ว

การสนับสนุนภาษาประจำชาติ (National Language Support หรือ NLS) ช่วยสนับสนุนการใช้ระบบปฏิบัติการ ในสภาพแวดล้อมภาษาต่างๆ เนื่องจากมีข้อมูลว่าการใช้ NLS มีความสำคัญมากขึ้น ต่อการได้รับประสิทธิภาพสูงสุดจากระบบ ดังนั้น ภาคผนวกนี้จึงมี บททบทวนโดยย่อเกี่ยวกับ NLS

NLS ช่วยให้สามารถปรับแต่งระบบปฏิบัติการให้ตรงกับความคาดหวังด้านภาษา และวัฒนธรรมของผู้ใช้แต่ละรายได้ *โลแคล* คือชุดเฉพาะของความต้องการด้าน ภาษาและภูมิศาสตร์หรือวัฒนธรรม ซึ่งระบุโดยชื่อประกอบ เช่น en_US (ภาษาอังกฤษแบบที่ใช้ในประเทศสหรัฐฯ) สำหรับแต่ละ โลแคลที่สนับสนุน มีชุดของแค็ตตาล็อกข้อความ ตารางค่าการจัดเรียง และข้อมูลอื่นที่กำหนดความต้องการของโลแคลนั้น เมื่อมีการติดตั้งระบบปฏิบัติการ ผู้ดูแลระบบสามารถเลือกข้อมูลโลแคล ที่ควรจะติดตั้งได้ หลังจากนั้น ผู้ใช้แต่ละรายสามารถ ควบคุมโลแคลของแต่ละ shell โดยการเปลี่ยนตัวแปร *LANG* และ *LC_ALL*

โลแคลหนึ่งที่ไม่เป็นไปตามโครงสร้างที่เพิ่งอธิบายไปคือโลแคล C (หรือ POSIX) โลแคล C คือโลแคลดีฟอลต์ของระบบ ยกเว้นว่าผู้ใช้ เลือกโลแคลอื่นด้วยตนเอง และยังเป็นโลแคลซึ่งเริ่มต้นแต่ละกระบวนการที่ forked ใหม่ด้วย การรันในโลแคล C ใกล้เคียงที่สุดกับระบบปฏิบัติการ ที่รันในรูปแบบ unilingual ดั้งเดิมของ UNIX ไม่มีแค็ตตาล็อกข้อความ C โปรแกรมที่พยายามเรียกใช้ข้อความจากแค็ตตาล็อก จะได้รับข้อความดีฟอลต์ซึ่งคอมไพล์เข้าในโปรแกรม แทน คำสั่งบางอย่าง เช่น คำสั่ง **sort** แปลงเป็น ขั้นตอนวิธีเฉพาะชุดอักขระดั้งเดิมของตน

โดยทั่วไป ประสิทธิภาพของ NLS แบ่งออกเป็นสามระดับ โดยทั่วไป โลแคล C คือ วิธีการที่รวดเร็วที่สุดสำหรับการดำเนินการคำสั่ง ตามด้วยโลแคลไบต์เดียว (ตัวอักษรลาติน) เช่น en_US และโลแคลหลายไบต์ที่มีการดำเนินการ คำสั่งช้าที่สุด

ข้อควรพิจารณาเกี่ยวกับการเขียนโปรแกรม

มีประเด็นการเขียนโปรแกรมหลายอย่างที่เกี่ยวข้องกับการสนับสนุน ภาษาประจำชาติ

ในอดีตที่ผ่านมา มีการใช้ภาษา C เพื่อแสดงหลักการเฉพาะภาษาจำนวนหนึ่ง ในการใช้ไบต์คำและอักขระที่แลกเปลี่ยนได้ ดังนั้น อาร์เรย์ `char foo[10]` ที่ใช้จึงเป็นอาร์เรย์ 10 ไบต์ แต่ไม่ใช่ว่าทุกภาษาในโลกนี้ จะสามารถบันทึกด้วยอักขระซึ่งสามารถระบุในไบต์เดียว ได้ ตัวอย่างเช่น ภาษาญี่ปุ่นและภาษาจีนต้องใช้ไบต์จำนวนสองไบต์ขึ้นไป เพื่อระบุรูปภาพเฉพาะที่จะแสดงขึ้น ด้วยเหตุนี้ เราจึงแยกความแตกต่างระหว่าง ไบต์ซึ่งประกอบด้วย 8 บิตของข้อมูล และอักขระซึ่งเป็นจำนวนข้อมูล ที่ต้องใช้ในการแสดงรูปภาพหนึ่งรูป

ลักษณะเฉพาะสองอย่างของแต่ละโลแคลคือจำนวนไบต์สูงสุดที่ต้องใช้ เพื่อระบุอักขระในโลแคลนั้น และจำนวนสูงสุดของตำแหน่งการแสดงเอาต์พุต ซึ่งอักขระตัวหนึ่งสามารถครอบครองได้ ค่าเหล่านี้สามารถหาได้โดยใช้แมโคร **MB_CUR_MAX** และ **MAX_DISP_WIDTH** ถ้าค่าทั้งสองเป็น 1 แสดงว่าโลแคลคือตำแหน่งที่ยังคงถือครองไบต์และอักขระได้อย่างเท่าเทียมกัน ถ้าค่าใดค่าหนึ่งมากกว่า 1 แสดงว่าโปรแกรมที่ใช้การประมวลผล ทีละอักขระ หรือที่เก็บประวัติจำนวนของตำแหน่งการแสดงผลที่ใช้ ต้องใช้ฟังก์ชัน ความเป็นสากลเพื่อดำเนินการดังกล่าว

เนื่องจากการเข้ารหัสหลายไบต์ประกอบด้วยจำนวนไบต์ต่ออักขระที่ไม่แน่นอน ดังนั้นจึงไม่สามารถประมวลผลเป็นอาร์เรย์ของอักขระได้ เพื่อให้การเข้ารหัสในกรณีที่ อักขระแต่ละตัวต้องมีการประมวลผลยาวนานเป็นไปอย่างมีประสิทธิภาพ จึงมีการกำหนดชนิดข้อมูลความกว้างไบต์คงที่, **wchar_t**, ขึ้น **wchar_t** กว้าง เพียงพอที่จะรองรับรูปแบบที่แปลของการเข้ารหัสอักขระที่สนับสนุนใดๆ ดังนั้น โปรแกรมเมอร์จึงสามารถใช้อาร์เรย์ของ **wchar_t** และประมวลผล ด้วย (อย่างคร่าวๆ) ตรรกะเดียวกันกับที่ใช้กับอาร์เรย์ของ **char** ได้ โดยใช้แอนะล็อก wide-character ของฟังก์ชัน `libc.a` ดั้งเดิม

โชคไม่ดีนัก การแปลจากรูปแบบหลายไบต์ซึ่งใช้ในการป้อนข้อความ จัดเก็บข้อความบนดิสก์ หรือบันทึกข้อความในการแสดงผล เป็นรูปแบบ **wchar_t** ต้องใช้ค่าใช้จ่ายค่อนข้างสูง การแปลดังกล่าวจึงควรทำเฉพาะในกรณีที่ ประสิทธิผลของการประมวลผลรูปแบบ **wchar_t** คุ้มค่ากว่าต้นทุนในการแปล ไปเป็นหรือแปลจากรูปแบบ **wchar_t** เท่านั้น

กฎการทำให้ง่ายขึ้นบางข้อ

สามารถบันทึกแอ็พพลิเคชันโปรแกรมหลายภาษาที่ทำงานช้าได้ ถ้าโปรแกรมเมอร์ไม่ทราบข้อจำกัดบางอย่างในการออกแบบชุดอักขระหลายไบต์ ซึ่งช่วยให้โปรแกรมจำนวนมากสามารถรันได้อย่างมีประสิทธิภาพในโลแคลหลาย ไบต์ ด้วยการใช้ฟังก์ชันการทำให้เป็นสากลเพียงเล็กน้อยเท่านั้น

ตัวอย่างเช่น:

- ในชุดรหัสทั้งหมดที่ได้รับการสนับสนุนจาก IBM รหัสอักขระ 0x00 ถึง 0x3F เป็นค่าเฉพาะและเข้ารหัสอักขระมาตรฐาน ASCII เฉพาะหมายความว่าชุดบิตเหล่านี้ไม่มีทางปรากฏเป็น ไบต์ใดไบต์หนึ่งของอักขระหลายไบต์ เนื่องจากอักขระ null เป็นส่วนประกอบหนึ่ง ของชุดนี้ ฟังก์ชัน **strlen( )**, **strcpy( )**, และ **strcat( )** จึงทำงาน ทั้งบนหลายไบต์และสตริงไบต์เดียว โปรแกรมเมอร์ต้องจำว่า ค่าที่ส่งคืนโดย **strlen( )** คือจำนวนไบต์ในสตริง ไม่ใช่จำนวนอักขระ
- ในลักษณะคล้ายกัน ฟังก์ชันสตริงมาตรฐาน **strchr(foostr, '/')** ทำงาน อย่างถูกต้องในโลแคลทั้งหมด เนื่องจาก / (เครื่องหมายขีด) เป็นส่วนประกอบหนึ่ง ของช่วงจุดรหัสเฉพาะ ในข้อเท็จจริง delimiters มาตรฐานส่วนใหญ่อยู่ในช่วง 0x00 ถึง 0x3F ดังนั้นการแจงส่วนส่วนใหญ่จึงสามารถทำได้โดยไม่ต้องกลับไปยังฟังก์ชันการทำเป็น สากลหรือแปลเป็นรูปแบบ **wchar_t**
- การเปรียบเทียบระหว่างสตริงแบ่งออกเป็นสองคลาสคือ: เท่ากันหรือไม่เท่ากัน ใช้ฟังก์ชัน **strcmp( )** มาตรฐานเพื่อทำการเปรียบเทียบ เมื่อคุณ บันทึก

```
if (strcmp(foostr,"a rose") == 0)
```

คุณไม่ได้ ค้นหา "a rose" ตามชื่ออื่นใด แต่คุณค้นหาชุดของบิตนั้น เท่านั้น ถ้า foostr มี "a rosE" จะไม่พบ รายการที่ตรงกัน

- การเปรียบเทียบแบบไม่เท่ากันเกิดขึ้นเมื่อคุณพยายามจัดเรียงสตริงใน ลำดับการจัดเรียงที่โลแคลกำหนด ในกรณีนั้น คุณจะใช้

  ```
  if (strcoll(foostr,barstr) > 0)
  ```

  และต้อง สูญเสียประสิทธิภาพเพื่อให้ได้ข้อมูลการจัดเรียงสำหรับอักขระ แต่ละตัว
- เมื่อโปรแกรมดำเนินการ โปรแกรมจะเริ่มต้นในโลแคล C เสมอ ถ้าจะใช้ฟังก์ชัน การทำเป็นสากลตั้งแต่หนึ่งรายการขึ้นไป รวมถึงการเข้าถึงแค็ตตาล็อกข้อความ โปรแกรมต้องดำเนินการ:

  ```
  setlocale(LC_ALL, "");
  ```

  เพื่อ สลับไปยังโลแคลของกระบวนการพาเรนต์ก่อนที่จะเรียกฟังก์ชัน การทำเป็นสากลใดๆ

การตั้งค่าโลแคล

ใช้คำสั่ง **export** เพื่อตั้งค่าโลแคล

ลำดับคำสั่งดังต่อไปนี้:

```
LANG=C
export LANG
```

ตั้งค่าดีฟอลต์โลแคลเป็น C (นั่นคือใช้ C ยกเว้นว่าตัวแปรที่กำหนด เช่น *LC_COLLATE* มีการตั้งค่าเป็นอย่างอื่น)

ลำดับดังต่อไปนี้:

```
LC_ALL=C
export LC_ALL
```

บังคับการตั้งค่าตัวแปรโลแคลทั้งหมดเป็น C โดยไม่คำนึงถึง ค่าติดตั้งก่อนหน้านี้

สำหรับรายงาน บนค่าติดตั้งปัจจุบันของตัวแปรโลแคล ให้พิมพ์ `locale`

พารามิเตอร์ที่ปรับได้

มีพารามิเตอร์ระบบปฏิบัติการหลายตัวที่อาจส่งผลกระทบต่อ ประสิทธิภาพได้

พารามิเตอร์มีการอธิบายในลำดับตามตัวอักษรภายในแต่ละส่วน

ตัวแปรสภาวะแวดล้อม

มีตัวแปรสภาวะแวดล้อมอยู่สองแบบ คือ: พารามิเตอร์ส่วนสนับสนุนเธรดที่สามารถปรับแต่งได้ และพารามิเตอร์เบ็ดเตล็ดที่สามารถปรับแต่งได้

พารามิเตอร์ที่ปรับได้ของการสนับสนุนเธรด

มีพารามิเตอร์การสนับสนุนเธรดหลายตัวที่สามารถปรับได้

1. **ACT_TIMEOUT**

ไอเท็ม	Descriptor
วัตถุประสงค์:	ปรับวินาทีสำหรับ ไทม์เอาต์การเรียกใช้
ค่า:	ดีฟอลต์: DEF_ACTOUT ช่วง: จำนวนเต็มบวก
การแสดงผล:	**echo $ACT_TIMEOUT** ค่านี้จะถูกเปิดใช้ภายใน และดังนั้นค่าดีฟอลต์เริ่มต้นจะไม่ถูกแสดงด้วยคำสั่ง **echo**
การเปลี่ยนแปลง:	**ACT_TIMEOUT=n export ACT_TIMEOUT** การเปลี่ยนแปลงจะมีผลบังคับใช้โดยทันทีใน shell นี้ การเปลี่ยนแปลง มีผลบังคับใช้จนกระทั่งล็อกเอาต์จาก shell นี้ การเปลี่ยนแปลงถาวรมีการทำโดยการเพิ่มคำสั่ง **ACT_TIMEOUT**=*n* ลงในไฟล์ /etc/environment
การวิเคราะห์:	N/A
การปรับ:	N/A

อ้างถึง: "ตัวแปรสภาพแวดล้อมเธรด" ในหน้า 78.

2. **AIXTHREAD_COND_DEBUG**

ไอเท็ม	Descriptor
วัตถุประสงค์:	เก็บรักษารายการของตัวแปรเงื่อนไข สำหรับใช้โดย debugger
ค่า:	ดีฟอลต์: OFF ช่วง: ON, OFF
การแสดงผล:	**echo $AIXTHREAD_COND_DEBUG** ค่านี้จะถูกเปิดใช้ภายใน และดังนั้นค่าดีฟอลต์เริ่มต้นจะไม่ถูกแสดงด้วยคำสั่ง **echo**
การเปลี่ยนแปลง:	AIXTHREAD_COND_DEBUG=*{ON|OFF}*export AIXTHREAD_COND_DEBUG การเปลี่ยนแปลงมีผลทันทีใน shell นี้ การเปลี่ยนแปลง มีผลบังคับใช้จนกระทั่งล็อกเอาต์จาก shell นี้ การเปลี่ยนแปลง ถาวรมีการทำโดยการเพิ่มคำสั่ง **AIXTHREAD_COND_DEBUG=*{ON|OFF}*** ลงในไฟล์ /etc/environment
การวิเคราะห์:	การปล่อยให้ตัวแปรนี้มีการตั้งค่า เป็น **ON** ทำให้การดีบักเธรดแอ็พพลิเคชันง่ายขึ้น แต่อาจส่งผลให้โอเวอร์เฮดเพิ่มขึ้น
การปรับ:	ถ้าโปรแกรมมี ตัวแปรเงื่อนไขที่ใช้งานอยู่จำนวนมาก และสร้างและทำลายตัวแปรเงื่อนไข บ่อยครั้ง อาจทำให้โอเวอร์เฮดสำหรับการเก็บรักษารายการของตัวแปรเงื่อนไข สูงขึ้น การตั้งค่าตัวแปรเป็นเป็น **OFF** จะปิดการใช้รายการ

โปรดอ้างอิง "อ็อพชันการดีบักเธรด" ในหน้า 83

3. **AIXTHREAD_DISCLAIM_GUARDPAGES**

ไอเท็ม	Descriptor
วัตถุประสงค์:	ควบคุมว่า stack guardpages ถูกทำให้ชัดเจนหรือไม่
ค่า:	ดีฟอลต์: OFF ช่วง: ON, OFF
การแสดงผล:	**echo $AIXTHREAD_DISCLAIM_GUARDPAGES** ค่านี้จะถูกเปิดใช้ภายใน และดังนั้นค่าดีฟอลต์เริ่มต้นจะไม่ถูกแสดงด้วยคำสั่ง **echo**
การเปลี่ยนแปลง:	AIXTHREAD_DISCLAIM_GUARDPAGES={ON|OFF};export AIXTHREAD_DISCLAIM_GUARDPAGES การเปลี่ยนแปลงมีผลทันทีใน shell นี้ การเปลี่ยนแปลงมีผลบังคับใช้จนกระทั่งล็อกเอาต์จาก shell นี้ การเปลี่ยนแปลงถาวรมีการทำโดยการเพิ่มคำสั่ง **AIXTHREAD_GUARDPAGES**=*n* ลงในไฟล์ /etc/environment
การวิเคราะห์:	NA
การปรับ:	ถ้า guardpages ถูกใช้สำหรับสแต็ก pthread การตั้งค่า **AIXTHREAD_DISCLAIM_GUARDPAGES = ON** จะทำให้ guardpages ชัดเจนขึ้นเมื่อ pthreads ถูกสร้างขึ้น พารามิเตอร์นี้สามารถลด footprint ของหน่วยความจำของแอ็พพลิเคชันที่มีเธรด

โปรดอ้างอิง "ตัวแปรสภาพแวดล้อมเธรด" ในหน้า 78

4. **AIXTHREAD_ENRUSG**

ไอเท็ม	**Descriptor**
วัตถุประสงค์:	เปิดใช้งานหรือปิดใช้งานการรวบรวมรีซอร์ส pthread
ค่า:	ดีฟอลต์: OFF ช่วง: ON, OFF
การแสดงผล:	**echo $AIXTHREAD_ENRUSG**
การเปลี่ยนแปลง:	ค่านี้ เปิดขึ้นภายใน ดังนั้นจะไม่เห็นค่าดีฟอลต์แรกเริ่ม เมื่อใช้คำสั่ง **echo** AIXTHREAD_ENRUSG=*{ON|OFF}*export AIXTHREAD_ENRUSG
การวิเคราะห์:	การเปลี่ยนแปลงมีผลทันทีใน shell นี้ การเปลี่ยนแปลง มีผลบังคับใช้จนกระทั่งล็อกเอาต์จาก shell นี้ การเปลี่ยนแปลงถาวร มีการทำโดยการเพิ่มคำสั่ง **AIXTHREAD_ENRUSG=*{ON|OFF}*** ลงในไฟล์ /etc/environment การตั้งค่าพารามิเตอร์นี้เป็น **ON** ช่วยให้สามารถจัดสรรรีซอร์สของ pthreads ทั้งหมดในกระบวนการ หนึ่งได้ แต่จะเพิ่มโอเวอร์เฮด
การปรับ:	N/A

โปรดอ้างอิง “ตัวแปรสภาพแวดล้อมเธรด” ในหน้า 78

5. **AIXTHREAD_GUARDPAGES**

ไอเท็ม	**Descriptor**
วัตถุประสงค์:	ควบคุมจำนวนของ guardpages ที่ต้องการเพิ่มเข้ากับท้ายของสแต็ก pthread
ค่า:	ดีฟอลต์: 1 (โดย 1 คือ ค่าทศนิยมสำหรับจำนวนเพจ ซึ่งสามารถเป็น 4K, 64K และ อื่นๆ) ขอบเขต: ขอบเขตของ n
การแสดงผล:	**echo $AIXTHREAD_GUARDPAGES**
การเปลี่ยนแปลง:	ค่านี้ เปิดขึ้นภายในระบบ ดังนั้นจะไม่เห็นค่าดีฟอลต์แรกเริ่ม เมื่อใช้คำสั่ง **echo** AIXTHREAD_GUARDPAGES=*n*export AIXTHREAD_GUARDPAGES
การวิเคราะห์:	การเปลี่ยนแปลงมีผลทันทีใน shell นี้ การเปลี่ยนแปลง มีผลบังคับใช้จนกระทั่งล็อกเอาต์จาก shell นี้ การเปลี่ยนแปลงถาวรมีการทำโดยการเพิ่มคำสั่ง **AIXTHREAD_GUARDPAGES=*n*** ลงในไฟล์ /etc/environment
การปรับ:	N/A

โปรดอ้างอิง “ตัวแปรสภาพแวดล้อมเธรด” ในหน้า 78

6. **AIXTHREAD_MINKTHREADS**

ไอเท็ม	**Descriptor**
วัตถุประสงค์:	ควบคุม จำนวนต่ำสุดของเคอร์นัลเธรดที่ควรจะใช้
ค่า:	ดีฟอลต์: 8 ช่วง: ค่าจำนวนเต็ม บวก
การแสดงผล:	**echo $AIXTHREAD_MINKTHREADS**
การเปลี่ยนแปลง:	ค่านี้ เปิดขึ้นภายในระบบ ดังนั้นจะไม่เห็นค่าดีฟอลต์แรกเริ่ม เมื่อใช้คำสั่ง **echo** **AIXTHREAD_MINKTHREADS=*n*export AIXTHREAD_MINKTHREADS**
การวิเคราะห์:	การเปลี่ยนแปลงมีผลทันทีใน shell นี้ การเปลี่ยนแปลงมีผลบังคับใช้จนกระทั่งล็อกเอาต์จาก shell นี้ การเปลี่ยนแปลงถาวรมีการทำโดยการเพิ่มคำสั่ง **AIXTHREAD_MINKTHREADS=*n*** ลงในไฟล์ /etc/environment N/A
การปรับ:	ตัวจัดตารางเวลาไลบรารี จะไม่เรียกร้องคืนเคอร์นัลเธรดต่ำกว่าค่าที่ตั้งไว้ในตัวแปรนี้ เคอร์นัลเธรดอาจถูกเรียกร้องคืนในเวลาใดก็ได้ในแบบเสมือน โดยทั่วไป มีความต้องการเคอร์นัลเธรดเนื่องจากการยุติ pthread

อ้างถึง: “ตัวแปรสำหรับขอบเขตการช่วงชิงทั้งกระบวนการ” ในหน้า 83.

7. **AIXTHREAD_MNRATIO**

ไอเท็ม **Descriptor**
วัตถุประสงค์: ควบคุมปัจจัยการปรับสเกล ของไลบรารี อัตราส่วนนี้ใช้เมื่อสร้างและยุติ pthreads
ค่า: ดีฟอลต์: 8:1 ช่วง: ค่าบวก สองค่า (*p:k*) โดยที่ *k* คือจำนวนของเคอร์นัลเธรด ที่ควรจะใช้เพื่อจัดการกับจำนวนของ pthreads ที่ดำเนินการได้ซึ่งกำหนดในตัวแปร *p*
การแสดงผล: **echo $AIXTHREAD_MNRATIO**

ค่านี้ เปิดขึ้นภายในระบบ ดังนั้นจะไม่เห็นค่าดีฟอลต์แรกเริ่ม เมื่อใช้คำสั่ง **echo**
การเปลี่ยนแปลง: **AIXTHREAD_MNRATIO=*p:k*export AIXTHREAD_MNRATIO**

การเปลี่ยนแปลงมีผลทันทีใน shell นี้ การเปลี่ยนแปลง มีผลบังคับใช้จนกระทั่งล็อกเอาต์จาก shell นี้ การเปลี่ยนแปลงถาวร มีการทำโดยการเพิ่มคำสั่ง **AIXTHREAD_MNRATIO=*p:k*** ลงในไฟล์ /etc/environment
การวิเคราะห์: N/A
การปรับ: อาจเป็นประโยชน์สำหรับแอ็พพลิเคชัน ที่มีเธรดจำนวนมาก อย่างไรก็ตาม ควรทดสอบอัตราส่วน 1:1 เสมอเนื่องจากเป็นอัตราส่วนที่ให้ประสิทธิภาพดีขึ้น

อ้างอิง ถึง: “ตัวแปรสำหรับขอบเขตการช่วงชิงทั้งกระบวนการ” ในหน้า 83

8. **AIXTHREAD_MUTEX_DEBUG**

ไอเท็ม **Descriptor**
วัตถุประสงค์: เก็บรักษารายการของ mutexes ที่ใช้งานอยู่สำหรับการใช้โดย debugger
ค่า: ดีฟอลต์: OFF ค่า ที่เป็นไปได้: ON, OFF
การแสดงผล: **echo $AIXTHREAD_MUTEX_DEBUG**

ค่านี้ เปิดขึ้นภายในระบบ ดังนั้นจะไม่เห็นค่าดีฟอลต์แรกเริ่ม เมื่อใช้คำสั่ง **echo**
การเปลี่ยนแปลง: **AIXTHREAD_MUTEX_DEBUG=*{ON|OFF}*export AIXTHREAD_MUTEX_DEBUG**

การเปลี่ยนแปลงนี้มีผลในทันที และมีผลจนกว่าคุณล็อกเอาต์ออกจาก shell นี้ การเปลี่ยนแปลงถาวร มีการทำโดยการเพิ่มคำสั่ง **AIXTHREAD_MUTEX_DEBUG=*{ON|OFF}*** ลงในไฟล์ /etc/environment
การวิเคราะห์: การตั้งค่าตัวแปรนี้เป็น ON ทำให้ การดีบักเธรดแอ็พพลิเคชันง่ายขึ้น แต่อาจส่งผลให้โอเวอร์เฮดเพิ่มขึ้น
การปรับ: ถ้าโปรแกรมมี mutexes ที่ใช้งานอยู่จำนวนมาก และสร้างและทำลาย mutexes บ่อยครั้ง อาจทำให้โอเวอร์เฮดสำหรับการเก็บรักษารายการของ mutexes สูงขึ้น การปล่อยให้ตัวแปรมีการตั้งค่าเป็น OFF จะปิดใช้งานรายการ

อ้างอิงถึง: “อ็อพชันการดีบักเธรด” ในหน้า 83

9. **AIXTHREAD_MUTEX_FAST**

ไอเท็ม **Descriptor**
วัตถุประสงค์: เปิดใช้งานการใช้กลไกการล็อก mutex ที่ดีที่สุด
ค่า: ดีฟอลต์: OFF ค่า ที่เป็นไปได้: ON, OFF
การแสดงผล: **echo $AIXTHREAD_MUTEX_FAST**

ค่านี้ เปิดขึ้นภายในระบบ ดังนั้นจะไม่เห็นค่าดีฟอลต์แรกเริ่ม เมื่อใช้คำสั่ง **echo**
การเปลี่ยนแปลง: **AIXTHREAD_MUTEX_FAST=*{ON|OFF}*export AIXTHREAD_MUTEX_FAST**

การเปลี่ยนแปลงนี้มีผลในทันที และมีผลจนกว่าคุณล็อกเอาต์ออกจาก shell นี้ การเปลี่ยนแปลงถาวร มีการทำโดยการเพิ่มคำสั่ง **AIXTHREAD_MUTEX_FAST=*{ON|OFF}*** ลงในไฟล์ /etc/environment
การวิเคราะห์: การตั้งค่าตัวแปรเป็น ON บังคับให้ เธรดแอ็พพลิเคชันใช้กลไกการล็อก mutex ที่ดีที่สุด ส่งผลให้ประสิทธิภาพเพิ่มขึ้น
การปรับ: ถ้าโปรแกรมพบกับ การเสื่อมถอยด้านประสิทธิภาพการทำงาน เนื่องจากมีการช่วงชิง mutex อย่างหนัก ด้วยเหตุนี้ การตั้งค่าตัวแปรนี้เป็น ON จะบังคับให้ไลบรารี pthread ใช้กลไกการล็อก mutex ที่ถูกปรับแต่ง ซึ่งทำงานบน mutex ส่วนตัวของกระบวนการ เท่านั้น ต้องเริ่มต้น mutexes ส่วนตัวของกระบวนการ เหล่านี้โดยใช้รูทีน pthread_mutex_init และต้องทำลาย โดยใช้รูทีน pthread_mutex_destroy

อ้างอิงถึง: “อ็อพชันการดีบักเธรด” ในหน้า 83

10. **AIXTHREAD_READ_GUARDPAGES**

ไอเท็ม **Descriptor**
วัตถุประสงค์: ควบคุมการเข้าถึง เพื่ออ่าน guardpages ที่ถูกเพิ่มเข้ากับท้ายของสแต็ก pthread
ค่า: ดีฟอลต์: OFF ช่วง: ON, OFF
การแสดงผล: **echo $AIXTHREAD_READ_GUARDPAGES**

การเปลี่ยนแปลง: ค่านี้ เปิดขึ้นภายในระบบ ดังนั้นจะไม่เห็นค่าดีฟอลต์แรกเริ่ม เมื่อใช้คำสั่ง **echo** AIXTHREAD_READ_GUARDPAGES={ON|OFF}export AIXTHREAD_READ_GUARDPAGES

การวิเคราะห์: การเปลี่ยนแปลงมีผลทันทีใน shell นี้ การเปลี่ยนแปลง มีผลบังคับใช้จนกระทั่งล็อกเอาต์จาก shell นี้ การเปลี่ยนแปลงถาวร มีการทำโดยการเพิ่มคำสั่ง **AIXTHREAD_READ_GUARDPAGES={ON|OFF}** ลงในไฟล์ /etc/environment
การปรับ: N/A
N/A

โปรดอ้างอิง “ตัวแปรสภาพแวดล้อมเธรด” ในหน้า 78

11. AIXTHREAD_RWLOCK_DEBUG

ไอเท็ม **Descriptor**
วัตถุประสงค์: เก็บรักษารายการของล็อกการอ่าน-การบันทึก สำหรับใช้โดย debugger
ค่า: ดีฟอลต์: OFF ช่วง: ON, OFF
การแสดงผล: **echo $AIXTHREAD_RWLOCK_DEBUG**

การเปลี่ยนแปลง: ค่านี้ เปิดขึ้นภายในระบบ ดังนั้นจะไม่เห็นค่าดีฟอลต์แรกเริ่ม เมื่อใช้คำสั่ง **echo** **AIXTHREAD_RWLOCK_DEBUG=*{ON|OFF}*export AIXTHREAD_RWLOCK_DEBUG**

การวิเคราะห์: การเปลี่ยนแปลงมีผลทันทีใน shell นี้ การเปลี่ยนแปลง มีผลบังคับใช้จนกระทั่งล็อกเอาต์จาก shell นี้ การเปลี่ยนแปลงถาวร มีการทำโดยการเพิ่มคำสั่ง **AIXTHREAD_RWLOCK_DEBUG=*{ON|OFF}*** ลงในไฟล์ /etc/environment
การปรับ: การตั้งค่าพารามิเตอร์นี้เป็น ON ทำให้ การดีบักเธรดแอ็พพลิเคชันง่ายขึ้น แต่อาจส่งผลให้โอเวอร์เฮดเพิ่มขึ้น
ถ้าโปรแกรมมีล็อก การอ่าน-การบันทึกที่ใช้งานอยู่จำนวนมาก และสร้างและทำลายล็อกการอ่าน-การ บันทึกบ่อยครั้ง อาจทำให้โอเวอร์เฮดสำหรับการเก็บรักษารายการของล็อกการอ่าน- การบันทึกสูงขึ้น การตั้งค่าตัวแปรเป็น OFF จะปิดใช้งานรายการ

อ้างอิงถึง: “อ็อพชันการดีบักเธรด” ในหน้า 83

12. AIXTHREAD_SUSPENDIBLE

ไอเท็ม **Descriptor**
วัตถุประสงค์: ป้องกัน deadlock ในแอ็พพลิเคชัน ที่ใช้รูทีนต่อไปนี้พร้อมกับรูทีน **pthread_suspend_np** หรือ **pthread_suspend_others_np**:

- pthread_getrusage_np
- pthread_cancel
- pthread_detach
- pthread_join
- pthread_getunique_np
- pthread_join_np
- pthread_setschedparam
- pthread_getschedparam
- pthread_kill

ค่า: ดีฟอลต์: OFF ช่วง: ON, OFF
การแสดงผล: **echo $AIXTHREAD_SUSPENDIBLE**

การเปลี่ยนแปลง: ค่านี้ เปิดขึ้นภายในระบบ ดังนั้นจะไม่เห็นค่าดีฟอลต์แรกเริ่ม เมื่อใช้คำสั่ง **echo** **AIXTHREAD_SUSPENDIBLE=*{ON|OFF}*export AIXTHREAD_SUSPENDIBLE**

การวิเคราะห์: การเปลี่ยนแปลงมีผลทันทีใน shell นี้ การเปลี่ยนแปลง มีผลบังคับใช้จนกระทั่งล็อกเอาต์จาก shell นี้ การเปลี่ยนแปลงถาวร มีการทำโดยการเพิ่มคำสั่ง **AIXTHREAD_SUSPENDIBLE=*{ON|OFF}*** ลงในไฟล์ /etc/environment
ตัวแปรนี้ส่งผลกระทบต่อประสิทธิภาพเล็กน้อย

ไอเท็ม **Descriptor**
การปรับ: ตัวแปรนี้ควรเปิดใช้งานเฉพาะถ้ามีการใช้ ฟังก์ชันที่ระบุไปก่อนหน้านี้พร้อมกับรูทีน **pthread_suspend_np** หรือรูทีน **pthread_suspend_others_np**

อ้างอิงถึง: "อ็อพชันการดีบักเธรด" ในหน้า 83

13. AIXTHREAD_SCOPE

ไอเท็ม **Descriptor**
วัตถุประสงค์: ควบคุมขอบเขตการช่วงชิง ค่า P ระบุขอบเขตการช่วงชิงบนกระบวนการ (M:N) ค่า S ระบุขอบเขตการช่วงชิงบนระบบ (1:1)
ค่า: ดีฟอลต์: P. ค่าที่ใช้ได้: P หรือ S
การแสดงผล: **echo $AIXTHREAD_SCOPE**

ค่านี้ เปิดขึ้นภายในระบบ ดังนั้นจะไม่เห็นค่าดีฟอลต์แรกเริ่ม เมื่อใช้คำสั่ง **echo**
การเปลี่ยนแปลง: **AIXTHREAD_SCOPE=***{P|S}***export AIXTHREAD_SCOPE**

การเปลี่ยนแปลงมีผลทันทีใน shell นี้ การเปลี่ยนแปลง มีผลบังคับใช้จนกระทั่งล็อกเอาต์จาก shell นี้ การเปลี่ยนแปลงถาวร มีการทำโดยการเพิ่มคำสั่ง **AIXTHREAD_SCOPE=***{P|S}* ลงในไฟล์ /etc/environment
การวิเคราะห์: ถ้ามีเธรดที่กำลังจัดส่งน้อยกว่า จำนวนที่คาดไว้ให้ลองขอบเขตระบบ
การปรับ: การทดสอบแสดงว่าบาง แอ็พพลิเคชันสามารถทำงานได้ดีขึ้นด้วยขอบเขตการช่วงชิงบนระบบ (S) การใช้ตัวแปรสภาพแวดล้อมนี้มีผลกระทบต่อเธรด ที่สร้างขึ้นด้วยดีฟอลต์แอ็ตทริบิวต์เท่านั้น ดีฟอลต์แอ็ตทริบิวต์ จะนำมาใช้เมื่อพารามิเตอร์ *attr* ของ **pthread_create** เป็น NULL

อ้างอิง ถึง: "ตัวแปรสภาพแวดล้อมเธรด" ในหน้า 78

14. AIXTHREAD_SLPRATIO

ไอเท็ม **Descriptor**
วัตถุประสงค์: ควบคุมจำนวนของ เคอร์นัลเธรดที่ควรจะเก็บสำรองไว้สำหรับเธรดที่กำลังพักอยู่
ค่า: ดีฟอลต์: 1:12 ช่วง: ค่าบวกสองค่า (*k:p*) โดยที่ k คือจำนวนของเคอร์นัลเธรด ที่ควรจะเก็บสำรองไว้สำหรับ pthreads ที่กำลังพักอยู่ จำนวน p
การแสดงผล: **echo $AIXTHREAD_SLPRATIO**

ค่านี้ เปิดขึ้นภายในระบบ ดังนั้นจะไม่เห็นค่าดีฟอลต์แรกเริ่ม เมื่อใช้คำสั่ง **echo**
การเปลี่ยนแปลง: **AIXTHREAD_SLPRATIO=***k:p***export AIXTHREAD_SLPRATIO**

การเปลี่ยนแปลงมีผลทันทีใน shell นี้ การเปลี่ยนแปลง มีผลบังคับใช้จนกระทั่งล็อกเอาต์จาก shell นี้ การเปลี่ยนแปลงถาวร มีการทำโดยการเพิ่มคำสั่ง **AIXTHREAD_SLPRATIO=***k:p* ลงในไฟล์ /etc/environment
การวิเคราะห์: N/A
การปรับ: โดยทั่วไป การสนับสนุน pthreads ที่กำลังพักอยู่ไม่ต้องการเคอร์นัลเธรดมากนัก เนื่องจากโดยทั่วไปแล้ว จะมีการตื่น ทีละหนึ่งรายการ ลักษณะเช่นนี้ช่วยประหยัดรีซอร์สเคอร์นัล

อ้างอิง ถึง: "ตัวแปรสำหรับขอบเขตการช่วงชิงทั้งกระบวนการ" ในหน้า 83

15. AIXTHREAD_STK=n

ไอเท็ม **Descriptor**
วัตถุประสงค์: จำนวนทศนิยมของไบต์ ที่ควรจะจัดสรรสำหรับแต่ละเธรด ค่านี้อาจถูกแทนที่โดยรูทีน **pthread_attr_setstacksize**
ค่า: ดีฟอลต์: 98 304 ไบต์ สำหรับแอ็พพลิเคชัน 32 บิต, 196 608 ไบต์สำหรับแอ็พพลิเคชัน 64 บิต ช่วง: ค่าจำนวนเต็มทศนิยมตั้งแต่ 0 ถึง 268 435 455 ซึ่งควรจะปัดเศษขึ้น เป็นหน้าที่ใกล้เคียงที่สุด (ปัจจุบัน 4 096)
การแสดงผล: **echo $AIXTHREAD_STK**

ค่านี้ เปิดขึ้นภายในระบบ ดังนั้นจะไม่เห็นค่าดีฟอลต์แรกเริ่ม เมื่อใช้คำสั่ง echo
การเปลี่ยนแปลง: **AIXTHREAD_STK=sizeexport AIXTHREAD_STK**

การเปลี่ยนแปลงมีผลทันทีใน shell นี้ การเปลี่ยนแปลง มีผลบังคับใช้จนกระทั่งล็อกเอาต์จาก shell นี้ การเปลี่ยนแปลงถาวร มีการทำโดยการเพิ่มคำสั่ง **AIXTHREAD_STK=size** ลงในไฟล์ /etc/environment
การวิเคราะห์: ถ้าการวิเคราะห์โปรแกรม ที่ล้มเหลวบ่งชี้สแต็คโอเวอร์โฟลว์ สามารถเพิ่มขนาดสแต็คดีฟอลต์ได้
การปรับ: ถ้าพยายามใช้งานจนถึงขีดจำกัดเธรด 32 000 บนแอ็พพลิเคชัน 32 บิต อาจจำเป็นต้อง ลดขนาดสแต็คดีฟอลต์

16. AIXTHREAD_AFFINITY

ไอเท็ม **Descriptor**
วัตถุประสงค์: ควบคุมการวางตำแหน่ง ของโครงสร้าง pthread, สแต็ก, และหน่วยเก็บ thread-local บนระบบที่ เปิดใช้งานความสัมพันธ์แบบเสริม ศักยภาพ
ค่า: ดีฟอลต์: ที่มีอยู่ ช่วง: ที่มีอยู่ เสมอ ความพยายาม
การแสดงผล: **echo $AIXTHREAD_AFFINITY**

ค่านี้ เปิดขึ้นภายในระบบ ดังนั้นจะไม่เห็นค่าดีฟอลต์แรกเริ่ม เมื่อใช้คำสั่ง **echo**
การเปลี่ยนแปลง: **AIXTHREAD_AFFINITY** ={default|strict|first-touch} **export**

AIXTHREAD_AFFINITY
การวิเคราะห์: การตั้งค่าตัวแปร เป็น **strict** จะช่วยพัฒนาประสิทธิภาพของเธรด อย่างไรก็ตาม อาจต้องใช้เวลาสตาร์ทอัพนานขึ้น

การตั้งค่าตัวแปร เป็น **default** จะรักษาการนำไปใช้งานที่สมดุลก่อนหน้านี้

การตั้งค่า ตัวแปรเป็น **first-touch** จะปรับความสมดุลระหว่างเวลาสตาร์ทอัพ ที่นานขึ้นกับผลประโยชน์รันไทม์
การปรับ: หากคาดว่าเธรดจะรัน ยาวนาน การตั้งค่าตัวแปรเป็น **strict** จะช่วย ปรับปรุงประสิทธิภาพ อย่างไรก็ตาม สำหรับเธรดจำนวนมากที่รัน ไม่นานนัก ควรตั้งค่าตัวแปรเป็น **default** หรือ **first touch** อย่างใดอย่างหนึ่ง

อ้างอิง ถึง: “ตัวแปรสภาพแวดล้อมเธรด” ในหน้า 78

17. MALLOCBUCKETS

ไอเท็ม **Descriptor**
วัตถุประสงค์: เปิดใช้งานส่วนขยายบน buckets ในตัวจัดสรรหน่วยความจำดีฟอลต์ที่อาจช่วยปรับปรุงประสิทธิภาพ ของแอ็พพลิเคชัน ซึ่งออกใช้การร้องขอการจัดสรรขนาดเล็กจำนวนมาก
ค่า: **MALLOCTYPE**=buckets

MALLOCBUCKETS=[[**number_of_buckets**:*n* | **bucket_sizing_factor**:*n* | **blocks_per_bucket**:*n* | **bucket_statistics**:[**stdout**|**stderr**|**pathname**]],...]

ตาราง ต่อไปนี้แสดงค่าดีฟอลต์ของ **MALLOCBUCKETS**

อ็อพชัน MALLOCBUCKETS
ค่าดีฟอลต์

number_of_buckets[1]
16

bucket_sizing_factor (32 บิต)[2]
32

bucket_sizing_factor (64 บิต)[3]
64

blocks_per_bucket
1024[4]

หมายเหตุ:

1 ค่าต่ำสุดที่ใช้ได้คือ 1 ค่าสูงสุดที่ใช้ได้ คือ 128

2 สำหรับการนำไปใช้แบบ 32 บิต ค่าที่ระบุ สำหรับ bucket_sizing_factor ต้องเป็นผลคูณของ 8

3 สำหรับการนำไปใช้แบบ 64 บิต ค่าที่ระบุ สำหรับ bucket_sizing_factor ต้องเป็นผลคูณของ 16

4. อ็อพชัน **bucket_statistics** ปิดใช้งานโดยค่าดีฟอลต์
การแสดงผล: **echo $MALLOCBUCKETS; echo $MALLOCTYPE**

ไอเท็ม **Descriptor**
การเปลี่ยนแปลง: ใช้วิธีการเฉพาะ shell ของการส่งออกตัวแปรสภาพแวดล้อม
การวิเคราะห์: ถ้าประสิทธิภาพ malloc ช้าและมีการออกใช้การร้องขอ malloc ขนาดเล็กจำนวนมาก คุณลักษณะนี้อาจช่วย ปรับปรุงประสิทธิภาพ
การปรับ: เมื่อต้องการเปิดใช้งาน malloc buckets ตัวแปรสภาพแวดล้อม *MALLOCTYPE* ต้องมีการตั้งค่า เป็นค่า "buckets"

ตัวแปรสภาพแวดล้อม *MALLOCBUCKETS* สามารถใช้เพื่อเปลี่ยนการตั้งค่าคอนฟิกดีฟอลต์ของ malloc buckets ได้ แม้ว่าค่าดีฟอลต์จะเพียงพอแล้วสำหรับแอ็พพลิเคชัน ส่วนใหญ่

อ็อพชัน **number_of_buckets:n** สามารถใช้เพื่อระบุ จำนวนของ buckets ที่มีอยู่ต่อ heap โดยที่ *n* คือ จำนวนของ buckets ค่าที่ระบุสำหรับ *n* จะใช้กับ heaps ที่มีอยู่ทั้งหมด

อ็อพชัน **bucket_sizing_factor:n** สามารถใช้ เพื่อระบุปัจจัยการกำหนดขนาด bucket โดยที่ *n* คือ ปัจจัยการกำหนดขนาด bucket ในหน่วยไบต์

อ็อพชัน **blocks_per_bucket:n** สามารถใช้เพื่อระบุ จำนวนบล็อกที่มีอยู่ในแต่ละ bucket ตั้งแต่แรกเริ่ม โดยที่ *n* คือ จำนวนบล็อก ค่านี้ใช้กับ buckets ทั้งหมด ค่า *n* ยังใช้เพื่อกำหนดจำนวน บล็อกที่จะเพิ่มโดยอัตโนมัติเมื่อ bucket ขยายขึ้น เนื่องจากบล็อกทั้งหมดถูกจัดสรรไปแล้ว

อ็อพชัน **bucket_statistics** จะทำให้ ระบบย่อย malloc ส่งเอาต์พุตสรุปข้อมูลสถิติเกี่ยวกับ malloc buckets ในทันทีที่แต่ละกระบวนการที่เรียกระบบย่อย malloc สิ้นสุดลงในขณะที่เปิดใช้งาน malloc buckets ข้อมูลสรุปนี้จะแสดงข้อมูล การตั้งค่าคอนฟิก buckets และจำนวนของการร้องขอการจัดสรรที่ประมวลผลแล้ว สำหรับแต่ละ bucket ถ้ามีการเปิดใช้งานหลาย heaps ด้วยวิธี malloc multiheap จำนวนของการร้องขอการจัดสรรที่แสดง สำหรับแต่ละ bucket จะเป็นผลรวมของการร้องขอการจัดส่งทั้งหมดที่ประมวลผล สำหรับ bucket นั้นสำหรับ heaps ทั้งหมด

สรุปสถิติ buckets จะถูกบันทึกลงใน ปลายทางเอาต์พุตอย่างใดอย่างหนึ่งดังต่อไปนี้ ตามที่ระบุ ด้วยอ็อพชัน **bucket_statistics**

stdout เอาต์พุตมาตรฐาน

stderr ข้อผิดพลาดมาตรฐาน

ชื่อพาธ ชื่อพาธที่ผู้ใช้ระบุ

ถ้ามีการระบุชื่อพาธที่ผู้ใช้ระบุ เอาต์พุตสถิติ จะถูกผนวกกับเนื้อหาที่มีอยู่ของไฟล์ (ถ้ามี) หลีกเลี่ยงการใช้เอาต์พุตมาตรฐานเป็นปลายทางเอาต์พุตสำหรับกระบวนการ ที่เป็นเจ้าของเอาต์พุตซึ่งไปป์เป็นอินพุตเข้าในกระบวนการอื่น

อ้างถึง: Malloc Buckets

18. MALLOCMULTIHEAP

ไอเท็ม **Descriptor**
วัตถุประสงค์: ควบคุมจำนวนของ heaps ภายในเซกเมนต์ส่วนตัวของกระบวนการ
ค่า: ดีฟอลต์: 32 ช่วง: ตัวเลขบวกระหว่าง 1 ถึง 32
การแสดงผล: **echo $MALLOCMULTIHEAP**

ค่านี้ เปิดขึ้นภายในระบบ ดังนั้นจะไม่เห็นค่าดีฟอลต์แรกเริ่ม เมื่อใช้คำสั่ง **echo**
การเปลี่ยนแปลง: **MALLOCMULTIHEAP=**[[**heaps:***n* | **considersize**],...] **export MALLOCMULTIHEAP**

การเปลี่ยนแปลงมีผลทันทีใน shell นี้ การเปลี่ยนแปลง มีผลบังคับใช้จนกระทั่งล็อกเอาต์จาก shell นี้ การเปลี่ยนแปลงถาวร มีการทำโดยการเพิ่มคำสั่ง **MALLOCMULTIHEAP=**[[**heaps:***n* | **considersize**],...] ลงในไฟล์ `/etc/environment`
การวิเคราะห์: ค้นหาการช่วงชิงล็อก บน malloc lock (ที่ตั้งอยู่ในเซกเมนต์ F) หรือน้อยกว่าเธรดที่รันได้ซึ่ง คาดไว้

ไอเท็ม	Descriptor
การปรับ:	จำนวน heaps น้อยลง สามารถช่วยลดขนาดของกระบวนการได้ กระบวนการผู้ใช้แบบมัลติเธรด บางอย่างที่ใช้ระบบย่อย malloc อย่างหนักอาจมีประสิทธิภาพดีขึ้น โดยการส่งออกตัวแปรสภาพแวดล้อม **MALLOCMULTIHEAP**=1 ก่อนที่จะเริ่มต้นแอ็พพลิเคชัน การพัฒนาประสิทธิภาพอาจเกิดขึ้นได้ มากเป็นพิเศษสำหรับโปรแกรม multithreaded C++ เนื่องจากโปรแกรมชนิดนี้ใช้ระบบย่อย malloc ในทุกครั้งที่เรียกตัวสร้าง หรือตัวทำลาย การพัฒนาประสิทธิภาพที่มีอยู่ใดๆ จะเห็นผลได้ชัดเจนที่สุดเมื่อกระบวนการผู้ใช้แบบมัลติเธรดกำลังรันอยู่บนระบบ SMP และโดยเฉพาะเมื่อใช้เธรดขอบเขตระบบ (อัตราส่วน M:N เป็น 1:1) อย่างไรก็ตาม ในบางกรณี การพัฒนาประสิทธิภาพยังอาจเห็นได้ชัดภายใต้สภาพอื่น และบนระบบแบบตัวประมวลผลเดียวด้วย ถ้ามีการระบุ อ็อพชัน **considersize** ขั้นตอนวิธีการเลือก heap อื่นที่ใช้ คือพยายามเลือก heap ที่มีอยู่ซึ่งมีพื้นที่ว่างเพียงพอที่จะ จัดการกับการร้องขอ การทำเช่นนี้อาจลดขนาดชุดทำงานของ กระบวนการโดยการลดจำนวนของการเรียก **sbrk( )** อย่างไรก็ตาม ขั้นตอนวิธีนี้อาจใช้เวลาการประมวลผลนานขึ้น

อ้างอิง ถึง: “ตัวแปรสภาพแวดล้อมเธรด” ในหน้า 78

19. NUM_RUNQ

ไอเท็ม	Descriptor
วัตถุประสงค์:	เปลี่ยนจำนวนของ จำนวนดีฟอลต์ของรันคิว
ค่า:	ดีฟอลต์: จำนวนของตัวประมวลผลที่ใช้งานอยู่ ซึ่งพบในขณะรัน ช่วง: จำนวนเต็มบวก
การแสดงผล:	**echo $NUM_RUNQ** ค่านี้ เปิดขึ้นภายในระบบ ดังนั้นจะไม่เห็นค่าดีฟอลต์แรกเริ่ม เมื่อใช้คำสั่ง **echo**
การเปลี่ยนแปลง:	**NUM_RUNQ=n export NUM_RUNQ** การเปลี่ยนแปลงมีผลทันทีใน shell นี้ การเปลี่ยนแปลง มีผลบังคับใช้จนกระทั่งล็อกเอาต์จาก shell นี้ การเปลี่ยนแปลงถาวรมีการทำโดยการเพิ่มคำสั่ง **NUM_RUNQ=***n* ลงในไฟล์ `/etc/environment`
การวิเคราะห์:	N/A
การปรับ:	N/A

อ้างอิง ถึง: “ตัวแปรสภาพแวดล้อมเธรด” ในหน้า 78

20. NUM_SPAREVP

ไอเท็ม	Descriptor
วัตถุประสงค์:	ตั้งค่าจำนวนของโครงสร้าง vp ที่จะถูก malloc’d ในระหว่างเวลา `pth_init`
ค่า:	ดีฟอลต์: NUM_SPARE_VP ช่วง: จำนวนเต็มบวก
การแสดงผล:	**echo $NUM_SPAREVP** ค่านี้ เปิดขึ้นภายในระบบ ดังนั้นจะไม่เห็นค่าดีฟอลต์แรกเริ่ม เมื่อใช้คำสั่ง **echo**
การเปลี่ยนแปลง:	**NUM_SPAREVP=n export NUM_SPAREVP** การเปลี่ยนแปลงจะมีผลบังคับใช้โดยทันทีใน shell นี้ การเปลี่ยนแปลง มีผลบังคับใช้จนกระทั่งล็อกเอาต์จาก shell นี้ การเปลี่ยนแปลงถาวรมีการทำ โดยการเพิ่มคำสั่ง **NUM_SPAREVP=***n* ลงในไฟล์ `/etc/environment`
การวิเคราะห์:	N/A
การปรับ:	N/A

อ้างอิง ถึง: “ตัวแปรสภาพแวดล้อมเธรด” ในหน้า 78

21. SPINLOOPTIME

ไอเท็ม **Descriptor**
วัตถุประสงค์: ควบคุมจำนวนครั้งในการ ลองล็อกที่ยุ่งใหม่ก่อนให้ตัวประมวลผลอื่น (เฉพาะ สำหรับ libpthreads เท่านั้น)
ค่า: ดีฟอลต์: 1 บนตัวประมวลผลเดียว, 40 บนหลายตัวประมวลผล ช่วง: จำนวนเต็มบวก
การแสดงผล: **echo $SPINLOOPTIME**

ค่านี้ เปิดขึ้นภายในระบบ ดังนั้นจะไม่เห็นค่าดีฟอลต์แรกเริ่ม เมื่อใช้คำสั่ง **echo**
การเปลี่ยนแปลง: **SPINLOOPTIME=*n*export SPINLOOPTIME**

การเปลี่ยนแปลงมีผลทันทีใน shell นี้ การเปลี่ยนแปลง มีผลบังคับใช้จนกระทั่งล็อกเอาต์จาก shell นี้ การเปลี่ยนแปลงถาวร มีการทำโดยการเพิ่มคำสั่ง **SPINLOOPTIME**=*n* ลงในไฟล์ /etc/environment
การวิเคราะห์: ถ้าเธรดจะพักบ่อย (เวลา idle สูง) แสดงว่า **SPINLOOPTIME** อาจ ไม่สูงเพียงพอ
การปรับ: การเพิ่มค่าจากค่าดีฟอลต์ 40 บนระบบหลายตัวประมวลผลอาจเป็นประโยชน์ ถ้ามีการช่วงชิง pthread mutex

อ้างอิง ถึง: "ตัวแปรสภาพแวดล้อมเธรด" ในหน้า 78

22. STEP_TIME

ไอเท็ม **Descriptor**
วัตถุประสงค์: ปรับจำนวนครั้ง ที่ใช้ในการสร้าง VP ในระหว่างไทม์เอาต์การเรียกใช้
ค่า: ดีฟอลต์: DEF_STEPTIME ช่วง: จำนวนเต็มบวก
การแสดงผล: **echo $STEP_TIME**

ค่านี้ เปิดขึ้นภายในระบบ ดังนั้นจะไม่เห็นค่าดีฟอลต์แรกเริ่ม เมื่อใช้คำสั่ง **echo**
การเปลี่ยนแปลง: **STEP_TIME=n export STEP_TIME**

การเปลี่ยนแปลงจะมีผลบังคับใช้โดยทันทีใน shell นี้ การเปลี่ยนแปลง มีผลบังคับใช้จนกระทั่งล็อกเอาต์จาก shell นี้ การเปลี่ยนแปลงถาวรมีการทำ โดยการเพิ่มคำสั่ง **STEP_TIME**=*n* ลงในไฟล์ /etc/environment
การวิเคราะห์: N/A
การปรับ: N/A

อ้างอิง ถึง: "ตัวแปรสภาพแวดล้อมเธรด" ในหน้า 78

23. VP_STEALMAX

ไอเท็ม **Descriptor**
วัตถุประสงค์: ปรับจำนวนของ VPs ที่สามารถถูกขโมยหรือปิดการขโมย VP
ค่า: ดีฟอลต์: ไม่มี ช่วง: จำนวนเต็มบวก
การแสดงผล: **echo $VP_STEALMAX**

ค่านี้ เปิดขึ้นภายในระบบ ดังนั้นจะไม่เห็นค่าดีฟอลต์แรกเริ่ม เมื่อใช้คำสั่ง **echo**
การเปลี่ยนแปลง: **VP_STEALMAX=n export VP_STEALMAX**

การเปลี่ยนแปลงจะมีผลบังคับใช้โดยทันทีใน shell นี้ การเปลี่ยนแปลง มีผลบังคับใช้จนกระทั่งล็อกเอาต์จาก shell นี้ การเปลี่ยนแปลงถาวรมีการทำ โดยการเพิ่มคำสั่ง **VP_STEALMAX**=*n* ลงในไฟล์ /etc/environment
การวิเคราะห์: N/A
การปรับ: N/A

อ้างอิง ถึง: "ตัวแปรสภาพแวดล้อมเธรด" ในหน้า 78

24. YIELDLOOPTIME

ไอเท็ม	Descriptor
วัตถุประสงค์:	ควบคุมจำนวนครั้ง ในการให้ตัวประมวลผลก่อนที่จะบล็อกล็อกที่ยุ่ง (สำหรับ libpthreads เท่านั้น) ตัวประมวลผลมีการให้ไปยังเคอร์นัลเธรดอื่น โดยสมมุติว่ามีเคอร์นัลเธรดที่ดำเนินการได้อื่นซึ่งมีระดับความสำคัญสูงเพียงพอ
ค่า:	ดีฟอลต์: 0 ช่วง: ค่าจำนวนเต็ม บวก
การแสดงผล:	**echo $YIELDLOOPTIME** ค่านี้ เปิดขึ้นภายในระบบ ดังนั้นจะไม่เห็นค่าดีฟอลต์แรกเริ่ม เมื่อใช้คำสั่ง **echo**
การเปลี่ยนแปลง:	**YIELDLOOPTIME=*n*export YIELDLOOPTIME** การเปลี่ยนแปลงมีผลทันทีใน shell นี้ การเปลี่ยนแปลง มีผลบังคับใช้จนกระทั่งล็อกเอาต์จาก shell นี้ การเปลี่ยนแปลงถาวร มีการทำโดยการเพิ่มคำสั่ง **YIELDLOOPTIME=*n*** ลงในไฟล์ /etc/environment
การวิเคราะห์:	ถ้าเธรดจะพักบ่อย (เวลา idle สูง) แสดงว่า **YIELDLOOPTIME** อาจ ไม่สูงเพียงพอ
การปรับ:	การเพิ่มค่าจากค่าดีฟอลต์ 0 อาจเป็นประโยชน์ ถ้าคุณไม่ต้องการให้เธรดไปพัก ในขณะที่เธรดกำลังรอล็อกอยู่

อ้างอิง ถึง: “ตัวแปรสภาพแวดล้อมเธรด” ในหน้า 78

พารามิเตอร์ที่สามารถปรับแต่งได้เบ็ดเตล็ด

พารามิเตอร์เบ็ดเตล็ดทั้งหลายที่มีอยู่ใน AIX สามารถปรับแต่งได้

1. **AIX_TZCACHE**

ไอเท็ม	Descriptor
วัตถุประสงค์:	เก็บสำเนาคงที่ของตัวแปร *TZ* สำหรับความยาวของกระบวนการ
ค่า:	ดีฟอลต์: ไม่ได้ตั้งค่าไว้ ค่าที่เป็นไปได้คือ: ON (เปิดใช้งาน พารามิเตอร์)
การแสดง:	**$AIX_TZCACHE**
การเปลี่ยน:	**export AIX_TZCACHE=ON** การเปลี่ยนแปลงมีผลบังคับใช้สำหรับทุกกระบวนการที่เริ่มต้นจากเชลล์นี้ในภายหลัง โดยแจ้งให้แอ็พพลิเคชันทราบเสมอ เมื่อใช้ค่าเริ่มต้นของตัวแปร *TZ* กระบวนการนี้ช่วยปรับปรุงประสิทธิภาพการทำงาน หากการค้นหาเขตเวลาถูกเรียกโดยแอ็พพลิเคชัน ตัวอย่างเช่น หากแอ็พพลิเคชันตรวจสอบเวลาท้องถิ่น บ่อยขึ้น อย่างไรก็ตาม การเปลี่ยนแปลงใดๆ ที่เกิดขึ้นกับตัวแปร *TZ* จะไม่ถูกจดจำไว้เมื่อแอ็พพลิเคชันเริ่มทำงานแล้ว
การวินิจฉัย:	พารามิเตอร์นี้ไม่แนะนำให้ใช้สำหรับคอนฟิกูเรชันในระบบสากล ในไฟล์ /etc/environment ใช้พารามิเตอร์นี้สำหรับแอ็พพลิเคชันที่ไม่เปลี่ยนตัวแปร *TZ* แต่สร้างคำร้องขอเขตเวลา
การปรับ:	N/A

2. **EXTSHM**

ไอเท็ม	Descriptor
วัตถุประสงค์:	เปิดใช้ตัวช่วยหน่วยความจำที่แบ่งใช้ซึ่งขยายเพิ่ม
ค่า:	ดีฟอลต์: ไม่ได้ตั้งค่าไว้ ค่าที่เป็นไปได้: ON, 1SEG, MSEG
การแสดง:	**echo $EXTSHM**
การเปลี่ยน:	**export EXTSHM** การเปลี่ยนแปลงจะมีผลบังคับใช้โดยทันทีใน shell นี้ การเปลี่ยนแปลงจะมีผลบังคับใช้จนกระทั่งล็อกเอาต์ออกจาก shell นี้ การเปลี่ยนแปลงแบบถาวรจะกระทำโดยการเพิ่มคำสั่ง **EXTSHM=ON** , **EXTSHM=1SEG** หรือ **EXTSHM=MSEG** ลงในไฟล์ /etc/environment
การวินิจฉัย:	N/A

ไอเท็ม	Descriptor
การปรับ:	การตั้งค่าเป็น *ON*, *1SEG* หรือ *MSEG* จะทำให้กระบวนการจัดสรรเซกเมนต์หน่วยความจำแบบแบ่งใช้ขนาดเล็กที่ที่สุด 1 ไบต์ สำหรับเพจที่ใกล้ที่สุด ซึ่งจะลบข้อจำกัดของเซกเมนต์หน่วยความจำแบบแบ่งใช้ของผู้ใช้ 11 เซกเมนต์อย่างมีประสิทธิภาพ สำหรับกระบวนการแบบ 32 บิต ขนาดสูงสุดของเซกเมนต์หน่วยความจำทั้งหมด คือ 2.75 GB การตั้งค่า **EXTSHM** ให้มีค่าเป็น *ON* มีผลกระทบแบบเดียวกับ การตั้งค่าตัวแปรให้มีค่า *1SEG* ด้วยค่าติดตั้งอย่างใดอย่างหนึ่งนี้ หน่วยความจำที่แบ่งใช้ใดๆ ที่มีขนาดน้อยกว่า 256 MB จะถูกสร้างขึ้นเป็นเซ็กเมนต์ mmap อยู่ภายใน และมีผลการทำงานแบบเดียวกับ mmap ที่เกี่ยวข้อง หน่วยความจำที่แบ่งใช้ใดๆ ที่มีขนาดมากกว่าหรือเท่ากับ 256 MB เซ็กเมนต์การทำงานภายใน ถ้า **EXTSHM** มีค่าเป็น *MSEG* หน่วยความจำที่แบ่งใช้ทั้งหมดจะถูกสร้างขึ้นเป็นเซ็กเมนต์ mmap อยู่ภายใน ซึ่งอนุญาตสำหรับการใช้ประโยชน์จากหน่วยความจำได้ดีขึ้นกว่าเดิม

โปรดอ้างถึง: “หน่วยความจำที่แบ่งใช้ซึ่งขยายเพิ่ม” ในหน้า 175

3. **LDR_CNTRL**

ไอเท็ม	Descriptor
วัตถุประสงค์:	อนุญาตให้ปรับโหลดเดอร์เคอร์เนล
ค่า:	ดีฟอลต์: ไม่ได้ตั้งค่าไว้ ค่าที่ใช้ได้: PREREAD_SHLIB, LOADPUBLIC, IGNOREUNLOAD, USERREGS, MAXDATA, MAXDATA32, MAXDATA64, DSA, PRIVSEG_LOADS, DATA_START_STAGGER, LARGE_PAGE_TEXT, LARGE_PAGE_DATA, HUGE_EXEC, NAMEDSHLIB, SHARED_SYMTAB หรือ SED
การแสดง:	**echo $LDR_CNTRL**
การเปลี่ยน:	**LDR_CNTRL=**{*PREREAD_SHLIB*|*LOADPUBLIC*|...}**export LDR_CNTRL** การเปลี่ยนแปลงจะมีผลบังคับใช้โดยทันทีใน shell นี้ การเปลี่ยนแปลงจะมีผลบังคับใช้จนกระทั่งล็อกเอาต์ออกจาก shell นี้ การเปลี่ยนแปลงแบบถาวร สามารถกระทำได้โดยเพิ่มบรรทัดลงในไฟล์ `/etc/environment`: **LDR_CNTRL=**{*PREREAD_SHLIB*|*LOADPUBLIC*|...}
การวินิจฉัย:	N/A

ไอเท็มการปรับ: **Descriptor**

ตัวแปรสภาวะแวดล้อม **LDR_CNTRL** สามารถนำมาใช้เพื่อควบคุมลักษณะการทำงานของโหลดเดอร์ของระบบตั้งแต่หนึ่งลักษณะขึ้นไป คุณสามารถระบุอ็อพชันจำนวนมากได้ด้วยตัวแปร **LDR_CNTRL** เมื่อระบุอ็อพชัน ให้แยกอ็อพชันด้วยเครื่องหมาย '@' ตัวอย่างของการระบุอ็อพชันจำนวนมากคือ : **LDR_CNTRL**=PREREAD_SHLIB@LOADPUBLIC การระบุอ็อพชัน PREREAD_SHLIB ทำให้ทั้งไลบรารีถูกอ่านทันทีที่ไลบรารีถูกเข้าถึง โดยใช้อ็อพชัน VMM ไลบรารีสามารถอ่านได้จากดิสก์และสามารถถูกแคชในหน่วยความจำก่อนที่โปรแกรมจะเริ่มเข้าถึงเพจ ขณะที่เมธอดนี้อาจใช้หน่วยความจำเพิ่มขึ้น ซึ่งยังอาจช่วยปรับปรุงผลการทำงานของโปรแกรมที่ใช้เพจไลบรารีที่แบ่งใช้จำนวนมาก หากรูปแบบการเข้าถึงไม่เป็นไปตามลำดับ (ตัวอย่างเช่น Catia)

การระบุอ็อพชัน LOADPUBLIC จะสั่งให้โหลดเดอร์ของระบบโหลดโมดูลทั้งหมดที่ร้องขอโดยแอ็พพลิเคชัน ลงในเซ็กเมนต์ไลบรารีที่แบ่งใช้แบบโกลบอล ถ้าโมดูลไม่สามารถโหลดในที่สาธารณะลงใน เซ็กเมนต์ไลบรารีที่แบ่งใช้แบบโกลบอล โมดูลนั้นจะโหลดอย่างเป็นส่วนตัว สำหรับแอ็พพลิเคชันนั้น

การระบุอ็อพชัน IGNOREUNLOAD จะป้องกันแอ็พพลิเคชัน จากการถอนการโหลดไลบรารี ข้อจำกัดนี้อาจป้องกันไม่ให้การการแตกแฟรกเมนต์หน่วยความจำและกำจัดโอเวอร์เฮดที่เกิดขึ้นเมื่อไลบรารีถูกโหลดและถูกยกเลิกการโหลดซ้ำๆ ถ้าคุณไม่ได้ระบุอ็อพชัน IGNOREUNLOAD ไว้ คุณอาจจบการทำงานด้วยสองอินสแตนซ์ข้อมูลของโมดูลหากคุณโหลดโมดูลในเวลาเดียวกับโหลดแอ็พพลิเคชัน และมีคำร้องขอให้โหลดหรือถอนการโหลดโมดูลแบบไดนามิก หลายครั้ง

การระบุอ็อพชัน USERREGS จะบอกให้ระบบบันทึกการเรียกใช้ระบบการลงทะเบียนผู้ใช้ทั่วไปทั้งหมดที่ทำโดยแอ็พพลิเคชัน ซึ่งจะมีประโยชน์ต่อแอ็พพลิเคชันในการเก็บรวบรวมขยะ

การระบุอ็อพชัน MAXDATA จะตั้งค่าขนาดของฮีปสูงสุดสำหรับกระบวนการ ซึ่งรวมถึงการแทนที่ค่า maxdata ใดๆ ที่ระบุในไฟล์ที่สามารถเรียกทำงาน ค่า maxdata ใช้เพื่อตั้งค่าขีดจำกัดรีซอร์สข้อมูลแบบซอร์ฟเริ่มต้นของกระบวนการ สำหรับโปรแกรมแบบ 32 บิต ค่า maxdata ที่ไม่ใช่ศูนย์จะเปิดใช้งานโหมดเพื่อที่แอดเดรสขนาดใหญ่ โปรดดูที่ การสนับสนุนโปรแกรมขนาดใหญ่ เมื่อต้องการเปิดใช้งานโมเดลพื้นที่แอดเดรสขนาดใหญ่ ให้ระบุค่า maxdata เป็นศูนย์ โดยการตั้งค่า **LDR_CNTRL=MAXDATA=0** สำหรับโปรแกรมแบบ 64 บิต ค่า maxdata จะรับประกันขนาดสูงสุดของฮีปข้อมูลของโปรแกรม ส่วนของพื้นที่แอดเดรสที่สำรองไว้สำหรับฮีปจะไม่สามารถใช้ได้โดยรูทีนย่อย **shmat( )** หรือ **mmap( )** ยกเว้นมีการระบุแอดเดรสที่ชัดเจน คุณสามารถระบุค่าใดๆ ได้ แต่พื้นที่ข้อมูลไม่สามารถขยายกลับไปเป็น 0x06FFFFFFFFFFFFFF โดยไม่ได้พิจารณาถึงค่า maxdata ที่ระบุไว้

มีอ็อพชัน maxdata เพิ่มเติมสองอ็อพชันเพื่อการควบคุมที่ละเอียดขึ้น โดยขึ้นอยู่กับว่าอ็อพชันเป็นแบบ 32 บิต หรือ 64 บิต อ็อพชัน maxdata เพิ่มเติมเหล่านี้จะแทนที่อ็อพชัน MAXDATA สำหรับโหมดอ็อบเจ็กต์ที่สอดคล้องกัน การระบุอ็อพชัน MAXDATA32 จะมีผลทำให้มีลักษณะการทำงานที่เหมือนกับ MAXDATA ยกเว้นค่าจะถูกข้ามสำหรับกระบวนการแบบ 64 บิต การระบุอ็อพชัน MAXDATA64 จะมีผลทำให้มีลักษณะการทำงานที่เหมือนกับ MAXDATA ยกเว้นค่าจะถูกข้ามสำหรับกระบวนการแบบ 32 บิต

การระบุอ็อพชัน PRIVSEG_LOADS จะเป็นการสั่งให้โหลดเดอร์ของระบบวางโมดูลส่วนตัวที่โหลดไว้แบบไดนามิก ลงในการประมวลผลเซ็กเมนต์ส่วนตัว ข้อกำหนดคุณลักษณะนี้อาจเพิ่มความพร้อมใช้งานของหน่วยความจำในแอ็พพลิเคชันที่ใช้โมเดลหน่วยความจำขนาดใหญ่ที่ดำเนินการโหลดแบบไดนามิกแบบไพรเวทและมีแนวโน้มที่หน่วยความจำจะไม่เพียงพอในฮีปของกระบวนการ ถ้าการประมวลผลเซ็กเมนต์ส่วนตัวขาดพื้นที่ที่เพียงพอ อ็อพชัน PRIVSEG_LOADS จะไม่เกิดผลกระทบใดๆ อ็อพชัน PRIVSEG_LOADS จะใช้ได้กับแอ็พพลิเคชันแบบ 32 บิตเท่านั้น พร้อมกับค่า MAXDATA ที่ไม่ใช่ศูนย์

การระบุอ็อพชัน DATA_START_STAGGER=Y จะเริ่มทำงานจากส่วนของข้อมูลของการประมวลผลที่อ็อฟเซ็ตต่อ MCM ซึ่งควบคุมโดยอ็อพชัน **data_stagger_interval** ของคำสั่ง **vmo** การประมวลผลข้อมูลเพจขนาดใหญ่ nth ที่เรียกใช้งานบน MCM ที่ได้กำหนดไว้จะมีส่วนของข้อมูลที่เริ่มทำงานที่อ็อฟเซ็ต (n * **data_stagger_interval** * PAGESIZE) % 16 MB อ็อพชัน DATA_START_STAGGER=Y จะใช้ได้กับการประมวลผลแบบ 64 บิตบนเคอร์เนลแบบ 64 บิตเท่านั้น

การระบุอ็อพชัน LARGE_PAGE_TEXT=Y จะระบุว่าตัวโหลดอาจพยายามใช้เพจขนาดใหญ่สำหรับเซกเมนต์ข้อความของกระบวนการ อ็อพชัน LARGE_PAGE_TEXT=Y จะใช้ได้เฉพาะกับกระบวนการ 64 บิตบนเคอร์เนล 64 บิตเท่านั้น

ไอเท็ม	Descriptor

	การระบุอ็อพชัน LARGE_PAGE_DATA=M จะจัดสรรเพจขนาดใหญ่ที่เพียงพอสำหรับเซ็กเมนต์ข้อมูลที่มีค่า **brk**มากที่สุดเท่านั้น แทนที่จะเป็นเซ็กเมนต์ทั้งหมด ซึ่งคือลักษณะการทำงานเมื่อคุณไม่ได้ระบุอ็อพชัน LARGE_PAGE_DATA=M ไว้ การเปลี่ยนค่าไปเป็น **brk** อาจเกิดความล้มเหลว หากมีเพจขนาดใหญ่ที่ไม่เพียงพอต่อการสนับสนุนการเปลี่ยนค่าไปเป็น **brk**
	
	การระบุอ็อพชัน RESOLVEALL จะบังคับให้โหลดเดอร์แก้ปัญหาสัญลักษณ์ที่ไม่ได้นิยามไว้ทั้งหมด ซึ่งได้อิมพอร์ตไว้ในเวลาที่โหลดโปรแกรมหรือเมื่อโปรแกรมโหลดโมดูลแบบไดนามิก การแก้ปัญหาสัญลักษณ์จะถูกดำเนินการในลำดับที่ลึกลงเป็นอันดับแรกของ AIX ถ้าคุณระบุ **LDR_CNTRL**=RESOLVEALL และสัญลักษณ์ที่อิมพอร์ตไม่สามารถแก้ปัญหาได้ โปรแกรมหรือโมดูลแบบไดนามิกอาจเกิดความล้มเหลวในการโหลดได้
	
	การระบุอ็อพชัน HUGE_EXEC จะจัดเตรียมการควบคุมผู้ใช้ผ่านการประมวลผลตำแหน่งพื้นที่แอดเดรสของเซ็กเมนต์ แบบอ่านอย่างเดียวที่สามารถเรียกทำงานแบบ 32 บิตได้ สำหรับข้อมูลเพิ่มเติม โปรดดู ความสามารถในการเรียกทำงานแบบ 32 บิตที่มีขนาดใหญ่
	
	การระบุอ็อพชัน `NAMEDSHLIB=name,[attr1],[attr2]...[attrN]` ทำให้กระบวนการสามารถเข้าถึง หรือสร้างพื้นที่ไลบรารีแบบแบ่งใช้ที่บ่งชี้โดยชื่อที่ระบุ คุณสามารถสร้างพื้นที่ไลบรารีที่แบ่งใช้ซึ่งตั้งชื่อแล้ว ด้วยเมธอดดังต่อไปนี้:
	• ไม่ระบุแอ็ตทริบิวต์
	• ระบุแอ็ตทริบิวต์ **doubletext32** ที่สร้างพื้นที่ไลบรารีที่แบ่งใช้ซึ่งตั้งชื่อแล้ว พร้อมกับสองเซ็กเมนต์ที่ใช้เฉพาะกับข้อความไลบรารีที่แบ่งใช้
	
	ถ้าการประมวลผลร้องขอการใช้พื้นที่ไลบรารีที่แบ่งใช้ซึ่งตั้งชื่อแล้วและไม่มีอยู่ พื้นที่ไลบรารีที่แบ่งใช้จะถูกสร้างขึ้นโดยอัตโนมัติ พร้อมกับชื่อที่ระบุไว้ ถ้ามีการระบุชื่อที่ไม่ถูกต้อง อ็อพชัน `NAMEDSHLIB=name,[attr1],[attr2]...[attrN]` จะถูกข้าม ชื่อที่ถูกต้องจะมีความยาวเป็นค่าบวก และมีตัวอักษรผสมตัวเลข ขีดเส้นใต้ และอักขระจุด
	
	การระบุอ็อพชัน `SHARED_SYMTAB=Y` จะทำให้ระบบสร้างตารางสัญลักษณ์แบบแบ่งใช้สำหรับโปรแกรมแบบ 64 บิต ถ้าโปรแกรมเอ็กซ์พอร์ตสัญลักษณ์ใดๆ ถ้าหลายอินสแตนซ์ของโปรแกรมรันพร้อมกัน การใช้ตารางสัญลักษณ์แบบแบ่งใช้สามารถช่วยลดจำนวนของหน่วยความจำระบบที่โปรแกรมต้องการ
	
	การระบุอ็อพชัน `SHARED_SYMTAB=N` จะป้องกันไม่ให้ระบบสร้างตารางสัญลักษณ์แบบแบ่งใช้สำหรับโปรแกรมแบบ 64 บิต อ็อพชันนี้จะแทนที่แฟล็ก `AOUT_SHR_SYMTAB` ในส่วนหัว XCOFF สำรอง
	
	การระบุอ็อพชัน SED จะตั้งค่าโหมด stack execution disable (SED) สำหรับกระบวนการ โดยการข้ามโหมด SED อื่นที่ระบุโดยไฟล์ที่สามารถเรียกทำงาน อ็อพชันนี้ต้องถูกตั้งค่าเป็นค่าใดค่าหนึ่งต่อไปนี้:
	`SED=system`
	`SED=request`
	`SED=exempt`

4. LDR_PRELOAD LDR_PRELOAD64

ไอเท็ม	Descriptor
วัตถุประสงค์:	ร้องขอการโหลดล่วงหน้าของ ไลบรารีที่แบ่งใช้ อ็อพชัน LDR_PRELOAD คือการประมวลผลแบบ 32 บิต และอ็อพชัน LDR_PRELOAD64 คือการประมวลผลแบบ 64 บิต ในระหว่างการแก้ปัญหาสัญลักษณ์ ไลบรารีที่โหลดไว้ล่วงหน้า ซึ่งแสดงอยู่ในตัวแปรนี้จะค้นหาสัญลักษณ์ที่อิมพอร์ตเข้ามาเป็นอันดับแรก และเมื่อไม่พบอยู่ในไลบรารีเหล่านั้น ระบบจะใช้การค้นหาแบบปกติแทนเท่านั้น การครอบครองสัญลักษณ์จากไลบรารีที่โหลดล่วงหน้า จะทำงานสำหรับทั้งดีฟอลต์การลิงก์ AIX และการลิงก์ในแบบรันไทม์ การแก้ปัญหาสัญลักษณ์ที่เลื่อนออกไปจะไม่มีการเปลี่ยนแปลง
ค่า:	ดีฟอลต์: ไม่ได้ตั้งค่าไว้ ค่าที่เป็นไปได้คือ: ชื่อไลบรารี **หมายเหตุ:** ถ้ามีมากกว่าหนึ่งไลบรารีที่แสดงรายการ ให้คั่นชื่อเหล่านั้นด้วยเครื่องหมายโคลอน (:) ระบุสมาชิกของไลบรารีที่เก็บถาวรไว้ในเครื่องหมายวงเล็บ
การแสดง:	**echo $LDR_PRELOAD** **echo $LDR_PRELOAD64**

ไอเท็ม **Descriptor**
การเปลี่ยน: **$LDR_PRELOAD="libx.so:liby.a(shr.o)"**

แก้ปัญหาสัญลักษณ์ใดๆ ที่จำเป็นอันดับแรกจากอ็อบเจ็กต์ libx.so ที่แบ่งใช้ จากนั้น จาก shr.o ซึ่งเป็นสมาชิกของ liby.a และท้ายสุดภายในการพึ่งพาของการประมวลผลโมดูลที่โหลดแบบไดนามิกทั้งหมด (โมดูลที่โหลดด้วย dlopen() หรือ load()) จะยังคงถูกแก้ปัญหาเป็นอันดับแรกจากไลบรารีที่โหลดไว้ล่วงหน้า ซึ่งแสดงรายการด้วยตัวแปร
การวินิจฉัย: N/A

5. NODISCLAIM

ไอเท็ม **Descriptor**
วัตถุประสงค์: ควบคุมวิธีการเรียก **free()** ที่จะถูกจัดการ เมื่อตั้งค่า **PSALLOC** ให้มีค่า *early* การเรียก **free()** ทั้งหมดจะส่งผลทำให้ระบบเรียก **disclaim()** เมื่อตั้งค่า **NODISCLAIM** ให้มีค่า *true* เหตุการณ์นี้จะไม่เกิดขึ้น
ค่า: ดีฟอลต์: ไม่ได้ตั้งค่าไว้

ค่าที่เป็นไปได้: True
การแสดง: **echo $NODISCLAIM**
การเปลี่ยน: **NODISCLAIM=true export NODISCLAIM**

การเปลี่ยนแปลงจะมีผลบังคับใช้โดยทันทีใน shell นี้ การเปลี่ยนแปลงจะมีผลบังคับใช้จนกระทั่งล็อกเอาต์ออกจาก shell นี้ การเปลี่ยนแปลงแบบถาวรจะกระทำได้โดยการเพิ่มคำสั่ง **NODISCLAIM=true** ลงในไฟล์ /etc/environment
การวินิจฉัย: ถ้าจำนวนของระบบที่เรียก **disclaim()** มีจำนวนสูงมาก คุณอาจต้องการตั้งค่าตัวแปรนี้
การปรับ: การตั้งค่าตัวแปรนี้จะกำจัดการเรียกอ็อพชัน **disclaim()** ออกจาก **free()** หากตั้งค่า **PSALLOC** ให้มีค่า *early*

โปรดอ้างถึง: "การจัดสรรพื้นที่เพจก่อนหน้า" ในหน้า 171

6. NSORDER

ไอเท็ม **Descriptor**
วัตถุประสงค์: เขียนทับชุดการแก้ไขปัญหาเรื่องชื่อ ตามลำดับการค้นหา
ค่า: ดีฟอลต์: bind, nis, local

ค่าที่เป็นไปได้: bind, local, nis, bind4, bind6, local4, local6, nis4 หรือ nis6
การแสดง: **echo $NSORDER**

ค่านี้ เปิดขึ้นภายในระบบ ดังนั้นจะไม่เห็นค่าดีฟอลต์แรกเริ่ม เมื่อใช้คำสั่ง **echo**
การเปลี่ยน: **NSORDER=***value*, *value*, ... **export NSORDER**

การเปลี่ยนแปลงจะมีผลบังคับใช้โดยทันทีใน shell นี้ การเปลี่ยนแปลงจะมีผลบังคับใช้จนกระทั่งล็อกเอาต์ออกจาก shell นี้ การเปลี่ยนแปลงแบบถาวรจะกระทำได้โดยการเพิ่มคำสั่ง **NSORDER=***value* ลงในไฟล์ /etc/environment
การวินิจฉัย: N/A
การปรับ: **NSORDER** จะแทนที่ไฟล์ /etc/netsvc.conf

โปรดอ้างถึง: "การปรับการแก้ไขชื่อ" ในหน้า 324

7. PSALLOC

ไอเท็ม **Descriptor**
วัตถุประสงค์: ตั้งค่าตัวแปรสภาวะแวดล้อม **PSALLOC** เพื่อกำหนดนโยบายการจัดสรรพื้นที่การเพจ
ค่า: ดีฟอลต์: ไม่ได้ตั้งค่าไว้

ค่าที่เป็นไปได้: early
การแสดง: **echo $PSALLOC**
การเปลี่ยน: **PSALLOC=early export PSALLOC**

การเปลี่ยนแปลงจะมีผลบังคับใช้โดยทันทีใน shell นี้ การเปลี่ยนแปลงจะมีผลบังคับใช้ จนกระทั่งล็อกเอาต์ออกจาก shell นี้
การวินิจฉัย: N/A

ไอเท็ม **Descriptor**
การปรับ: เพื่อตรวจสอบให้แน่ใจว่า การประมวลผลไม่ได้ถูกหยุดทำงานเนื่องจากเงื่อนไขการเพจที่ต่ำ การประมวลผลนี้สามารถจัดสรรพื้นที่การเพจล่วงหน้าได้โดยการใช้นโยบาย การจัดสรรพื้นที่เพจล่วงหน้า อย่างไรก็ตาม การกระทำเช่นนี้ อาจส่งผลทำให้มีพื้นที่การเพจที่ไม่ได้ใช้ คุณยังอาจต้องการตั้งค่าตัวแปรสภาวะแวดล้อม **NODISCLAIM**

โปรดอ้างถึง: "การจัดสรรและการเรียกคืนสล็อตพื้นที่การเพจ" ในหน้า 58 และ "การจัดสรรพื้นที่เพจก่อนหน้า" ในหน้า 171

8. **RT_GRQ**

ไอเท็ม **Descriptor**
วัตถุประสงค์: เป็นสาเหตุทำให้เธรดถูกวางไว้ในคิวการทำงานแบบโกลบอล แทนที่คิวการทำงานแบบต่อ CPU
ค่า: ดีฟอลต์: ไม่ได้ตั้งค่าไว้

ช่วง: ON, OFF
การแสดง: **echo $RT_GRQ**
การเปลี่ยน: **RT_GRQ=***{OFF/ON}***export RT_GRQ**

การเปลี่ยนแปลงจะมีผลบังคับใช้โดยทันที การเปลี่ยนแปลงจะมีผลบังคับใช้จนกระทั่งบูตครั้งถัดไป การเปลี่ยนแปลงแบบถาวรจะกระทำได้โดยการเพิ่มคำสั่ง **RT_GRQ**=*{ON|OFF}* ลงในไฟล์ /etc/environment
การวินิจฉัย: N/A
การปรับ: อาจถูกปรับ สำหรับระบบแบบมัลติโพรเซสเซอร์ การตั้งค่าตัวแปรให้มีค่าเป็น *ON* จะเป็นสาเหตุทำให้เธรด ถูกวางอยู่ในคิวการทำงานแบบโกลบอล ในกรณีนี้ คิวการทำงานแบบโกลบอลจะถูกค้นหา เพื่อดูเธรดที่มีระดับความสำคัญที่ดีที่สุด ซึ่งอาจอนุญาตให้ระบบขอรับเธรดที่จัดส่งได้เร็วขึ้น และสามารถปรับปรุงผลการทำงานสำหรับเธรดที่กำลังรัน SCHED_OTHER และผลักดันให้มีการอินเตอร์รัปต์

โปรดอ้างถึง: "รันคิวของตัวจัดตารางเวลา" ในหน้า 48

9. **RT_MPC**

ไอเท็ม **Descriptor**
วัตถุประสงค์: เมื่อคุณกำลังรันเคอร์เนลในโหมดเรียลไทม์ (โปรดดูคำสั่ง **bosdebug**) MPC จะสามารถส่งไปยัง CPU อื่นๆ เพื่ออินเตอร์รัปต์ MPC นั้น หากเธรดที่มีระดับความสำคัญที่ดีกว่าสามารถรันได้ ดังนั้น เธรดนี้จะสามารถถูกจัดส่งได้โดยทันที
ค่า: ดีฟอลต์: ไม่ได้ตั้งค่าไว้

ช่วง: ON
การแสดง: **echo $RT_MPC**
การเปลี่ยน: **RT_MPC=ON export RT_MPC**

การเปลี่ยนแปลง จะมีผลบังคับใช้โดยทันที การเปลี่ยนแปลงจะมีผลบังคับใช้จนกระทั่งบูตในครั้งถัดไป การเปลี่ยนแปลงแบบถาวรจะกระทำได้โดยการเพิ่มคำสั่ง **RT_MPC=ON** ลงในไฟล์ /etc/environment
การวินิจฉัย: N/A

10. **TZ**

ไอเท็ม **Descriptor**
วัตถุประสงค์: ตั้งค่าเขตเวลา
ค่า: ดีฟอลต์: เขตเวลา Olson

ค่าที่เป็นไปได้: เขตเวลา Olson หรือเขตเวลา POSIX
การแสดง: **echo $TZ**
การเปลี่ยน: **TZ = value export TZ**

การเปลี่ยนแปลงจะมีผลทันทีใน shell การเปลี่ยนแปลงจะมีผลจนกว่าคุณจะล็อกเอาต์ออกจาก shell การเปลี่ยนแปลงแบบถาวรสามารถทำได้โดยการเพิ่มคำสั่ง **TZ= value** ลงในไฟล์ /etc/environment
การวินิจฉัย: N/A
การปรับ: POSIX อาจถูกใช้โดยแอ็พพลิเคชันที่ขึ้นอยู่ประสิทธิภาพและไม่ขึ้นอยู่กับความถูกต้องของการเปลี่ยนแปลงกฎของเขตเวลาและเวลาออมแสง

11. VMM_CNTRL

ไอเท็ม	**Descriptor**
วัตถุประสงค์:	อนุญาตให้ปรับโปรแกรมจัดการ หน่วยความจำเสมือน
ค่า:	ดีฟอลต์: ไม่ได้ตั้งค่าไว้
	ค่าที่ เป็นไปได้คือ: vmm_fork_policy, ESID_ALLOCATOR, SHM_1TB_SHARED, SHM_1TB_UNSHARED
การแสดง:	**echo $ VMM_CNTRL**
การเปลี่ยน:	**VMM_CNTRL={vmm_fork_policy=... | ESID_ALLOCATOR=... | ...}export VMM_CNTRL**
	การเปลี่ยนแปลงจะมีผลบังคับใช้โดยทันทีใน shell นี้ การเปลี่ยนแปลงมีผลบังคับใช้จนกว่าคุณล็อกเอาต์ออกจาก เชลล์นี้ การเปลี่ยนแปลงถาวรสามารถทำได้โดยเพิ่มตัวแปรสภาวะแวดล้อม VMM_CNTRL= ลงในไฟล์ /etc/environment
การวินิจฉัย:	N/A
การปรับ:	ตัวแปรสภาวะแวดล้อม VMM_CNTRL สามารถนำมาใช้เพื่อควบคุมโปรแกรมจัดการหน่วยความจำเสมือน คุณสามารถ ระบุหลายอ็อพชันโดยใช้ตัวแปรสภาวะแวดล้อม VMM_CNTRL และโดยคั่นระหว่างอ็อพชันด้วยเครื่องหมาย '@' ตัวอย่าง ของการระบุหลายอ็อพชันเป็นดังนี้:
	VMM_CNTRL=vmm_fork_policy=COW@SHM_1TB_SHARED=5
	เมื่อ คุณระบุอ็อพชัน vmm_fork_policy=COW vmm จะใช้ นโยบาย copy-on-write fork-tree ในทุกเมื่อที่กระบวนการถูก fork นี่เป็นการทำงานดีฟอลต์ เมื่อต้องการป้องกันไม่ให้ vmm ใช้นโยบาย copy-on-write ให้ใช้อ็อพชัน vmm_fork_policy=COR ถ้าระบุอ็อพชัน vmm_fork_policy โกลบอล vmm_fork_policy ที่ปรับได้จะถูกละเว้น
	ถ้าระบุอ็อพชัน ESID_ALLOCATOR ระบบจะควบคุมโปรแกรมจัดสรรจากการจัดสรร shmat และ mmap ทางอ้อม ดูที่ "การกำหนดสมนามเซกเมนต์ 1 TB" ในหน้า 176 เพื่อดูข้อมูลเพิ่มเติม
	ถ้าระบุ SHM_1TB_SHARED หรือ SHM_1TB_UNSHARED ระบบจะควบคุมการใช้ ส่วนหน่วยความจำที่แบ่งใช้ 1 TB ดูที่ "การกำหนดสมนามเซกเมนต์ 1 TB" ในหน้า 176 เพื่อดูข้อมูลเพิ่มเติม

12. AIX_STDBUFSZ

ไอเท็ม	**Descriptor**
วัตถุประสงค์:	กำหนดคอนฟิกขนาดบัฟเฟอร์สำหรับการเรียกระบบการอ่านและการเขียน ที่สร้างโดยคำสั่ง **cp,mv,cat, cpio** สิ่งนี้สามารถเรียกทำงานได้สำหรับการทำบัฟเฟอร์สตรีม
ค่า:	ค่าดีฟอลต์: ไม่ได้ตั้งค่าไว้
	ค่าที่อาจเป็นไปได้: ค่าเลขจำนวนเต็มที่ระบุขนาดบัฟเฟอร์ในไบต์, KB, MB
การแสดง:	**echo $ AIX_STDBUFSZ**
การเปลี่ยน:	**AIX_STDBUFSZ=1024; export AIX_STDBUFSZ** (เมื่อกำหนดคอนฟิก ขนาดบัฟเฟอร์ 1024)
	การเปลี่ยนแปลงนี้มีผลบังคับใช้ในทันทีในเชลล์นี้ การเปลี่ยนแปลงมีผลบังคับใช้จนกว่าคุณล็อกเอาต์ออกจาก เชลล์นี้ การเปลี่ยนแปลงขนาดบัฟเฟอร์แบบถาวรสามารถทำได้โดยเพิ่มตัวแปรสภาวะแวดล้อม AIX_STDBUFSZ ลงในไฟล์ /etc/environment
การวินิจฉัย:	N/A
การปรับ:	ระบุค่าด้วยวิธีต่อไปนี้
	• ระบุค่าเลขจำนวนเต็มโดยใช้รูปแบบ export AIX_STDBUFSZ=1024
	• ระบุค่าเลขฐานหกโดยใช้รูปแบบ export AIX_STDBUFSZ=0x400
	• ข้อจำกัด: ค่าจำกัดต่ำสุดคือ 64 ไบต์และค่าจำกัดสูงสุดคือ 127 MB
	• เลขจำนวนเต็มที่ถูกต้องที่อยู่ภายนอกค่าจำกัดเหล่านี้จะถูกแปลงกลับไปเป็นค่าจำกัดที่ใกล้ที่สุด
	• หากค่าที่ระบุไม่ใช่กำลัง 2 ค่านั้นจะถูกปัดให้เป็นค่าที่ต่ำที่ใกล้เคียงกับกำลัง 2 มากที่สุด
	• ถ้าค่าของพารามิเตอร์ AIX_STDBUFSZ ไม่ถูกต้อง ค่านั้นจะถูกละเว้น

13. AIX_LDSYM

ไอเท็ม	Descriptor
วัตถุประสงค์:	ข้อมูลบรรทัดต้นทางในไฟล์ Lightweight_core ไม่ถูกแสดงโดยดีฟอลต์เมื่อขนาดเพจข้อความเป็น 64 K เมื่อขนาดเพจข้อความเป็น 64K ให้ใช้ตัวแปรสภาวะแวดล้อม AIX_LDSYM=ON เพื่อดู ข้อมูลบรรทัดต้นทางในไฟล์ Lightweight_core
ค่า:	ค่าดีฟอลต์: ไม่ได้ตั้งค่าไว้ ค่าที่เป็นไปได้: ON.
การแสดง:	**echo $ AIX_LDSYM**
การเปลี่ยน:	**export AIX_LDSYM=ON** การเปลี่ยนแปลงนี้มีผลบังคับใช้ในทันทีในเชลล์นี้ การเปลี่ยนแปลงมีผลบังคับใช้จนกว่าคุณล็อกเอาต์ออกจาก เชลล์นี้ การเปลี่ยนแปลงกับระบบแบบถาวร สามารถทำได้โดยการเพิ่มตัวแปรสภาวะแวดล้อม AIX_LDSYM=ON เข้ากับไฟล์ /etc/environment
การวินิจฉัย:	N/A
การปรับ:	ใช้พารามิเตอร์นี้สำหรับแอ็พพลิเคชันที่มีขนาดเพจข้อความ 64 K และต้องการข้อมูลบรรทัดต้นทางในไฟล์ Lighweight_core

ความสามารถในการเรียกทำงานขนาดใหญ่แบบ 32 บิต

สำหรับความสามารถในการเรียกทำงานแบบ 32 บิตโดยส่วนใหญ่ ซึ่งข้อความและส่วนของโหลดเดอร์จะอยู่ภายในไฟล์แรกที่มีขนาด 256 MB ซึ่ง AIX จะจองเซ็กเมนต์ 0x1 ของการประมวลผลพื้นที่แอดเดรสสำหรับข้อมูลแบบอ่านอย่างเดียวที่สามารถเรียกทำงานได้

อย่างไรก็ตาม สำหรับความสามารถในการเรียกทำงานขนาดใหญ่ขนาด 32 บิต ซึ่งขนาดโดยรวมของส่วนของข้อความและโหลดเดอร์จะมีขนาดใหญ่กว่าส่วน และจำเป็นต้องมีส่วนแบบอ่านอย่างเดียวที่ต่อเนื่องกัน

ตำแหน่งพื้นที่แอดเดรสการประมวลผลของเซ็กเมนต์แบบอ่านอย่างเดียว ที่สามารถเรียกทำงานได้ซึ่งมีขนาดใหญ่:

อ็อพชัน **HUGE_EXEC** ของตัวแปรสภาวะแวดล้อม **LDR_CNTRL** จัดเตรียมให้ผู้ใช้ควบคุมผ่านตำแหน่งพื้นที่แอดเดรสการประมวลผล ของเซ็กเมนต์แบบอ่านอย่างเดียว

การใช้อ็อพชันนี้มีดังต่อไปนี้:

```
LDR_CNTRL=[...@]HUGE_EXEC={<segno>|0}[,<attribute>][@...]
```

โดยที่ *segno* คือหมายเลขเซ็กเมนต์เริ่มต้นที่ร้องขอในพื้นที่แอดเดรสการประมวลผลแบบ 32 บิต หรือศูนย์

ถ้าคุณระบุค่า segno ที่ไม่ใช่ศูนย์ โหลดเดอร์ของระบบจะพยายามแทรก เซ็กเมนต์แบบอ่านอย่างเดียวที่สามารถเรียกทำงานได้ซึ่งมีขนาดใหญ่ลงในพื้นที่แอดเดรสการประมวลผล ที่ตำแหน่งที่สอดคล้องกับหมายเลขเซ็กเมนต์ที่เริ่มต้นการร้องขอ

ถ้าคุณระบุค่า segno ที่เป็นศูนย์ โหลดเดอร์ของระบบจะพยายามแทรก เซ็กเมนต์แบบอ่านอย่างเดียวที่สามารถเรียกทำงานได้ซึ่งมีขนาดใหญ่ลงในพื้นที่แอดเดรสการประมวลผล ที่ตำแหน่งที่สอดคล้องกับหมายเลขเซ็กเมนต์ที่เริ่มต้นการร้องขอ

ถ้าคุณระบุค่า segno ที่เป็นศูนย์ (หรือในการไม่มีอยู่ของอ็อพชัน **HUGE_EXEC** ให้กับ **LDR_CNTRL**) โหลดเดอร์ของระบบจะเลือกหมายเลขเซ็กเมนต์เริ่มต้น ตามแบบจำลองพื้นที่แอดเดรส อัลกอริธึมที่ใช้เพื่อดำเนินการกับการเลือกนี้ จะคล้ายกับแฟล็ก **MAP_VARIABLE** ของรูทีนย่อย **mmap** :

- ถ้าไม่มี Dynamic Segment Allocation (DSA) หรือข้อมูลเพจขนาดใหญ่ถูกร้องขอโดยระบบ ระบบจะเลือกชุดของเซ็กเมนต์ที่ต่อเนื่องกันโดยทันที ตามด้วยฮีปการประมวลผล
- หรือ ระบบจะเลือกชุดของเซ็กเมนต์ที่ต่อเนื่องกันโดยทันที ซึ่งอยู่ต่ำกว่าเซ็กเมนต์พื้นที่ไลบรารีที่แบ่งใช้ต่ำที่สุด ถ้ามี

หมายเลขเซ็กเมนต์เริ่มต้นต้องไม่ขัดแย้งกับเซ็กเมนต์ใดๆ ที่จองแล้วโดยแบบจำลองพื้นที่แอดเดรสการประมวลผลที่ร้องขอ การพิจารณาข้อขัดแย้งที่มีอยู่ ซึ่งต้องการฮีปการประมวลผลนั้น และเซ็กเมนต์พื้นที่ไลบรารีที่แบ่งใช้ ต้องถูกนำมาพิจารณาถ้า

มีในกรณีที่เซ็กเมนต์ฮีปการประมวลผล ถูกจัดสรรแบบไดนามิก (DSA หรือข้อมูลเพจขนาดใหญ่) เฉพาะเซ็กเมนต์ของฮีปเริ่มต้นเท่านั้นที่จะถูกนำมาพิจารณาเพื่อจอง ถ้าหมายเลขเซ็กเมนต์เริ่มต้นที่เลือกไว้ขัดแย้งกับเซ็กเมนต์ที่จองไว้ใดๆ การประมวลผลจะล้มเหลวพร้อมกับ **ENOMEM**

สภาพพร้อมใช้งานเซ็กเมนต์ 0x1 สำหรับข้อความไลบรารีที่แบ่งใช้:

ความสามารถเรียกทำงานที่มีขนาดใหญ่ประกอบด้วยเซ็กเมนต์แบบอ่านอย่างเดียวอย่างต่อเนื่อง และไม่สามารถตั้งอยู่ในเซ็กเมนต์ 0x1 เป็นบางส่วน เนื่องจากความสามารถเรียกทำงานที่มีขนาดใหญ่ ไม่สามารถสร้างการใช้เซ็กเมนต์ 0x1 ได้ ส่วนนี้ของพื้นที่แอดเดรสการประมวลผล สามารถทำให้พร้อมใช้งานได้สำหรับวัตถุประสงค์อื่นๆ

แอ็ตทริบิวต์เผื่อเลือกในอ็อพชันการควบคุมโหลดเดอร์ **HUGE_EXEC** จะอนุญาตให้คุณร้องขอเซ็กเมนต์ข้อความไลบรารีที่แบ่งใช้เพื่อวางอยู่ในเซ็กเมนต์ 0x1 แทน 0xD:

```
HUGE_EXEC={<segno>|0},shtext_in_one
```

เนื่องจากข้อมูลการเปลี่ยนตำแหน่งของพื้นที่ไลบรารีที่แบ่งใช้ใหม่มีประโยชน์ เมื่อเซ็กเมนต์ข้อความที่แบ่งใช้ตั้งอยู่ในเซ็กเมนต์ 0xD การประมวลผลที่ร้องขออ็อพชันนี้ ไม่มีผลประโยชน์ของข้อมูลไลบรารีที่เปลี่ยนตำแหน่งล่วงหน้า ถัดมา ข้อมูลไลบรารีที่แบ่งใช้จะตั้งอยู่ในฮีปการประมวลผล ซึ่งมีผลประโยชน์ของการล้างข้อมูล เซ็กเมนต์ 0x3–0xF สำหรับการแบ่งใช้โดยฮีปการประมวลผล (mmap/shmat) และสามารถเรียกทำงานได้

หมายเหตุ: แอ็ตทริบิวต์ **shtext_in_one** ที่ใช้ในการเชื่อมต่อกับ **maxdata** และค่าติดตั้ง DSA ที่ต้องขัดขวางก่อนการประมวลผลจากการใช้ประโยชน์พื้นที่ไลบรารีที่แบ่งใช้ (ตัวอย่างเช่น `maxdata>0xA0000000/dsa` หรือ `maxdata=0/dsa`) อนุญาตให้การประมวลผลใช้ประโยชน์จาก ผลการทำงานที่ข้อความไลบรารีที่แบ่งใช้จัดเตรียมไว้

ถ้าพื้นที่ไลบรารีที่แบ่งใช้ของการประมวลผลตั้งชื่อพื้นที่ที่สร้างด้วยแอ็ตทริบิวต์ **doubletext32** ดังนั้น จึงไม่มีการเปลี่ยนตำแหน่งเซ็กเมนต์ข้อมูลล่วงหน้า และเซ็กเมนต์ข้อความไลบรารีที่แบ่งใช้ต้องถูกนำมาใช้ในกรณีนี้ เซ็กเมนต์หลัก (โดยปกติตั้งอยู่ในเซ็กเมนต์ 0xD) จะถูกย้ายไปยังเซ็กเมนต์ 0x1 และเซ็กเมนต์ข้อความไลบรารีที่แบ่งใช้สำรองยังคงอยู่ในเซ็กเมนต์ 0xF นี่เป็นการเพิ่มจำนวนสูงสุดของเซ็กเมนต์ (0x3–0xE) อย่างต่อเนื่อง ซึ่งสามารถแบ่งใช้โดยการประมวลผลฮีป (mmap/shmat) และสามารถเรียกทำงานได้

ขณะที่ความสามารถในการเรียกทำงานที่มีขนาดไม่ใหญ่ พร้อมด้วยค่า **maxdata** จะมีค่ามากกว่า 0xA0000000 และ DSA ที่เปิดใช้งานซึ่งได้รับการป้องกันจากการใช้พื้นที่ไลบรารีที่แบ่งใช้ในทุกกรณี ความสามารถเรียกทงำานที่มีขนาดใหญ่ (1) ใช้พื้นที่ไลบรารีที่แบ่งใช้ที่มีชื่อพร้อมกับแอ็ตทริบิวต์ **doubletext32** และ (2) ระบุแอ็ตทริบิวต์ **shtext_in_one** ซึ่งสามารถร้องขอค่า **maxdata** ได้มากสุด 0xC0000000 ก่อนที่จะสูญเสียสภาพพร้อมใช้งานในพื้นที่

ตัวอย่างความสามารถในการเรียกทำงานขนาดใหญ่:

ตัวอย่างสถานการณ์จำลองของการใช้ความสามารถในการเรียกทำงานขนาดใหญ่

ตัวอย่างแบบจำลองพื้นที่แอดเดรสของโปรแกรมที่มีขนาดใหญ่

ถ้าแบบจำลองพื้นที่แอดเดรสที่ต้องการ เป็นดังนี้:

- `MAXDATA=0x50000000` (non-DSA)
- ส่วนของ shmat/mmap ที่จำเป็นต้องมีในเซ็กเมนต์ 0xB, 0xC และ 0xE
- ข้อความพื้นที่ไลบรารีที่แบ่งใช้และความสามารถในการเข้าถึงข้อมูลที่เปลี่ยนที่ตั้งล่วงหน้า

โครงร่างพื้นที่แอดเดรสมีอัตราผลประโยชน์ดังต่อไปนี้:

```
0x0: System Segment
0x1:
0x2: Exec-time Private Dependencies / Stack
0x3: Process Heap
0x4: Process Heap
0x5: Process Heap
0x6: Process Heap
0x7: Process Heap
0x8:
0x9:
0xA:
0xB: shmat/mmap (mapped after exec)
0xC: shmat/mmap (mapped after exec)
0xD: Shared Library Text
0xE: shmat/mmap (mapped after exec)
0xF: Pre-relocated Library Data
```

คุณสามารถมองเห็นได้จากตัวอย่างนี้ ซึ่งเซ็กเมนต์ 0x8–0xA พร้อมใช้งานสำหรับความสามารถในการเรียกทำงาน

สมมุติฐานที่ว่า ขนาดที่สามารถเรียกทำงานได้จะมีขนาดใหญ่กว่า 256 MB และน้อยกว่า 512 MB นั้น จะมีรูปแบบค่าติดตั้ง **HUGE_EXEC** สำหรับสถานการณ์นี้ดังต่อไปนี้:

1. HUGE_EXEC=0
2. HUGE_EXEC=0x8
3. HUGE_EXEC=0x9

อ็อพชัน 1 และ 2 จะแทรกความสามารถในการเรียกทำงานลงในเซ็กเมนต์ 0x8–0x9 ขณะที่อ็อพชัน 3 จะแทรกความสามารถในการเรียกทำงานลงในเซ็กเมนต์ 0x9–0xA

ตัวอย่างแบบจำลองพื้นที่แอดเดรสของโปรแกรมที่มีขนาดใหญ่มาก

ถ้าแบบจำลองพื้นที่แอดเดรสที่ต้องการ เป็นดังนี้:

- MAXDATA=0x50000000 DSA
- ส่วนของ shmat/mmap ที่จำเป็นต้องมีในเซ็กเมนต์ 0xB, 0xC และ 0xE
- ข้อความพื้นที่ไลบรารีที่แบ่งใช้และความสามารถในการเข้าถึงข้อมูลที่เปลี่ยนที่ตั้งล่วงหน้า

โครงร่างพื้นที่แอดเดรสมีอัตราผลประโยชน์ดังต่อไปนี้:

```
0x0: System Segment
0x1:
0x2: Exec-time Private Dependencies / Stack
0x3: Process Heap
0x4:      |
0x5:      |
0x6:      v
0x7: __________ (data limit)
0x8:
0x9:
0xA:
0xB: shmat/mmap (mapped after exec)
```

```
0xC: shmat/mmap (mapped after exec)
0xD: Shared Library Text
0xE: shmat/mmap (mapped after exec)
0xF: Pre-relocated Library Data
```

คุณสามารถมองเห็นได้จากตัวอย่างนี้ ซึ่งเซ็กเมนต์ 0x4–0xA จะพร้อมใช้งานสำหรับความสามารถในการเรียกทำงาน

สมมุติฐานที่ว่า ขนาดที่สามารถเรียกทำงานได้จะมีขนาดใหญ่กว่า 256 MB และน้อยกว่า 512 MB นั้น จะมีรูปแบบค่าติดตั้ง **HUGE_EXEC** สำหรับสถานการณ์นี้ดังต่อไปนี้:

1. HUGE_EXEC=0x8
2. HUGE_EXEC=0x9

อ็อพชัน 1 จะแทรกความสามารถในการเรียกทำงานลงในเซ็กเมนต์ 0x8–0x9 ขณะที่อ็อพชัน 2 จะแทรกความสามารถในการเรียกทำงานลงในเซ็กเมนต์ 0x9–0xA

หมายเหตุ: ค่าติดตั้ง HUGE_EXEC=0 จะไม่เหมาะสมสำหรับลูกค้านี้ เนื่องจากระบบจะเลือกเซ็กเมนต์ 0xB–0xC สำหรับความสามารถในการเรียกทำงาน (เนื่องจาก DSA) ซึ่งจะปกป้องเซ็กเมนต์เหล่านี้จากความพร้อมใช้งาน สำหรับ shmat/mmap หลังจาก exec การตั้งค่า **HUGE_EXEC** ไปเป็นเซ็กเมนต์ 0x4, 0x5, 0x6 หรือ 0x7 ขณะที่อนุญาตให้ใช้การแทรก ตามคำร้องขอจะส่งผลทำให้มีข้อจำกัดในการประมวลผลฮีปที่ทำให้เซ็กเมนต์โตขึ้น ซึ่งจะอยู่ต่ำกว่าจุดเริ่มต้นของเซ็กเมนต์ที่ร้องขอ

ตัวอย่างแบบจำลองพื้นที่แอดเดรสของโปรแกรมที่มีขนาดใหญ่มากที่ไม่ได้เข้าถึง พื้นที่ไลบรารีที่แบ่งใช้

ถ้าแบบจำลองพื้นที่แอดเดรสที่ต้องการ เป็นดังนี้:

- MAXDATA=0xB0000000 DSA
- ไม่มีส่วนของ shmat/mmap
- ไม่มีความสามารถในการเข้าถึงพื้นที่ไลบรารีที่แบ่งใช้

โครงร่างพื้นที่แอดเดรสมีอัตราผลประโยชน์ดังต่อไปนี้:

```
0x0: System Segment
0x1:
0x2: Exec-time Private Dependencies / Stack
0x3: Process Heap
0x4:      |
0x5:      |
0x6:      |
0x7:      |
0x8:      |
0x9:      |
0xA:      |
0xB:      |
0xC:      v
0xD: ___________ (data limit)
0xE:
0xF:
```

คุณสามารถมองเห็นได้จากตัวอย่างนี้ ซึ่งเซ็กเมนต์ 0x4–0xF จะพร้อมใช้งานสำหรับความสามารถในการเรียกทำงาน

สมมุติฐานที่ว่า ขนาดที่สามารถเรียกทำงานได้จะมีขนาดใหญ่กว่า 256 MB และน้อยกว่า 512 MB นั้น จะมีรูปแบบค่าติดตั้ง **HUGE_EXEC** สำหรับสถานการณ์นี้ดังต่อไปนี้:

1. HUGE_EXEC=0
2. HUGE_EXEC=0xE

อ็อพชันทั้งสองจะแทรกความสามารถในการเรียกทำงานลงในเซ็กเมนต์ 0xE–0xF

หมายเหตุ: การตั้งค่า HUGE_EXEC ไปเป็นเซ็กเมนต์ 0x4–0xD ขณะที่อนุญาตให้ใช้การแทรกตามคำร้องขอ จะส่งผลทำให้มีข้อจำกัดในการประมวลผลฮีปที่ทำให้เซ็กเมนต์นั้นโตขึ้น ซึ่งจะอยู่ต่ำกว่าจุดเริ่มต้นของเซ็กเมนต์ที่ร้องขอ

ตัวอย่างแบบจำลองพื้นที่แอดเดรสของการประมวลผลที่เป็นค่าดีฟอลต์

ถ้าแบบจำลองพื้นที่แอดเดรสที่ต้องการ เป็นดังนี้:

- MAXDATA=0 (non-DSA)
- ไม่มีส่วนของ shmat/mmap
- ข้อความพื้นที่ไลบรารีที่แบ่งใช้และความสามารถในการเข้าถึงข้อมูลที่เปลี่ยนที่ตั้งล่วงหน้า

โครงร่างพื้นที่แอดเดรสมีอัตราผลประโยชน์ดังต่อไปนี้:

```
0x0: System Segment
0x1:
0x2: Exec-time Private Dependencies / Process Heap / Stack
0x3:
0x4:
0x5:
0x6:
0x7:
0x8:
0x9:
0xA:
0xB:
0xC:
0xD: Shared Library Text
0xE:
0xF: Pre-relocated Library Data
```

คุณสามารถมองเห็นได้จากตัวอย่างนี้ ซึ่งเซ็กเมนต์ 0x3–0xC จะพร้อมใช้งานสำหรับความสามารถในการเรียกทำงาน

สมมุติฐานที่ว่า ขนาดที่สามารถเรียกทำงานได้จะมีขนาดใหญ่กว่า 256 MB และน้อยกว่า 512 MB นั้น จะมีรูปแบบค่าติดตั้ง **HUGE_EXEC** สำหรับสถานการณ์นี้ดังต่อไปนี้:

```
 1. HUGE_EXEC=0
 2. HUGE_EXEC=0x3
 ...
10. HUGE_EXEC=0xB
```

อ็อพชัน 1 และ 2 มีผลลัพธ์เฉพาะ นั่นคือ– การแทรกความสามารถในการเรียกทำงานลงในเซ็กเมนต์ 0x3–0x4

ตัวอย่าง shtext_in_one พร้อมกับเซ็กเมนต์ข้อความพื้นที่ไลบรารีที่แบ่งใช้ แบบเดี่ยว

ถ้าแบบจำลองพื้นที่แอดเดรสที่ต้องการเป็นดังนี้:

- MAXDATA=0x70000000 (non-DSA)
- ส่วนของ shmat/mmap ที่จำเป็นต้องมีในเซ็กเมนต์ 0xC, 0xD, 0xE และ 0xF
- ความสามารถในการเข้าถึงพื้นที่ไลบรารีที่แบ่งใช้

โครงร่างพื้นที่แอดเดรสมีอัตราผลประโยชน์ดังต่อไปนี้:

```
0x0: System Segment
0x1: Shared Library Text
0x2: Exec-time Private Dependencies / Stack
0x3: Process Heap
0x4: Process Heap
0x5: Process Heap
0x6: Process Heap
0x7: Process Heap
0x8: Process Heap
0x9: Process Heap
0xA:
0xB:
0xC: shmat/mmap (mapped after exec)
0xD: shmat/mmap (mapped after exec)
0xE: shmat/mmap (mapped after exec)
0xF: shmat/mmap (mapped after exec)
```

คุณสามารถมองเห็นได้จากตัวอย่างนี้ ซึ่งเซ็กเมนต์ 0xA–0xB จะพร้อมใช้งานสำหรับความสามารถในการเรียกทำงาน

สมมุติฐานที่ว่า ขนาดที่สามารถเรียกทำงานได้จะมีขนาดใหญ่กว่า 256 MB และน้อยกว่า 512 MB นั้น จะมีรูปแบบค่าติดตั้ง **HUGE_EXEC** สำหรับสถานการณ์นี้ดังต่อไปนี้:

1. HUGE_EXEC=0,shtext_in_one
2. HUGE_EXEC=0xA,shtext_in_one

อ็อพชันทั้งสองจะแทรกความสามารถในการเรียกทำงานลงในเซ็กเมนต์ 0xA–0xB และแทรกข้อความไลบรารีที่แบ่งใช้ลงในเซ็กเมนต์ 0x1

หมายเหตุ: การตั้งค่า HUGE_EXEC ไปเป็น 0xB–0xE ขณะที่อนุญาตให้ใช้การแทรกตามคำร้องขอ จะป้องกันเซ็กเมนต์ 0xC–0xF บางส่วนจากความพร้อมใช้งานสำหรับ shmat/mmap หลังจากที่สามารถเรียกทำงานได้

ตัวอย่าง shtext_in_one พร้อมกับเซ็กเมนต์ข้อความพื้นที่ไลบรารีที่แบ่งใช้สองเซ็กเมนต์

ถ้าแบบจำลองพื้นที่แอดเดรสที่ต้องการ เป็นดังนี้:

- MAXDATA=0x70000000 DSA
- ส่วนของ shmat/mmap ที่จำเป็นต้องมีในเซ็กเมนต์ 0xA และ 0xB
- ความสามารถในการเข้าถึงข้อความพื้นที่ไลบรารีที่แบ่งใช้ (สร้างขึ้นด้วยแอ็ตทริบิวต์ **doubletext32**)

โครงร่างพื้นที่แอดเดรสมีอัตราผลประโยชน์ดังต่อไปนี้:

```
0x0: System Segment
0x1: Shared Library Text (primary)
0x2: Exec-time Private Dependencies / Stack
0x3: Process Heap
0x4:      |
0x5:      |
0x6:      |
0x7:      |
0x8:      v
0x9: ___________ (data limit)
0xA: shmat/mmap (mapped after exec)
0xB: shmat/mmap (mapped after exec)
0xC:
0xD:
0xE:
0xF: Shared Library Text (secondary)
```

คุณสามารถมองเห็นได้จากตัวอย่างนี้ ซึ่งเซ็กเมนต์ 0xC–0xE จะพร้อมใช้งานสำหรับความสามารถในการเรียกทำงาน

สมมุติฐานที่ว่า ขนาดที่สามารถเรียกทำงานได้จะมีขนาดใหญ่กว่า 512 MB และน้อยกว่า 768 MB นั้น จะมีรูปแบบค่าติดตั้ง **HUGE_EXEC** สำหรับสถานการณ์นี้ดังต่อไปนี้:

1. `HUGE_EXEC=0,shtext_in_one`
2. `HUGE_EXEC=0xC,shtext_in_one`

อ็อพชันทั้งสองจะแทรกความสามารถในการเรียกทำงานลงในเซ็กเมนต์ 0xC–0xE และแทรกข้อความไลบรารีที่แบ่งใช้ลงในเซ็กเมนต์ 0x1 และ 0xF

หมายเหตุ: การตั้งค่า HUGE_EXEC ไปเป็นเซ็กเมนต์ 0x4–0x7 ขณะที่อนุญาตให้ใช้การแทรกตามคำร้องขอ จะส่งผลทำให้มีข้อจำกัดในการประมวลผลฮีปที่ทำให้เซ็กเมนต์นั้นโตขึ้น ซึ่งจะอยู่ต่ำกว่าจุดเริ่มต้นของเซ็กเมนต์ที่ร้องขอ

ตัวแปรสภาวะแวดล้อม ASO

active system optimizer (ASO) ของระบบที่สามารถปรับแต่งได้จะถูกจัดการโดยใช้คำสั่ง **asoo** คุณสามารถตั้งค่า ASO แบบกำหนดเองสำหรับกระบวนการเดียวโดยใช้ตัวแปรสภาวะแวดล้อม ASO

ASO_ENABLED

ไอเท็ม	Descriptor
วัตถุประสงค์:	ใช้เพื่อรวมหรือแยกกระบวนการออกจากการปรับแต่ง ASO อย่างชัดเจน
ค่า:	ดีฟอลต์: ASO จะปรับกระบวนการให้เหมาะสม ถ้ายอมรับเกณฑ์การปรับ ASO ให้เหมาะสม
	ค่าที่ใช้ได้: ALWAYS หรือ NEVER
	ALWAYS – ASO จะจัดลำดับความสำคัญกระบวนการนี้
	NEVER – ASO จะไม่ปรับกระบวนการนี้ให้เหมาะสม
การแสดงผล:	**echo $ASO_ENABLED**
	ค่านี้จะถูกเปิดใช้ภายใน และดังนั้นค่าดีฟอลต์เริ่มต้นจะไม่ถูกแสดงด้วยคำสั่ง **echo**

ไอเท็ม	Descriptor
การเปลี่ยนแปลง:	**ASO_ENABLED=[ALWAYS|NEVER] export ASO_ENABLED** การเปลี่ยนแปลงจะมีผลกับกระบวนการที่รันหลังจากการตั้งค่าตัวแปรแล้ว การเปลี่ยนแปลงนี้จะมีผลจนกว่าคุณจะล็อกเอาต์ออกจาก shell การเปลี่ยนแปลงแบบถาวรจะทำได้โดยการเพิ่มคำสั่ง **ASO_ENABLED=[ALWAYS|NEVER]** ลงในไฟล์ `/etc/environment`
การวิเคราะห์:	N/A
การปรับ:	N/A

พารามิเตอร์ที่สามารถปรับแต่งเคอร์เนลได้

พารามิเตอร์การปรับแต่เคอร์เนล AIX จะถูกจัดหมวดหมู่ออกเป็นหกกลุ่มคือ: ตัวกำหนดตารางเวลาและความสามารถในการปรับการควบคุมการโหลดหน่วยความจำ ความสามารถในการปรับ VMM ความสามารถในการปรับ I/O แบบซิงโครนัส ความสามารถในการปรับ I/O แบบอะซิงโครนัส ความสามารถในการปรับดิสก์และดิสก์อะแด็ปเตอร์ และความสามารถในการปรับการสื่อสารสำหรับการประมวลผลระหว่างกัน

การแก้ไข

AIX นำเสนอโหมด ส่วนกลางที่มีความยืดหยุ่นมากขึ้นสำหรับการตั้งค่าพารามิเตอร์การปรับเคอร์เนล AIX ส่วนใหญ่

ขณะนี้ สามารถทำการเปลี่ยนแปลงถาวรได้โดยไม่ต้องแก้ไขไฟล์ **rc** ด้วยวิธีการวางค่ารีบูตสำหรับพารามิเตอร์ที่ปรับได้ทั้งหมดไว้ใน ไฟล์ `/etc/tunables/nextboot` stanza ใหม่ เมื่อรีบูตเครื่อง ค่าในไฟล์นั้นจะมีการนำไปใช้โดยอัตโนมัติ

ไฟล์ `/etc/tunables/lastboot` stanza มีการสร้างขึ้นโดยอัตโนมัติ ด้วยค่าทั้งหมดที่ตั้งไว้ทันทีหลังจากรีบูต ทำให้สามารถกลับไปยังค่าเหล่านั้นได้ตลอดเวลา ล็อกไฟล์ `/etc/tunables/lastboot.log` จะบันทึกการเปลี่ยนแปลงใดๆ ที่ทำหรือไม่สามารถทำได้ระหว่างการรีบูต

คำสั่งต่อไปนี้พร้อมใช้งานเพื่อปรับเปลี่ยนไฟล์ที่ปรับได้:

คำสั่ง	วัตถุประสงค์
tunsave	บันทึกค่าในไฟล์ stanza
tunchange	อัพเดตค่าในไฟล์ stanza
tunrestore	ใช้ค่าพารามิเตอร์ที่ใช้ได้ซึ่งมีการระบุไว้ในไฟล์
tuncheck	ตรวจสอบไฟล์ที่สร้างขึ้นด้วยตนเอง
tundefault	รีเซ็ตพารามิเตอร์ที่ปรับได้เป็นค่าดีฟอลต์

คำสั่งข้างบนทั้งหมดทำงานทั้งบนค่าพารามิเตอร์ที่ปรับได้ปัจจุบันและ รีบูต หากต้องการข้อมูลเพิ่มเติม ให้ดูหน้าหลักตามลำดับ

หากต้องการข้อมูลเพิ่มเติมเกี่ยวกับการแก้ไขพารามิเตอร์การปรับเคอร์นัลเหล่านี้ ให้ดูส่วนการปรับเคอร์นัลใน *Performance Tools Guide and Reference*

การเปลี่ยนคำสั่ง vmtune และ schedtune

คำสั่ง **vmtune** และ **schedtune** ถูกแทนที่ ด้วยคำสั่ง **vmo, ioo,** และ **schedo** ทั้งคำสั่ง **vmo** และ **ioo** แทนที่ คำสั่ง **vmtune** ในขณะที่ คำสั่ง **schedo** แทนที่ **schedtune** คำสั่งใหม่ใช้พารามิเตอร์ที่มีอยู่ ทั้งหมด

คำสั่ง **ioo** จัดการพารามิเตอร์การปรับที่เกี่ยวข้องกับ I/O ทั้งหมด ในขณะที่คำสั่ง **vmo** จัดการพารามิเตอร์ Virtual Memory Manager หรือ VMM อื่นทั้งหมดซึ่งก่อนหน้านี้มีการจัดการ โดยคำสั่ง **vmtune** คำสั่งทั้งสามเป็นส่วนประกอบของชุดไฟล์ `bos.perf.tune` ซึ่งยังมีคำสั่ง **tunsave, tunrestore, tuncheck,** และ **tundefault** ด้วย ชุดไฟล์ `bos.adt.samples` ยังคงมีคำสั่ง **vmtune** และ **schedtune** ซึ่งเป็น shell สคริปต์ความเข้ากันได้ที่เรียก **vmo, ioo,** และ **schedo** ตามความเหมาะสม สคริปต์ความเข้ากันได้เหล่านี้ สนับสนุนเฉพาะการเปลี่ยนแปลงพารามิเตอร์ซึ่งสามารถเปลี่ยนแบบโต้ตอบเท่านั้น พารามิเตอร์ที่ต้องการ **bosboot** จากนั้นต้องรีบูตเครื่อง เพื่อให้มีผลบังคับใช้ไม่ได้รับการสนับสนุนจากสคริปต์ **vmtune** อีกต่อไป เพื่อเปลี่ยนพารามิเตอร์เหล่านี้ ขณะนี้ ผู้ใช้ต้องใช้คำสั่ง **vmo -r** อ็อพชันและพารามิเตอร์คำสั่ง **vmtune** ที่น่าสนใจ มีดังนี้:

อ็อพชัน vmtune ก่อนหน้านี้	การใช้งาน	คำสั่งใหม่
-C 0|1	การกำหนดสีหน้า	**vmo -r -o pagecoloring=0|1**
-g *n1* -L *n2*	จำนวนขนาดหน้าขนาดใหญ่ของหน้าขนาดใหญ่ที่จะสำรองไว้	**vmo -r -o lgpg_size=n1 -o lgpg_regions=n2**
-m *n*	พูลหน่วยความจำ	**vmo -r -o mempools=n**
-v *n*	จำนวนเฟรมต่อพูลหน่วยความจำ	**vmo -r -o framesets=n**
-i *n*	ช่วงสำหรับตัวระบุเซ็กเมนต์ข้อมูลพิเศษ	**vmo -r -o spec_dataseg_int=n**
-V *n*	จำนวนตัวระบุเซกเมนต์ข้อมูลพิเศษที่จะสำรองไว้	**vmo -r -o num_spec_dataseg=n**
-y 0|1	p690 affinity หน่วยความจำ	**vmo -r -o memory_affinity=0|1**

สคริปต์ความเข้ากันได้ **vmtune** และ **schedtune** ไม่ได้จัดส่งพร้อมกับ AIX คุณสามารถอ้างอิงตารางต่อไปนี้เพื่อย้ายค่าติดตั้งของคุณไปยังคำสั่ง ใหม่:

ตัวเลือก schedtune	ค่าเทียบเท่า schedo	ฟังก์ชัน
-a ตัวเลข	-o affinity_lim=number	ตั้งค่าจำนวนของการสลับบริบทหลังจากที่นโยบาย SCHED_FIF02 ไม่ต้องการเธรดอีกต่อไป
-b ตัวเลข	-o idle_migration_barrier=number	ตั้งค่าตัวกั้นการย้าย idle
-c ตัวเลข	-o %usDelta=number	ควบคุมการปรับการเลื่อนนาฬิกา
-d ตัวเลข	-o sched_D=number	ตั้งค่าปัจจัยที่ใช้เพื่อ decay การใช้ CPU
-e ตัวเลข	-o v_exempt_seconds=number	ตั้งค่าเวลาก่อนหน้าที่กระบวนการซึ่งพักไว้และเรียกคืนก่อนหน้านี้ มีสิทธิพักไว้ใหม่ได้
-f ตัวเลข	-o pacefork=number	ตั้งค่าจำนวนของ clock ticks ที่จะล่าช้าก่อนที่จะลองการเรียก **fork** ที่ล้มเหลวใหม่
-F ตัวเลข	-o fixed_pri_global=number	เก็บเธรดระดับความสำคัญคงที่ไว้ในรันคิวสากล

ตัวเลือก schedtune	ค่าเทียบเท่า schedo	ฟังก์ชัน
-h ตัวเลข	-o v_repage_hi=number	เปลี่ยนเกณฑ์ทั่วทั้งระบบที่ใช้เพื่อกำหนด เวลาเริ่มต้นและสิ้นสุดของการพักกระบวนการ
-m ตัวเลข	-o v_min_process=number	ตั้งค่าระดับหลายโปรแกรมต่ำสุด
-p ตัวเลข	-o v_repage_proc=number	เปลี่ยนเกณฑ์สำหรับแต่ละกระบวนการที่ใช้เพื่อกำหนด กระบวนการที่จะพักไว้
-r ตัวเลข	-o sched_R=number	ตั้งค่าอัตราการสะสมการใช้ CPU
-s ตัวเลข	-o maxspin=number	ตั้งค่าจำนวนครั้งในการหมุนล็อกก่อนที่จะพักไว้
-t ตัวเลข	-o timeslice=number	ตั้งค่าจำนวนของส่วนเวลา 10 ms
-w ตัวเลข	-o v_sec_wait=number	ตั้งค่าจำนวนวินาทีที่จะรอหลังจาก thrashing สิ้นสุดลง ก่อนที่จะเพิ่มกระบวนการกลับเข้าใน mix

อ็อพชัน vmtune	ค่าเทียบเท่า vmo	ค่าเทียบเท่า ioo	ฟังก์ชัน
-b ตัวเลข		-o numfsbuf=number	ตั้งค่าจำนวนของ bufstructs ระบบไฟล์
-B ตัวเลข		-o hd_pbuf_cnt=number	พารามิเตอร์นี้ถูกแทนที่ด้วยพารามิเตอร์ *pv_min_pbuf*
-c ตัวเลข		-o numclust=number	ตั้งค่าจำนวนของคลัสเตอร์ 16 KB ที่ประมวลผลโดย write behind
-C 0|1	-r -o pagecoloring=0|1		ปิดใช้งานหรือเปิดใช้งานการกำหนดสีหน้าสำหรับฮาร์ดแวร์ แพล็ตฟอร์มเฉพาะ
-d 0|1	-o deffps=0|1		เปิดและปิดการจัดสรรพื้นที่ว่างหน้าที่เลื่อนออกไป
-e 0|1		-o jfs_clread_enabled=0|1	ควบคุมว่า JFS ใช้การอ่านที่คลัสเตอร์บนไฟล์ทั้งหมดหรือไม่
-E 0|1		-o jfs_use_read_lock=0|1	ควบคุมว่า JFS ใช้ล็อกแบบแบ่งใช้เมื่ออ่าน จากไฟล์หรือไม่
-f ตัวเลข	-o minfree=number		ตั้งค่าจำนวนเฟรมบนรายการที่ว่าง
-F ตัวเลข	-o maxfree=number		ตั้งค่าจำนวนเฟรมบนรายการที่ว่างซึ่ง จะหยุดการขโมยเฟรม
-g ตัวเลข	-o lgpg_size number		ตั้งค่าขนาดในหน่วยไบต์ของหน้าขนาดใหญ่ที่ฮาร์ดแวร์ สนับสนุน
-H ตัวเลข		-o pgahd_scale_thresh=number	ตั้งค่าจำนวนของหน้าที่ว่างใน mempool ซึ่งระบบปรับสเกล read-ahead ย้อนกลับ
-i ตัวเลข	-r -o spec_dataseg_int=number		ตั้งค่าช่วงที่จะใช้เมื่อสำรองตัวระบุ เซกเมนต์ข้อมูลพิเศษ
-j ตัวเลข		-o j2_nPagesPerWriteBehindCluster=number	ตั้งค่าจำนวนหน้าต่อ write-behind คลัสเตอร์
-J ตัวเลข		-o j2_maxRandomWrite=number	ตั้งค่าจำนวน random-write threshold
-k ตัวเลข	-o npskill=number		ตั้งค่าจำนวนของหน้าพื้นที่ว่างการเพจซึ่งจะเริ่มต้น killing กระบวนการ

อ็อพชัน vmtune	ค่าเทียบเท่า vmo	ค่าเทียบเท่า ioo	ฟังก์ชัน
-l ตัวเลข	-o lrubucket=number		ตั้งค่าขนาดของขนาด bucket การเปลี่ยนหน้าต่ำสุดที่ใช้ล่าสุด
-L ตัวเลข	-o lgpg_regions=number		ตั้งค่าจำนวนของหน้าขนาดใหญ่ที่จะสำรองไว้
-m ตัวเลข	-r -o mempools=number		พารามิเตอร์นี้ไม่มีอยู่
-M ตัวเลข	-o maxpin=number		ตั้งค่าเปอร์เซ็นต์สูงสุดของหน่วยความจำจริงที่สามารถ pinned ได้
-n ตัวเลข	-o nokilluid=number		ระบุช่วง **uid** ของกระบวนการที่ไม่ควรถูก killed เมื่อพื้นที่ว่างการเพจต่ำ
-N ตัวเลข		-o pd_npages=number	ตั้งค่าจำนวนหน้าที่ควรจะลบในหนึ่งก้อน ออกจาก RAM เมื่อลบไฟล์
-p ตัวเลข	-o minperm%=number		ตั้งค่าจุดต่ำสุดซึ่งหน้าไฟล์จะได้รับการป้องกันจากขั้นตอนวิธี repage
-P ตัวเลข	-o maxperm%=number		ตั้งค่าจุดสูงสุดซึ่งขั้นตอนวิธีการขโมยหน้า จะขโมยเฉพาะหน้าไฟล์เท่านั้น
-q ตัวเลข		-o j2_minPageReadAhead=number	ตั้งค่าจำนวนหน้าต่ำสุดที่จะ read ahead
-Q ตัวเลข		-o j2_maxPageReadAhead=number	ตั้งค่าจำนวนหน้าสูงสุดที่จะ read ahead
-r ตัวเลข		-o minpgahead=number	ตั้งค่าจำนวนหน้าซึ่ง read-ahead ตามลำดับ จะเริ่มต้น
-R ตัวเลข		-o maxpgahead=number	ตั้งค่าจำนวนหน้าสูงสุดที่จะ read-ahead
-s 0|1		-o sync_release_ilock=0|1	เปิดใช้งานหรือปิดใช้งานรหัสที่ลดเวลาที่ใช้ inode ล็อกให้เหลือน้อยที่สุดในระหว่าง **sync**
-S 0|1	-o v_pinshm=0|1		เปิดใช้งานหรือปิดใช้งานแฟล็ก **SHM_PIN** บนการเรียกระบบ **shmget**
-t ตัวเลข	-o maxclient%=number		ตั้งค่าจุดสูงสุดซึ่งขั้นตอนวิธีการขโมยหน้า จะขโมยเฉพาะหน้าไฟล์ไคลเอ็นต์เท่านั้น
-T ตัวเลข	-o pta_balance_threshold= number		ตั้งค่าจุดซึ่งจะจัดสรรเซกเมนต์ PTA ใหม่
-u ตัวเลข	-o lvm_bufcnt=number		ตั้งค่าจำนวนของบัฟเฟอร์ LVM สำหรับ raw ฟิสิคัล I/Os
-v ตัวเลข	-r -o framesets=number		ตั้งค่าจำนวนของ framesets ต่อพูลหน่วยความจำ
-V ตัวเลข	-r -o num_spec_dataseg= number		ตั้งค่าจำนวนของตัวระบุเซกเมนต์ข้อมูลพิเศษที่จะสำรองไว้
-w ตัวเลข	-o npswarn=number		ตั้งค่าจำนวนของหน้าพื้นที่ว่างการเพจที่ว่างซึ่งจะส่ง สัญญาณ SIGDANGER ไปยังกระบวนการ
-W ตัวเลข		-o maxrandwrt=number	ตั้งค่า threshold สำหรับการบันทึกแบบสุ่มที่จะสะสมใน RAM ก่อนที่หน้าจะถูกซิงโครไนซ์กับดิสก์โดยใช้ขั้นตอนวิธี write-behind

อ็อพชัน vmtune	ค่าเทียบเท่า vmo	ค่าเทียบเท่า ioo	ฟังก์ชัน
-y 0|1	-r -o memory_affinity=0|1		พารามิเตอร์นี้ไม่มีอยู่ Affinity หน่วยความจำเปิดอยู่เสมอ ถ้าฮาร์ดแวร์สนับสนุน
-z ตัวเลข		-o j2_nRandomCluster=number	ตั้งค่าระยะทาง threshold การบันทึกแบบสุ่ม
-Z ตัวเลข		-o j2_nBufferPerPagerDevice= number	ตั้งค่าจำนวนบัฟเฟอร์ต่ออุปกรณ์ pager

การปรับปรุงคำสั่ง no และ nfso

คำสั่ง **no** และ **nfso** มีการปรับปรุงเพื่อให้คุณสามารถเปลี่ยนแปลงพารามิเตอร์ที่สามารถปรับได้อย่างถาวร ด้วยไฟล์ /etc/tunables/nextboot คำสั่งเหล่านี้จะมีแฟล็ก **-h** ที่สามารถนำมาใช้เพื่อแสดงวิธีใช้เกี่ยวกับพารามิเตอร์ใดๆ

เนื้อหาของรายละเอียดเกี่ยวกับวิธีใช้ประกอบด้วย:

- วัตถุประสงค์ของพารามิเตอร์
- ค่าที่เป็นไปได้ เช่น ค่าดีฟอลต์ ช่วง และชนิด
- ข้อมูลการวินิจฉัยและการปรับเพื่อตัดสินใจว่า เมื่อใดควรเปลี่ยน ค่าพารามิเตอร์

คำสั่งการปรับทั้งหมดนี้ ซึ่งรวมถึง **ioo, nfso, no, vmo, raso** และ **schedo** จะใช้ไวยากรณ์ทั่วไป สำหรับรายละเอียดเพิ่มเติม และรายการของพารามิเตอร์การปรับที่สนับสนุนโดยสมบูรณ์ จะมองเห็น man page สำหรับแต่ละคำสั่ง

โหมดความเข้ากันได้ของ AIX

เมื่อย้ายไปยังโหมดความเข้ากันได้ มีการใช้เฉพาะคำสั่ง **no** และ **nfso** เนื่องจากคำสั่ง **vmtune** และ **schedtune** ไม่มีอยู่อีกต่อไป คุณสามารถใช้โหมดความเข้ากันได้เพื่อย้ายไปยัง กรอบงานการปรับใหม่ แต่ไม่แนะนำให้ใช้กับรีลีส AIX

โหมดความเข้ากันได้ช่วยให้คุณสามารถทำการเปลี่ยนแปลงแบบถาวรใน พารามิเตอร์ที่ปรับได้โดยฝังการเรียกคำสั่งการปรับในสคริปต์ ซึ่งเรียกระหว่างกระบวนการบูต เฉพาะความแตกต่างที่สังเกตเห็นได้เท่านั้น ที่ไฟล์ /etc/tunables/lastboot และ /etc/tunables/lastboot.log จะถูกสร้างขึ้นในระหว่างการรีบูต ไฟล์ lastboot.log จะมีคำเตือนที่บอกว่า AIX กำลังรันอยู่ในโหมดความเข้ากันได้ในปัจจุบัน และไฟล์ nextboot ไม่ได้ถูกนำมาใช้

ยกเว้นสำหรับพารามิเตอร์ของชนิด *Bosboot* (โปรดดู “การเปลี่ยนคำสั่ง vmtune และ schedtune” ในหน้า 488) ซึ่งไม่อ็อพชันการรีบูตใหม่และอ็อพชันถาวร แฟล็ก **-r** และ **-p** ตามลำดับ ของคำสั่งการปรับจะมีความหมาย เนื่องจากเนื้อหาของไฟล์ไม่ได้ใช้ในเวลาที่รีบูต คำสั่งการปรับจะไม่ควบคุมค่ารีบูตของพารามิเตอร์ เหมือนกับคำสั่งที่อยู่ในโหมดความเข้ากันไม่ได้ พารามิเตอร์ชนิด *Bosboot* จะถูกสงวนไว้ในระหว่างการโอนย้าย เก็บในไฟล์ /etc/tunables/nextboot และสามารถปรับแต่งได้โดยใช้อ็อพชัน **-r** ซึ่งคุณกำลังรันอยู่ในโหมดความเข้ากันไม่ได้ หรือไม่ ห้ามลบไฟล์ /etc/tunables/nextboot

โหมดความเข้ากันได้จะควบคุมแอ็ตทริบิวต์ **sys0** ใหม่ที่เรียกว่า **pre520tune** ซึ่งจะถูกตั้งค่าเป็น enable ในรหว่างการติดตั้ง การโอนย้าย ในโหมดปิดใช้งาน การเรียกที่ฝังไว้เพื่อเรียกคำสั่งการปรับในสคริปต์จะเรียกในระหว่างการรีบูตจะถูกเขียนโดยเนื้อหาของไฟล์ nextboot ค่าติดตั้งปัจจุบันของแอ็ตทริบิวต์ **pre520tune** สามารถดูได้โดยรันคำสั่งต่อไปนี้ :

```
# lsattr -E -l sys0
```

และ เปลี่ยนแปลงโดยใช้คำสั่งต่อไปนี้:

```
# chdev -l sys0 -a pre520tune=disable
```

เมื่อปิดใช้งานโหมดความเข้ากันได้ พารามิเตอร์คำสั่ง **no** ต่อไปนี้จะเป็นชนิดทั้งหมดของ *รีบูต* ซึ่งหมายความว่า พารามิเตอร์เหล่านี้อาจเปลี่ยนแปลงได้ในระหว่างการรีบูต ซึ่งไม่สามารถเปลี่ยนแปลงได้โดยไม่ได้ใช้แฟล็ก **–r** :

- arptab_bsiz
- arptab_nb
- extendednetstats
- ifsize
- inet_stack_size
- ipqmaxlen
- nstrpush
- pseintrstack

การสวิตช์ไปเป็นโหมดความเข้ากันไม่ได้ขณะที่สงวนค่าติดตั้งการรีบูตปัจจุบัน ซึ่งสามารถทำได้โดยเปลี่ยนแอ็ตทริบิวต์ **pre520tune** ในครั้งแรก จากนั้นรันคำสั่งต่อไปนี้:

```
# tunrestore -r -f lastboot
```

ซึ่งจะทำสำเนาเนื้อหาของไฟล์ `lastboot` ในไฟล์ `nextboot` สำหรับรายละเอียดเกี่ยวกับโหมด การปรับ โปรดดูส่วนการปรับเคอร์เนลใน *Performance Tools Guide and Reference*

โพรซีเดอร์การกู้คืนระบบ AIX

ถ้าเครื่องไม่มั่นคงหลังจากรีบูตและแอ็ตทริบิวต์ **pre520tune** มีการตั้งค่าเป็น `enable` ให้ลบการเรียกที่ไม่ถูกต้องในคำสั่งการปรับ ออกจากสคริปต์ที่เรียกในระหว่างรีบูต

เพื่อตรวจหาพารามิเตอร์ที่มีการตั้งค่าในระหว่างรีบูต ให้ดูที่ไฟล์ `/etc/tunables/lastboot` และค้นหาพารามิเตอร์ที่ไม่มีการทำเครื่องหมายด้วย # DEFAULT VALUE หากต้องการข้อมูลเพิ่มเติมเกี่ยวกับเนื้อหาของไฟล์ที่ ปรับได้ ให้ดูส่วน รูปแบบไฟล์ที่ปรับได้ ใน *Files Reference*

หรือ ถ้าต้องการรีเซ็ตพารามิเตอร์ที่ปรับได้ทั้งหมดเป็นค่าดีฟอลต์ ให้ปฏิบัติตามขั้นตอน ต่อไปนี้:

1. ลบไฟล์ `/etc/tunables/nextboot`
2. ตั้งค่าแอ็ตทริบิวต์ **pre520tune** เป็น `disable`
3. รันคำสั่ง **bosboot**
4. รีบูตเครื่อง

พารามิเตอร์ที่ปรับได้ของตัวจัดตารางเวลาและการควบคุมโหลดหน่วยความจำ

มีพารามิเตอร์หลายตัวที่เกี่ยวข้องกับตัวจัดตารางเวลาและการควบคุม โหลดหน่วยความจำ

พารามิเตอร์ที่ปรับได้ของตัวจัดตารางเวลาและการควบคุมโหลดหน่วยความจำส่วนใหญ่มีการอธิบายไว้โดยสมบูรณ์ในหน้าหลัก **schedo** พารามิเตอร์ที่เกี่ยวข้องอื่นสองสามตัว มีดังต่อไปนี้:

1. การปรับแต่งพารามิเตอร์ **maxuproc**:

Item	Descriptor
วัตถุประสงค์:	ระบุจำนวนสูงสุดของ กระบวนการต่อ ID ผู้ใช้
ค่า:	ดีฟอลต์: 40; ช่วง: 1 ถึง 131072
การแสดงผล:	**lsattr -E -l sys0 -a maxuproc**
การเปลี่ยนแปลง:	**chdev -l sys0 -a maxuproc=***NewValue*การเปลี่ยนแปลงมีผลทันทีและ มีการเก็บรักษาไว้เมื่อบูต ถ้าค่าลดลง แสดงว่ามีผลหลังจากบูตระบบแล้ว เท่านั้น
การวิเคราะห์:	ผู้ใช้ไม่สามารถ fork กระบวนการเพิ่มเติม ใดๆ ได้
การปรับ:	นี่เป็นตัวป้องกันไม่ให้ผู้ใช้ สร้างกระบวนการมากเกินไป

2. การปรับแต่งพารามิเตอร์ **ncargs**:

Item	Descriptor
วัตถุประสงค์:	ระบุขนาดสูงสุดที่ใช้ได้ของ รายการ ARG/ENV (ในบล็อก 4 KB) ในขณะกำลังรันรูทีนย่อย **exec( )**
ค่า:	ดีฟอลต์: 256; ช่วง: 256 ถึง 1024
การแสดงผล:	**lsattr -E -l sys0 -a ncargs**
การเปลี่ยนแปลง:	**chdev -l sys0 -a ncargs=***NewValue*การเปลี่ยนแปลงมีผลทันทีและ มีการเก็บรักษาไว้เมื่อบูต
การวิเคราะห์:	ผู้ใช้ไม่สามารถดำเนินการกระบวนการเพิ่มเติม ใดๆ ได้เนื่องจากรายการอาร์กิวเมนต์ที่ส่งผ่านไปยังการเรียกระบบ **exec( )** ยาวเกินไป ค่าดีฟอลต์ต่ำอาจทำให้บางโปรแกรมล้มเหลวโดยมีข้อความแสดงข้อผิดพลาด `arg list too long` ในกรณีนี้ คุณอาจลองเพิ่มค่า **ncargs** โดยใช้คำสั่ง **chdev** ข้างบน แล้วรีรันโปรแกรม
การปรับ:	นี่เป็นกลไกเพื่อป้องกันไม่ให้รูทีนย่อย **exec( )** ล้มเหลวถ้ารายการอาร์กิวเมนต์ยาวเกินไป โปรดทราบว่าการปรับเป็นค่า **ncargs** ที่สูงขึ้นสร้างข้อจำกัดเพิ่มเติม ในรีซอร์สหน่วยความจำของระบบ

พารามิเตอร์ที่ปรับได้ของ Virtual Memory Manager

คำสั่ง **vmo** จัดการพารามิเตอร์ที่ปรับได้ของ Virtual Memory Manager

สำหรับข้อมูลเพิ่มเติม โปรดดูคำสั่ง **vmo**

พารามิเตอร์ที่ปรับได้ของซิงโครนัส I/O

มีพารามิเตอร์ที่ปรับได้อยู่หลายตัวสำหรับซิงโครนัส I/O

พารามิเตอร์ที่ปรับได้ของซิงโครนัส I/O ส่วนใหญ่มีการอธิบายไว้โดยสมบูรณ์ในหน้าหลัก **ioo** พารามิเตอร์ที่เกี่ยวข้องอื่นสองสามตัว มีดังต่อไปนี้:

1. **maxbuf**

Item	Descriptor
วัตถุประสงค์:	จำนวนหน้า (4 KB) ในบัฟเฟอร์แคชของ บล็อก-I/O
ค่า:	ดีฟอลต์: 20; ช่วง: 20 ถึง 1000
การแสดงผล:	**lsattr -E -l sys0 -a maxbuf**
การเปลี่ยนแปลง:	**chdev -l sys0 -a maxbuf=***NewValue*การเปลี่ยนแปลงมีผลในทันทีและเป็นแบบถาวร ถ้ามีการใช้แฟล็ก **-T** การเปลี่ยนแปลงจะมีผล ในทันทีและคงอยู่จนกระทั่งบูตครั้งถัดไป ถ้ามีการใช้แฟล็ก **-P** การเปลี่ยนแปลงจะถูก เลื่อนออกไปจนกระทั่งบูตครั้งถัดไป และเป็นแบบถาวร
การวิเคราะห์:	ถ้าคำสั่ง **sar —b** แสดง `breads` หรือ `bwrites` ที่มี `%rcache` และ `%wcache` ต่ำ คุณอาจต้องการปรับพารามิเตอร์นี้
การปรับ:	โดยปกติแล้ว พารามิเตอร์นี้มีผลต่อประสิทธิภาพ ของระบบที่ I/O ปกติไม่ได้ใช้บัฟเฟอร์แคชของบล็อก-I/O น้อยมาก

อ้างอิงถึง: การปรับอะซิงโครนัสดิสก์ I/O

2. **maxpout**

Item	Descriptor
วัตถุประสงค์:	ระบุจำนวนสูงสุดของ I/O ที่ค้างอยู่ที่ไฟล์
ค่า:	ดีฟอลต์: 8193; ช่วง: 0 ถึง n (n ควรเป็นผลคูณของ 4, บวก 1)
การแสดงผล:	**lsattr -E -l sys0 -a maxpout**
การเปลี่ยนแปลง:	**chdev -l sys0 -a maxpout=***NewValue*การเปลี่ยนแปลงมีผลในทันทีและเป็นแบบถาวร ถ้ามีการใช้แฟล็ก **-T** การเปลี่ยนแปลงจะมีผล ในทันทีและคงอยู่จนกระทั่งบูตครั้งถัดไป ถ้ามีการใช้แฟล็ก **-P** การเปลี่ยนแปลงจะถูก เลื่อนออกไปจนกระทั่งบูตครั้งถัดไป และเป็นแบบถาวร
การวิเคราะห์:	ถ้าเวลาตอบกลับบนพื้นหน้าแย่ลง ในบางครั้งเมื่อโปรแกรมที่มีดิสก์เอาต์พุตตามลำดับจำนวนมากกำลังรันอยู่ ดิสก์ I/O อาจต้องมีการ paced หนักขึ้น ถ้าประสิทธิภาพตามลำดับ ด้อยลงมากจนไม่สามารถรับได้ I/O pacing อาจต้องถูกลดลง หรือปิดใช้งาน
การปรับ:	ถ้าประสิทธิภาพพื้นหน้าไม่สามารถ ยอมรับได้ ให้ลดค่าของทั้ง **maxpout** และ **minpout** ถ้าประสิทธิภาพตามลำดับด้อยลงมากจนไม่สามารถยอมรับได้ ให้เพิ่มหนึ่งหรือสองค่า หรือตั้งค่าทั้งสองเป็น 0 เพื่อปิดใช้งาน I/O pacing

3. **minpout**

Item	Descriptor
วัตถุประสงค์:	ระบุจุดซึ่งโปรแกรมที่มีค่า ถึง **maxpout** สามารถกลับมาบันทึกลงในไฟล์ต่อได้
ค่า:	ดีฟอลต์: 4096; ช่วง: 0 ถึง n (n ควรเป็นผลคูณของ 4 และควรน้อยกว่า **maxpout** อย่างน้อย 4)
การแสดงผล:	**lsattr -E -l sys0 -a minpout**
การเปลี่ยนแปลง:	**chdev -l sys0 -a minpout=***NewValue*การเปลี่ยนแปลงมีผลในทันทีและเป็นแบบถาวร ถ้ามีการใช้แฟล็ก **-T** การเปลี่ยนแปลงจะมีผล ในทันทีและคงอยู่จนกระทั่งบูตครั้งถัดไป ถ้ามีการใช้แฟล็ก **-P** การเปลี่ยนแปลงจะถูก เลื่อนออกไปจนกระทั่งบูตครั้งถัดไป และเป็นแบบถาวร
การวิเคราะห์:	ถ้าเวลาตอบกลับบนพื้นหน้าแย่ลง ในบางครั้งเมื่อโปรแกรมที่มีดิสก์เอาต์พุตตามลำดับจำนวนมากกำลังรันอยู่ ดิสก์ I/O อาจต้องมีการ paced หนักขึ้น ถ้าประสิทธิภาพตามลำดับ ด้อยลงมากจนไม่สามารถรับได้ I/O pacing อาจต้องถูกลดลง หรือปิดใช้งาน
การปรับ:	ถ้าประสิทธิภาพพื้นหน้าไม่สามารถ ยอมรับได้ ให้ลดค่าของทั้ง **maxpout** และ **minpout** ถ้าประสิทธิภาพตามลำดับด้อยลงมากจนไม่สามารถยอมรับได้ ให้เพิ่มหนึ่งหรือสองค่า หรือตั้งค่าทั้งสองเป็น 0 เพื่อปิดใช้งาน I/O pacing

4. **mount -o nointegrity**

Item	Descriptor
วัตถุประสงค์:	อ็อพชันการติดตั้งใหม่ (nointegrity) อาจช่วยให้ประสิทธิภาพของระบบไฟล์แบบโลคัลพัฒนาขึ้นสำหรับแอ็พพลิเคชันที่เน้นการบันทึกบางแอ็พพลิเคชัน โดยพื้นฐาน อ็อพชันนี้ใช้เพื่อตัดการบันทึกลงในบันทึก JFS หมายเหตุว่า ประสิทธิภาพที่พัฒนาขึ้นอาจได้มาโดยต้องแลกกับบูรณภาพของ metadata ดังนั้น ควรใช้อ็อพชันนี้ด้วยความระวังเป็นพิเศษ เนื่องจากล้มเหลวของระบบอาจทำให้ระบบไฟล์ที่ติดตั้งด้วยอ็อพชันนี้ไม่สามารถกู้คืนได้ อย่างไรก็ตาม บางคลาสของแอ็พพลิเคชัน ไม่ต้องการให้ข้อมูลไฟล์คงอยู่อย่างต่อเนื่องหลังจากระบบล้มเหลว และแอ็พพลิเคชันเหล่านี้อาจได้รับประโยชน์จากการใช้อ็อพชัน nointegrity สองตัวอย่างซึ่งระบบไฟล์ nointegrity อาจเป็นประโยชน์คือ สำหรับไฟล์ชั่วคราวของคอมไพเลอร์ และสำหรับการทำการติดตั้ง nonmigration หรือ mksysb

5. ขนาดพื้นที่ว่างการเพจ

Item	Descriptor
วัตถุประสงค์:	จำนวนของพื้นที่ว่างดิสก์ที่ต้องการ เพื่อจัดเก็บหน้าของหน่วยเก็บที่กำลังทำงาน
ค่า:	ดีฟอลต์: ขึ้นอยู่กับการตั้งค่าคอนฟิก; ช่วง: 32 MB ถึง n MB สำหรับ hd6, 16 MB ถึง n MB สำหรับ non-hd6
การแสดงผล:	**lsps -a mkps** หรือ **chps** หรือ **smitty pgsp**
การเปลี่ยนแปลง:	การเปลี่ยนแปลงมีผล ในทันทีและเป็นแบบถาวร อย่างไรก็ตาม พื้นที่ว่างการเพจไม่จำเป็นต้องนำไปใช้ในทันที
การวิเคราะห์:	รัน: **lsps -a** ถ้ากระบวนการถูก killed เนื่องจากขาดพื้นที่ว่างการเพจ ให้มอนิเตอร์สถานการณ์ ด้วยรูทีนย่อย **psdanger( )**
การปรับ:	ถ้าพื้นที่ว่างการเพจไม่เพียงพอที่จะ จัดการเวิร์กโหลดปกติ ให้เพิ่มพื้นที่ว่างการเพจใหม่บนฟิสิคัลวอลุ่มอื่น หรือทำให้พื้นที่ว่างการเพจที่มีอยู่ใหญ่ขึ้น

6. syncd Interval

Item	Descriptor
วัตถุประสงค์:	เวลาระหว่างการเรียก **sync( )** โดย **syncd**
ค่า:	ดีฟอลต์: 60; ช่วง: 1 ถึงจำนวนเต็มบวก ใดๆ
การแสดงผล:	**grep syncd** /sbin/rc.boot **vi** /sbin/rc.boot หรือ
การเปลี่ยนแปลง:	การเปลี่ยนแปลงมีผลในการบูตครั้งถัดไป และเป็นแบบถาวร วิธีการอื่นคือการใช้คำสั่ง **kill** เพื่อยุติ **syncd** daemon และรีสตาร์ทจากบรรทัดคำสั่ง ด้วยคำสั่ง /usr/sbin/syncd interval
การวิเคราะห์:	I/O ไปที่ไฟล์ถูกบล็อกเมื่อ **syncd** กำลัง รันอยู่
การปรับ:	ที่ระดับดีฟอลต์ พารามิเตอร์นี้มีผลต่อ ประสิทธิภาพน้อยมาก ไม่แนะนำให้เปลี่ยนแปลง การลดค่าใน **syncd interval** ลงอย่างมากเพื่อให้บูรณภาพข้อมูลดีขึ้น (ดังเช่นสำหรับ HACMP™) อาจทำให้ประสิทธิภาพด้อยลงในภายหลังได้

การเปลี่ยนค่าที่สามารถปรับแต่งได้สำหรับ I/O แบบอะซิงโครนัส

AIO ที่สามารถปรับแต่งได้ทั้งหมดจะมีค่าปัจจุบัน ค่าดีฟอลต์ ค่าต่ำสุด และค่าสูงสุดที่สามารถดูได้ด้วยคำสั่ง **ioo**

เฉพาะค่าปัจจุบันของค่าที่สามารถปรับแต่งได้เท่านั้นที่สามารถเปลี่ยนแปลงได้ด้วยคำสั่ง **ioo** ค่าอื่นๆ สามค่าจะไม่เปลี่ยนแปลง และจะแสดงเพื่อแจ้งให้คุณทราบถึงการโยงของ ค่าที่สามารถปรับแต่งได้ คุณสามารถเปลี่ยนค่าปัจจุบันของค่าที่สามารถปรับแต่งได้ทุกเวลา และทำให้ค่านั้นคงอยู่สำหรับการรีสตาร์ทระบบปฏิบัติการ ค่าที่สามารถปรับแต่งได้ทั้งหมด จะต้องเป็นไปตามกฎปกติ และอ็อพชันต้องถูกควบคุมด้วยคำสั่ง **ioo** ซึ่งอยู่ในชุดไฟล์ของทูลด้านประสิทธิภาพการทำงาน

ค่าที่สามารถปรับแต่งได้โดยไม่มีข้อจำกัดจะถูกสรุปไว้ในตารางต่อไปนี้:

ไอเท็ม	คำอธิบาย
minservers	บ่งชี้ถึงจำนวนต่ำสุดของการประมวลผลเคอร์เนลต่อตัวประมวลผล เฉพาะงานกับการประมวลผล AIO เนื่องจากการประมวลผลเคอร์เนลแต่ละตัวจะใช้หน่วยความจำ ค่า minservers ที่สามารถปรับแต่งได้เมื่อคูณด้วยจำนวนของตัวประมวลผล ต้องมีขนาดที่ไม่ใหญ่ เมื่อจำนวนของ AIO ที่คาดการณ์ไว้มีขนาดเล็ก ค่าดีฟอลต์สำหรับ minservers ที่สามารถปรับแต่งได้คือ 3
maxservers	บ่งชี้จำนวนสูงสุดของการประมวลผลเคอร์เนลต่อตัวประมวลผล ที่เฉพาะงานกับการประมวลผล AIO ค่าที่สามารถปรับแต่งได้นี้เมื่อคูณด้วยจำนวนของตัวประมวลผล จะบ่งชี้ถึงข้อจำกัดเกี่ยวกับคำร้องขอพาธ I/O ในการดำเนินการที่ช้าลงในแต่ละครั้ง และแสดงข้อจำกัดสำหรับ I/O ที่มีภาวะพร้อมกัน ค่าดีฟอลต์สำหรับ maxservers ที่สามารถปรับแต่งได้คือ 30
maxreqs	บ่งชี้จำนวนสูงสุดของคำร้องขอ AIO ที่สามารถค้างอยู่ได้ในหนึ่งครั้ง คำร้องขอจะสอดแทรกจำนวนเหล่านั้นที่อยู่ในการดำเนินการ พร้อมกับจำนวนที่รอที่จะเริ่มต้น จำนวนสูงสุดของคำร้องขอ AIO ไม่สามารถมีค่าน้อยกว่า AIO_MAX ตามที่ได้กำหนดไว้ในไฟล์ /usr/include/sys/limits.h แต่ค่านั้นสามารถมีค่าที่มากกว่าได้ ซึ่งเป็นค่าที่เหมาะสำหรับระบบด้วยวอลุ่มที่สูงของ AIO เพื่อให้มีจำนวนสูงสุดของคำร้องขอ AIO ที่มีขนาดใหญ่กว่า AIO_MAX ค่าดีฟอลต์ สำหรับ maxreqs ที่สามารถปรับแต่งได้คือ 16384
server_inactivity	บ่งชี้ค่าหมดเวลาใช้งานในหน่วยวินาที ที่เป็นสาเหตุทำให้เซิร์ฟเวอร์ออกหากเซิร์ฟเวอร์ไม่ได้ทำงาน (sleeping) โดยไม่ให้บริการคำร้องขอ AIO ถ้าการออกเป็นสาเหตุทำให้จำนวนทั้งหมดของเซิร์ฟเวอร์ตกลง minservers * จำนวนของ CPU ซึ่งเซิร์ฟเวอร์กลับสู่สถานะ sleep เพื่อรอ AIO ให้บริการ กลไกนี้ช่วยให้เพิ่มผลการทำงานของระบบทั้งหมด โดยลดจำนวนของการประมวลผลที่ sleeping ซึ่งไม่ได้ถูกนำมาใช้เพื่อให้บริการคำร้องขอ AIO ค่าดีฟอลต์สำหรับ server_inactivity ที่สามารถปรับแต่งได้คือ 300

พารามิเตอร์ที่สามารถปรับแต่งดิสก์และดิสก์อะแด็ปเตอร์

มีพารามิเตอร์ที่สามารถปรับแต่งดิสก์จำนวนมากและดิสก์อะแด็ปเตอร์ของเคอร์เนลใน AIX

1. ข้อจำกัดเกี่ยวกับคำร้องขอดิสก์อะแด็ปเตอร์ที่คงค้าง

ไอเท็ม	คำอธิบาย
วัตถุประสงค์:	จำนวนสูงสุดของคำร้องขอ ที่สามารถคงค้างบนบัส SCSI ได้ (ใช้ได้กับ SCSI-2 Fast/Wide Adapter)
ค่า:	ดีฟอลต์: 40; ช่วง: 40 ถึง 128
การแสดง:	**lsattr -E -l scsi*n* -a num_cmd_elems**
การเปลี่ยน:	**chdev -l scsi*n* -a num_cmd_elems=***NewValue*การเปลี่ยนมีผลในทันที เป็นแบบถาวร ถ้าใช้แฟล็ก **-T** การเปลี่ยนแปลงจะมีผลในทันที และเป็นการเปลี่ยนแปลงล่าสุดจนกว่าจะบูตในครั้งถัดไป ถ้าใช้แฟล็ก **-P** การเปลี่ยนแปลงจะถูกเลื่อนออกไป จนกว่าจะบูตในครั้งถัดไป และจะเป็นการเปลี่ยนแปลงแบบถาวร
การวินิจฉัย:	แอ็พพลิเคชันที่ดำเนินการกับการเขียนขนาดใหญ่ เพื่อ strip โลจิคัลวอลุ่มที่ไม่ได้ขอรับอัตราทรูพุตที่ต้องการ
การปรับ:	ค่าควรมีค่าเท่ากับจำนวนของฟิสิคัลไดร์ฟ (ซึ่งรวมถึงค่าที่อยู่ในดิสก์อาร์เรย์) บน SCSI บัส คูณด้วยความลึกของคิวของไดร์ฟแต่ละไดร์ฟ

2. ความลึกของคิวดิสก์ไดร์ฟ

ไอเท็ม	คำอธิบาย
วัตถุประสงค์:	จำนวนสูงสุดของคำร้องขอดิสก์ไดร์ฟ สามารถพักได้ในคิว
ค่า:	ดีฟอลต์: IBM disks=3; Non-IBM disks=0; Range: ระบุโดยผู้ผลิต
การแสดง:	**lsattr -E -l hdisk*n***
การเปลี่ยน:	**chdev -l hdisk*n* -a q_type=simple -a queue_depth=***NewValue*** การเปลี่ยนแปลงมีผลในทันทีและเป็นการเปลี่ยนแปลงแบบถาวร ถ้าใช้แฟล็ก **-T** การเปลี่ยนแปลงจะมีผลในทันที และเป็นการเปลี่ยนแปลงล่าสุดจนกว่าจะบูตในครั้งถัดไป ถ้าใช้แฟล็ก **-P** การเปลี่ยนแปลงจะถูกเลื่อนออกไป จนกว่าจะบูตในครั้งถัดไป และจะเป็นการเปลี่ยนแปลงแบบถาวร
การวินิจฉัย:	N/A
การปรับ:	ถ้าดิสก์ไดร์ฟที่ไม่ใช่ของ IBM มีความสามารถของการจัดคิวคำร้องขอ ให้ทำการเปลี่ยนแปลงเพื่อมั่นใจว่า ระบบปฏิบัติการใช้ประโยชน์ของ ความสามารถนี้

โปรดอ้างถึง: ค่าติดตั้ง SCSI-Adapter และข้อจำกัดเกี่ยวกับคิวอุปกรณ์ดิสก์

พารามิเตอร์ที่สามารถปรับแต่งการสื่อสารระหว่างการประมวลผลได้

AIX มีพารามิเตอร์ ที่สามารถปรับแต่งการสื่อสารระหว่างการประมวลผลได้

1. การปรับแต่งพารามิเตอร์ **msgmax**:

Item	Descriptor
วัตถุประสงค์:	ระบุขนาดของข้อความสูงสุด
ค่า:	เป็นแบบไดนามิกที่มีค่าสูงสุด 4 MB
การแสดง:	N/A
การเปลี่ยน:	N/A
การวินิจฉัย:	N/A
การปรับ:	ไม่จำเป็นต้องปรับ เนื่องจากจะถูกปรับเปลี่ยนแบบไดนามิกตามความต้องการโดยเคอร์เนล

2. การปรับแต่งพารามิเตอร์ **msgmnb**:

Item	Descriptor
วัตถุประสงค์:	ระบุจำนวนไบต์สูงสุด สำหรับคิว
ค่า:	เป็นแบบไดนามิกที่มีค่าสูงสุด 4 MB
การแสดง:	N/A
การเปลี่ยน:	N/A
การวินิจฉัย:	N/A
การปรับ:	ไม่จำเป็นต้องปรับ เนื่องจากจะถูกปรับเปลี่ยนแบบไดนามิกตามความต้องการโดยเคอร์เนล

3. การปรบัแต่ง พารามิเตอร์ **msgmni**:

Item	Descriptor
วัตถุประสงค์:	ระบุจำนวนสูงสุดของ message queue ID
ค่า:	เป็นแบบไดนามิกที่มีค่าสูงสุด 131072
การแสดง:	N/A
การเปลี่ยน:	N/A
การวินิจฉัย:	N/A
การปรับ:	ไม่จำเป็นต้องปรับ เนื่องจากจะถูกปรับเปลี่ยนแบบไดนามิกตามความต้องการโดยเคอร์เนล

4. การปรบัแต่ง พารามิเตอร์ **msgmnm**:

Item	Descriptor
วัตถุประสงค์:	ระบุจำนวนสูงสุดของข้อความ ต่อคิว
ค่า:	เป็นแบบไดนามิกที่มีค่าสูงสุด 524288
การแสดง:	N/A
การเปลี่ยน:	N/A
การวินิจฉัย:	N/A

Item	Descriptor
การปรับ:	ไม่จำเป็นต้องปรับ เนื่องจากจะถูกปรับเปลี่ยนแบบไดนามิกตามความต้องการโดยเคอร์เนล

5. การปรับแต่ง พารามิเตอร์ **semaem**:

Item	Descriptor
วัตถุประสงค์:	ระบุค่าสูงสุดสำหรับการปรับเปลี่ยน เกี่ยวกับการออก
ค่า:	เป็นแบบไดนามิกที่มีค่าสูงสุด 16384
การแสดง:	N/A
การเปลี่ยน:	N/A
การวินิจฉัย:	N/A
การปรับ:	ไม่จำเป็นต้องปรับ เนื่องจากจะถูกปรับเปลี่ยนแบบไดนามิกตามความต้องการโดยเคอร์เนล

6. การปรับแต่ง พารามิเตอร์ **semmni**:

Item	Descriptor
วัตถุประสงค์:	ระบุจำนวนสูงสุดของ ID อุปกรณ์สัญญาณ
ค่า:	เป็นแบบไดนามิกที่มีค่าสูงสุด 131072
การแสดง:	N/A
การเปลี่ยน:	N/A
การวินิจฉัย:	N/A
การปรับ:	ไม่จำเป็นต้องปรับ เนื่องจากจะถูกปรับเปลี่ยนแบบไดนามิกตามความต้องการโดยเคอร์เนล

7. การปรับแต่ง พารามิเตอร์ **semmsl**:

Item	Descriptor
วัตถุประสงค์:	ระบุจำนวนสูงสุดของอุปกรณ์สัญญาณ ต่อ ID
ค่า:	เป็นแบบไดนามิกที่มีค่าสูงสุด 65535
การแสดง:	N/A
การเปลี่ยน:	N/A
การวินิจฉัย:	N/A
การปรับ:	ไม่จำเป็นต้องปรับ เนื่องจากจะถูกปรับเปลี่ยนแบบไดนามิกตามความต้องการโดยเคอร์เนล

8. การปรับแต่ง **semopm** parameter:

Item	Descriptor
วัตถุประสงค์:	ระบุจำนวนสูงสุดของการดำเนินการต่อการเรียก **semop( )**
ค่า:	เป็นแบบไดนามิกที่มีค่าสูงสุด 1024
การแสดง:	N/A
การเปลี่ยน:	N/A
การวินิจฉัย:	N/A
การปรับ:	ไม่จำเป็นต้องปรับ เนื่องจากจะถูกปรับเปลี่ยนแบบไดนามิกตามความต้องการโดยเคอร์เนล

9. การปรบัแต่ง พารามิเตอร์ **semume:**

Item	Descriptor
วัตถุประสงค์:	ระบุจำนวนสูงสุดของรายการเลิกทำ ต่อการประมวลผล
ค่า:	เป็นแบบไดนามิกที่มีค่าสูงสุด 1024
การแสดง:	N/A
การเปลี่ยน:	N/A
การวินิจฉัย:	N/A
การปรับ:	ไม่จำเป็นต้องปรับ เนื่องจากจะถูกปรับเปลี่ยนแบบไดนามิกตามความต้องการโดยเคอร์เนล

10. การปรบัแต่ง พารามิเตอร์ **semvmx:**

Item	Descriptor
วัตถุประสงค์:	ระบุค่าสูงสุดของอุปกรณ์สัญญาณ
ค่า:	เป็นแบบไดนามิกที่มีค่าสูงสุด 32767
การแสดง:	N/A
การเปลี่ยน:	N/A
การวินิจฉัย:	N/A
การปรับ:	ไม่จำเป็นต้องปรับ เนื่องจากจะถูกปรับเปลี่ยนแบบไดนามิกตามความต้องการโดยเคอร์เนล

11. การปรบัแต่ง พารามิเตอร์ **shmmax:**

Item	Descriptor
วัตถุประสงค์:	ระบุขนาดสูงสุดของเซ็กเมนต์หน่วยความจำ ที่แบ่งใช้
ค่า:	เป็นแบบไดนามิกที่มีค่าสูงสุด 256 MB สำหรับการประมวลผลแบบ 32 บิตและ 0x80000000u สำหรับ 64 บิต
การแสดง:	N/A
การเปลี่ยน:	N/A
การวินิจฉัย:	N/A
การปรับ:	ไม่จำเป็นต้องปรับ เนื่องจากจะถูกปรับเปลี่ยนแบบไดนามิกตามความต้องการโดยเคอร์เนล

12. การปรับแต่งพารามิเตอร์ **shmmin**:

Item	Descriptor
วัตถุประสงค์:	ระบุขนาดต่ำสุดของเซ็กเมนต์หน่วยความจำ ที่แบ่งใช้
ค่า:	เป็นแบบไดนามิกที่มีค่าต่ำสุดคือ 1
การแสดง:	N/A
การเปลี่ยน:	N/A
การวินิจฉัย:	N/A
การปรับ:	ไม่จำเป็นต้องปรับ เนื่องจากจะถูกปรับเปลี่ยนแบบไดนามิกตามความต้องการโดยเคอร์เนล

13. การปรับแต่งพารามิเตอร์ **shmmni**:

Item	Descriptor
วัตถุประสงค์:	ระบุจำนวนสูงสุดของ ID หน่วยความจำที่แบ่งใช้
ค่า:	ไดนามิกที่มีค่าสูงสุด 1048576
การแสดง:	N/A
การเปลี่ยน:	N/A
การวินิจฉัย:	N/A
การปรับ:	ไม่จำเป็นต้องปรับ เนื่องจากจะถูกปรับเปลี่ยนแบบไดนามิกตามความต้องการโดยเคอร์เนล

พารามิเตอร์ที่ปรับได้ของเครือข่าย

พารามิเตอร์ที่ปรับได้ของเครือข่ายมีอยู่สองกลุ่มคือ: อ็อพชันเครือข่าย และอ็อพชัน NFS

พารามิเตอร์ที่ปรับได้ของอ็อพชันเครือข่าย

มีพารามิเตอร์หลายตัวที่เกี่ยวข้องกับพารามิเตอร์ที่ปรับได้ของอ็อพชัน เครือข่ายใน AIX

พารามิเตอร์ที่ปรับได้ของอ็อพชันเครือข่ายส่วนใหญ่มีการอธิบายไว้โดย สมบูรณ์ในหน้าหลัก **no** ใน *Commands Reference, Volume 4* หากต้องการข้อมูลเกี่ยวกับพารามิเตอร์ที่ปรับได้ของเครือข่ายซึ่งต้องการความสนใจเป็นพิเศษ ในสภาพแวดล้อม SP ให้ดู *RS/6000 การปรับประสิทธิภาพของระบบ SP* พารามิเตอร์ที่เกี่ยวข้องอื่นสองสามตัวมีดังต่อไปนี้:

1. **maxmbuf**

ไอเท็ม	Descriptor
วัตถุประสงค์:	กิโลไบต์สูงสุดของหน่วยความจำจริง ที่ใช้ได้สำหรับ MBUFS
ค่า:	ดีฟอลต์: 0, ช่วง: x ถึง y
การแสดงผล:	**lsattr -E -l sys0 -a maxmbuf**
การเปลี่ยนแปลง:	**chdev -l sys0 -a maxmbuf=***NewValue*
	การเปลี่ยนแปลงมีผล ในทันทีและเป็นแบบถาวร ถ้าใช้แฟล็ก **-T** การเปลี่ยนแปลงจะมีผลในทันที และเป็นการเปลี่ยนแปลงล่าสุดจนกว่าจะบูตในครั้งถัดไป ถ้าใช้แฟล็ก **-P** การเปลี่ยนแปลงจะถูกเลื่อนออกไป จนกว่าจะบูตในครั้งถัดไป และจะเป็นการเปลี่ยนแปลงแบบถาวร
การวิเคราะห์:	N/A
การปรับ:	ถ้า *maxmbuf* มากกว่า 0 ระบบจะใช้ค่า *maxmbuf* โดยไม่คำนึงถึง ค่าของ *thewall* ขีดจำกัดสูงสุดของ mbufs คือค่าที่มากกว่าของ *maxmbuf* หรือ *thewall*

ไอเท็ม **Descriptor**
อ้างอิงถึง: “คำสั่ง netstat -m เพื่อมอนิเตอร์ mbuf พูล” ในหน้า 321

2. MTU

ไอเท็ม **Descriptor**
วัตถุประสงค์: จำกัดขนาดของแพ็กเก็ตที่ จะส่งผ่านบนเครือข่าย
ค่า: ดีฟอลต์: ขึ้นอยู่กับการตั้งค่าคอนฟิก
การแสดงผล: **lsattr -E -l** *interface_name*
การเปลี่ยนแปลง: **chdev -l** *interface_name* **-a mtu=***NewValue*

ด้วยคำสั่ง **chdev** ไม่สามารถเปลี่ยนอินเตอร์เฟส ในขณะที่ถูกใช้งานอยู่ได้ การเปลี่ยนแปลงมีผลเมื่อรีบูต วิธีการอื่นมีดังนี้: **ifconfig** *interface_name* **mtu** *NewValue* วิธีนี้ เปลี่ยนขนาด MTU บนระบบที่กำลังรันอยู่ แต่จะไม่เก็บรักษาค่า เมื่อรีบูตระบบ
การวิเคราะห์: สถิติการแบ่งเฟรกเมนต์แพ็กเก็ต
การปรับ: เพิ่มขนาด MTU สำหรับอินเตอร์เฟส เครือข่าย สำหรับอะแด็ปเตอร์กิกะบิตอีเทอร์เน็ต ให้ใช้แอ็ตทริบิวต์อุปกรณ์ **jumbo_frames=yes** เพื่อ เปิดใช้งาน jumbo frames (การตั้งค่า MTU เป็น 9000 บนอินเตอร์เฟสเพียงอย่างเดียวไม่เพียงพอ)
อ้างอิงถึง: “การปรับประสิทธิภาพ TCP และ UDP” ในหน้า 282

3. rfc1323

ไอเท็ม **Descriptor**
วัตถุประสงค์: เปิดใช้งานการเพิ่มประสิทธิภาพ TCP ตามที่ระบุโดย RFC 1323 (ส่วนขยาย TCP สำหรับประสิทธิภาพสูง) ค่า 1 บ่งชี้ว่า *tcp_sendspace* และ *tcp_recvspace* อาจเกินกว่า 64 KB ได้
ค่า: ดีฟอลต์: 0; ช่วง 0 ถึง 1
การแสดงผล: **lsattr -El** *interface* หรือ **ifconfig** *interface*
การเปลี่ยนแปลง: **ifconfig** *interface* **rfc1323** *NewValue*OR **chdev -l** *interface* **-a rfc1323=***NewValue*

คำสั่ง **ifconfig** ตั้งค่าเป็นการชั่วคราว ซึ่งมีประโยชน์สำหรับการทดสอบ คำสั่ง **chdev** เปลี่ยน ODM ดังนั้นจึงส่งคืนค่าที่กำหนดเองหลังจากรีบูตระบบ
การวิเคราะห์: N/A
การปรับ: ค่าดีฟอลต์ 0 ปิดใช้งานการเพิ่มประสิทธิภาพ RFC บนสเกลทั่วทั้งระบบ ค่า 1 ระบุว่าการเชื่อมต่อ TCP ทั้งหมดจะพยายามเจรจาการเพิ่มประสิทธิภาพ RFC SOCKETS แอ็พพลิเคชันสามารถแทนที่พฤติกรรมดีฟอลต์บนการเชื่อมต่อ TCP แต่ละรายการ ได้โดยใช้รูทีนย่อย **setsockopt( )** นี่เป็นแอ็ตทริบิวต์แบบรันไทม์ ทำการเปลี่ยนแปลง ก่อนที่จะพยายามตั้งค่า *tcp_sendspace* และ *tcp_recvspace* เป็น มากกว่า 64 KB
อ้างอิงถึง: “การปรับเวิร์กโหลด TCP” ในหน้า 298

4. tcp_mssdflt

ไอเท็ม **Descriptor**
วัตถุประสงค์: ขนาดเซกเมนต์สูงสุดดีฟอลต์ที่ใช้ในการสื่อสารกับเครือข่ายแบบรีโมต
ค่า: ดีฟอลต์: 512 ไบต์
การแสดงผล: **lsattr -El** *interface* หรือ **ifconfig** *interface*
การเปลี่ยนแปลง: **ifconfig** *interface* **tcp_mssdflt** *NewValue*OR **chdev -l** *interface* **-a tcp_mssdflt=***NewValue*

คำสั่ง **ifconfig** ตั้งค่าเป็นการชั่วคราว ซึ่งมีประโยชน์สำหรับการทดสอบ คำสั่ง **chdev** เปลี่ยน ODM ดังนั้นจึงส่งคืนค่าที่กำหนดเองหลังจากรีบูตระบบ
การวิเคราะห์: N/A
การปรับ: *tcp_mssdflt* มีการใช้ ถ้าการค้นหาพาธ MTU ไม่ได้เปิดใช้งาน หรือการค้นหาพาธ MTU ไม่พบพาธ MTU การจำกัด ข้อมูลเป็น (MTU - 52) ไบต์ช่วยให้มั่นใจว่าจะส่งเฉพาะแพ็กเก็ตแบบเต็มเท่านั้น ถ้าเป็นไปได้ นี่เป็นแอ็ตทริบิวต์แบบรันไทม์
อ้างอิงถึง: “การปรับขนาดเซกเมนต์สูงสุดของ TCP” ในหน้า 315

5. tcp_nodelay

ไอเท็ม	**Descriptor**
วัตถุประสงค์:	ระบุว่าซ็อกเก็ตที่ใช้ TCP บนอินเตอร์เฟสนี้ต้องปฏิบัติตามขั้นตอนวิธี Nagle เมื่อส่งข้อมูลโดยค่าดีฟอลต์ TCP ปฏิบัติตามขั้นตอนวิธี Nagle
ค่า:	ดีฟอลต์: 0; ช่วง 0 ถึง 1
การแสดงผล:	**lsattr -El** *interface* หรือ **ifconfig** *interface*
การเปลี่ยนแปลง:	**ifconfig** *interface* **tcp_nodelay** *NewValue*OR **chdev -l** *interface* **-a tcp_nodelay=***NewValue*
	คำสั่ง **ifconfig** ตั้งค่าเป็นการชั่วคราว ซึ่งมีประโยชน์สำหรับการทดสอบ คำสั่ง **chdev** เปลี่ยน ODM ดังนั้นจึงส่งคืนค่าที่กำหนดเองหลังจากรีบูตระบบ
การวิเคราะห์:	N/A
การปรับ:	นี่เป็นอ็อพชัน Interface-Specific Network Option (ISNO หรืออ็อพชันเครือข่ายเฉพาะอินเตอร์เฟส)
อ้างอิงถึง:	“Interface-Specific Network Options” ในหน้า 293

6. tcp_recvspace

ไอเท็ม	**Descriptor**
วัตถุประสงค์:	ระบุขนาดบัฟเฟอร์ของซ็อกเก็ตดีฟอลต์ของระบบ สำหรับการรับข้อมูล ค่านี้มีผลกระทบต่อขนาดหน้าต่างที่ใช้โดย TCP
ค่า:	ดีฟอลต์: 16384 ไบต์
การแสดงผล:	**lsattr -El** *interface* หรือ **ifconfig** *interface*
การเปลี่ยนแปลง:	**ifconfig** *interface* **tcp_recvspace** *NewValue*OR **chdev -l** *interface* **-a tcp_recvspace=***NewValue*
	คำสั่ง **ifconfig** ตั้งค่าเป็นการชั่วคราว ซึ่งมีประโยชน์สำหรับการทดสอบ คำสั่ง **chdev** เปลี่ยน ODM ดังนั้นจึงส่งคืนค่าที่กำหนดเองหลังจากรีบูตระบบ
การวิเคราะห์:	N/A
การปรับ:	การตั้งค่าขนาดบัฟเฟอร์ของซ็อกเก็ต เป็น 16 KB (16 384) ช่วยปรับปรุงประสิทธิภาพบนเครือข่ายอีเทอร์เน็ตและ Token-Ring มาตรฐาน ค่าดีฟอลต์คือ 16 384 เครือข่ายที่มีแบนด์วิธต่ำกว่า เช่น Serial Line Internet Protocol (SLIP) หรือเครือข่ายที่มีแบนด์วิธสูงกว่า เช่น Serial Optical Link ควรมีขนาดบัฟเฟอร์ที่เหมาะสมแตกต่างออกไป ขนาดบัฟเฟอร์ที่เหมาะสมเป็นผลมาจากแบนด์วิธของสื่อบันทึกและเวลาเดินทางไปกลับโดยเฉลี่ย ของแพ็กเก็ต แอ็ตทริบิวต์ *tcp_recvspace* ต้องระบุขนาดบัฟเฟอร์ของซ็อกเก็ตที่น้อยกว่าหรือเท่ากับค่าติดตั้งของแอ็ตทริบิวต์ **sb_max** นี่เป็นแอ็ตทริบิวต์แบบไดนามิก แต่สำหรับ daemons ที่เริ่มต้นโดย **inetd** daemon ให้รันคำสั่งดังต่อไปนี้: • **stopsrc -s inetd** • **startsrc -s inetd**
อ้างอิงถึง:	“การปรับเวิร์กโหลด TCP” ในหน้า 298

7. tcp_sendspace

ไอเท็ม	**Descriptor**
วัตถุประสงค์:	ระบุขนาดบัฟเฟอร์ของซ็อกเก็ตดีฟอลต์ของระบบ สำหรับการส่งข้อมูล
ค่า:	ดีฟอลต์: 16384 ไบต์
การแสดงผล:	**lsattr -El** *interface* หรือ **ifconfig** *interface*
การเปลี่ยนแปลง:	**ifconfig** *interface* **tcp_sendspace** *NewValue*OR **chdev -l** *interface* **-a tcp_sendspace=***NewValue*
	คำสั่ง **ifconfig** ตั้งค่าเป็นการชั่วคราว ซึ่งมีประโยชน์สำหรับการทดสอบ คำสั่ง **chdev** เปลี่ยน ODM ดังนั้นจึงส่งคืนค่าที่กำหนดเองหลังจากรีบูตระบบ
การวิเคราะห์:	N/A

ไอเท็ม **Descriptor**
การปรับ: ค่านี้มีผลกระทบต่อขนาดหน้าต่างที่ใช้โดย TCP การตั้งค่าขนาดบัฟเฟอร์ของซ็อกเก็ต เป็น 16 KB (16 384) ช่วยปรับปรุงประสิทธิภาพบนเครือข่ายอีเทอร์เน็ตและ Token-Ring มาตรฐาน ค่าดีฟอลต์คือ 16 384 เครือข่ายที่มีแบนด์วิธต่ำกว่า เช่น Serial Line Internet Protocol (SLIP) หรือเครือข่ายที่มีแบนด์วิธสูงกว่า เช่น Serial Optical Link ควรมีขนาดบัฟเฟอร์ที่เหมาะสมแตกต่างออกไป ขนาดบัฟเฟอร์ที่เหมาะสมเป็นผลมาจากแบนด์วิธของ สื่อบันทึกและเวลาเดินทางไปกลับโดยเฉลี่ยของแพ็กเก็ต: `optimum_window=bandwidth * average_round_trip_time`

แอ็ตทริบิวต์ **tcp_sendspace** ต้องระบุ ขนาดบัฟเฟอร์ของซ็อกเก็ตที่น้อยกว่าหรือเท่ากับค่าติดตั้งของแอ็ตทริบิวต์ **sb_max** พารามิเตอร์ **tcp_sendspace** เป็นแอ็ตทริบิวต์แบบไดนามิก แต่สำหรับ daemons ที่เริ่มต้นโดย **inetd** daemon ให้รันคำสั่งดังต่อไปนี้:

- **stopsrc -s inetd**
- **startsrc -s inetd**

อ้างอิงถึง: “การปรับเวิร์กโหลด TCP” ในหน้า 298

8. **use_sndbufpool**

ไอเท็ม **Descriptor**
วัตถุประสงค์: ระบุว่าควรจะใช้บัฟเฟอร์พูลการส่ง สำหรับซ็อกเก็ตหรือไม่
ค่า: ดีฟอลต์: 1
การแสดงผล: **netstat -m**
การเปลี่ยนแปลง: อ็อพชันนี้สามารถเปิดใช้งานได้โดยการ ตั้งค่าเป็น 1 หรือปิดใช้งานโดยการตั้งค่าเป็น 0
การวิเคราะห์: N/A
การปรับ: เป็นอ็อพชันบูลีน เวลาโหลด

9. **xmt_que_size**

ไอเท็ม **Descriptor**
วัตถุประสงค์: ระบุจำนวนสูงสุดของ บัฟเฟอร์การส่งที่สามารถจัดคิวสำหรับอินเตอร์เฟสได้
ค่า: ดีฟอลต์: ขึ้นอยู่กับการตั้งค่าคอนฟิก
การแสดงผล: **lsattr -E -l** *interface_name*
การเปลี่ยนแปลง: **ifconfig** *interface_name* **detach** chdev **-l** *interface_name* **-a***que_size_name*=*NewValue* **ifconfig** *interface_name hostname* **up**

ไม่สามารถเปลี่ยนได้ในขณะที่ อินเตอร์เฟสถูกใช้งานอยู่ การเปลี่ยนแปลงมีผลเมื่อรีบูต
การวิเคราะห์: **netstat -i** (`Oerr > 0`)
การปรับ: เพิ่มขนาด
อ้างอิงถึง: “คำสั่ง netstat” ในหน้า 328

พารามิเตอร์ที่ปรับได้ของอ็อพชัน NFS

มีพารามิเตอร์หลายตัวที่เกี่ยวข้องกับพารามิเตอร์ที่ปรับได้ของอ็อพชัน NFS ใน AIX

พารามิเตอร์ที่ปรับได้ของอ็อพชัน NFS ส่วนใหญ่มีการอธิบายไว้โดยสมบูรณ์ในหน้าหลัก **nfso** พารามิเตอร์ที่เกี่ยวข้องอื่นสองสามตัว มีดังต่อไปนี้:

1. **biod Count**

ไอเท็ม	Descriptor
วัตถุประสงค์:	จำนวนของกระบวนการ biod ที่มีอยู่สำหรับ การจัดการร้องขอ NFS บนไคลเอ็นต์
ค่า:	ดีฟอลต์: 6; ช่วง: 1 ถึงจำนวนเต็มบวก ใดๆ
การแสดงผล:	**ps -efa | grep biod**
การเปลี่ยนแปลง:	**chnfs -b** *NewValue* โดยปกติแล้ว การเปลี่ยนแปลงมีผลทันทีและเป็นแบบถาวร แฟล็ก **-N** ทำให้การเปลี่ยนแปลง มีผลทันทีและเป็นแบบชั่วคราว แฟล็ก **-I** ทำให้การเปลี่ยนแปลงมีผล เมื่อบูตครั้งถัดไป
การวิเคราะห์:	**netstat -s** เพื่อ ค้นหาบัฟเฟอร์โอเวอร์โฟลว์ของซ็อกเก็ต UDP
การปรับ:	เพิ่มจำนวนจนกว่าบัฟเฟอร์โอเวอร์โฟลว์ของ ซ็อกเก็ตจะยุติลง
อ้างอิงถึง:	“จำนวนของเธรด biod ที่จำเป็น” ในหน้า 382

2. combehind

ไอเท็ม	Descriptor
วัตถุประสงค์:	เปิดใช้งานพฤติกรรม commit-behind บนไคลเอ็นต์ NFS ในขณะที่บันทึกไฟล์ขนาดใหญ่มากบนจุดติดตั้ง NFS เวอร์ชัน 3
ค่า:	ดีฟอลต์: 0; ช่วง: 0 ถึง 1
การแสดงผล:	**mount**
การเปลี่ยนแปลง:	**mount -o combehind**
การวิเคราะห์:	ผลผลิตไม่ดีในขณะที่บันทึกไฟล์ขนาดใหญ่มาก (โดยหลักแล้ว เป็นเพราะไฟล์มีขนาดใหญ่กว่าจำนวนของหน่วยความจำระบบใน ไคลเอ็นต์ NFS) บนจุดติดตั้ง NFS เวอร์ชัน 3
การปรับ:	ใช้อ็อพชันการติดตั้งนี้บนไคลเอ็นต์ NFS ถ้าการใช้งานหลักของ NFS คือการบันทึกไฟล์ขนาดใหญ่มากลงในเซิร์ฟเวอร์ NFS หมายเหตุว่าคุณลักษณะที่ไม่พึงประสงค์ของอ็อพชันนี้คือ การแคช VMM ของข้อมูล ไฟล์ NFS ถูกปิดใช้งานโดยสิ้นเชิงบนไคลเอ็นต์ ดังนั้นจึงไม่แนะนำให้ใช้อ็อพชันนี้ ในสภาพแวดล้อมที่จำเป็นต้องใช้ประสิทธิภาพการอ่าน NFS ที่ดี

3. nfsd Count

ไอเท็ม	Descriptor
วัตถุประสงค์:	ระบุจำนวนสูงสุดของ เธรดเซิร์ฟเวอร์ NFS ที่สร้างขึ้นเพื่อให้บริการการร้องขอ NFS ขาเข้า
ค่า:	ดีฟอลต์: 3891; ช่วง: 1 ถึง 3891
การแสดงผล:	**ps -efa | grep nfsd**
การเปลี่ยนแปลง:	**chnfs -n** *NewValue* การเปลี่ยนแปลงมีผลทันทีและเป็นแบบถาวร แฟล็ก **-N** ทำให้การเปลี่ยนแปลง มีผลทันทีและเป็นแบบชั่วคราว แฟล็ก **-I** ทำให้การเปลี่ยนแปลงมีผล เมื่อบูตครั้งถัดไป
การวิเคราะห์:	ให้ดู **nfs_max_threads**
การปรับ:	ให้ดู **nfs_max_threads**
อ้างอิงถึง:	“จำนวนของเธรด biod ที่จำเป็น” ในหน้า 382

4. numclust

ไอเท็ม	Descriptor
วัตถุประสงค์:	ใช้ร่วมกับอ็อพชัน **combehind** เพื่อพัฒนาประสิทธิภาพผลผลิตการบันทึกในขณะที่บันทึกไฟล์ขนาดใหญ่บนจุดติดตั้ง NFS เวอร์ชัน 3
ค่า:	ดีฟอลต์: 128; ช่วง: 8 ถึง 1024
การแสดงผล:	**mount**
การเปลี่ยนแปลง:	`mount -o numclust=NewValue`
การวิเคราะห์:	ผลผลิตไม่ดีในขณะที่บันทึกไฟล์ขนาดใหญ่มาก (โดยหลักแล้ว เป็นเพราะไฟล์มีขนาดใหญ่กว่าจำนวนของหน่วยความจำระบบใน ไคลเอ็นต์ NFS) บนจุดติดตั้ง NFS เวอร์ชัน 3
การปรับ:	ใช้อ็อพชันการติดตั้งนี้บนไคลเอ็นต์ NFS ถ้าการใช้งานหลักของ NFS คือการบันทึกไฟล์ขนาดใหญ่มากลงในเซิร์ฟเวอร์ NFS โดยพื้นฐาน ค่าแสดงถึงจำนวนหน้าต่ำสุดซึ่ง VMM จะ พยายามสร้างการดำเนินงาน commit จากไคลเอ็นต์ NFS ค่าที่ต่ำเกินไป อาจส่งผลทำให้ผลผลิตไม่ดี เนื่องจากจำนวน commits ที่มากเกินไป (commit แต่ละรายการทำให้เกิดการบันทึกแบบซิงโครนัสบนเซิร์ฟเวอร์) ค่าที่สูงเกินไปอาจส่งผล ทำให้ผลผลิตไม่ดีเช่นเดียวกัน เนื่องจากหน่วยความจำไคลเอ็นต์ NFS เต็มไปด้วย หน้าที่แก้ไขซึ่งเป็นสาเหตุให้เรียกใช้ LRU daemon เพื่อเริ่มต้นการเรียกหน้า คืน เมื่อ **lrud** รัน การบันทึก V3 จะต้องกลายเป็นแบบซิงโครนัส เนื่องจากการบันทึกแต่ละรายการจบลงโดยการ commit สถานการณ์นี้สามารถ หลีกเลี่ยงได้โดยใช้อ็อพชัน **numclust** และ **combehind**

แอ็ตทริบิวต์สตรีมที่ปรับได้

รายการทั้งหมดของแอ็ตทริบิวต์สตรีมที่ปรับได้สามารถจัดหาได้โดยการรันคำสั่ง **no** พร้อมกับอ็อพชัน **–L**

สถานการณ์จำลองกรณีทดสอบ

แต่ละกรณีอธิบายชนิดของระบบและปัญหาที่ พบ มีการอธิบายวิธีการทดสอบปัญหาประสิทธิภาพเฉพาะ และวิธีการแก้ไขปัญหาถ้าตรวจพบ ถ้าคุณมีสถานการณ์จำลองที่คล้ายกัน ในสภาพแวดล้อมของคุณเอง ให้ใช้ข้อมูลในกรณีทดสอบเหล่านี้ ช่วยคุณ

การปรับประสิทธิภาพขึ้นอยู่กับระบบและแอ็พพลิเคชันเป็นหลัก อย่างไรก็ตาม มีวิธีการปรับทั่วไปหลายอย่างที่สามารถใช้ได้บนระบบ AIX ส่วนใหญ่

การปรับปรุงผลการทำงานสำหรับการเขียนไคลเอ็นต์ไฟล์ NFS ที่มีขนาดใหญ่

การเขียนขนาดใหญ่ ไฟล์ลำดับผ่านระบบไฟล์ที่ประกอบเข้ากับ NFS สามารถทำให้ลดความรุนแรงในอัตราการถ่ายโอนไฟล์ไปยังเซิร์ฟเวอร์ NFS ในสถานการณ์นี้ คุณระบุสถานการณ์ที่มีอยู่ และใช้ขั้นตอนต่างๆ เพื่อแก้ไขปัญหา

สิ่งที่ควรนำมาพิจารณา

The information in this how-to scenario was tested using specific versions of AIX. The results you obtain might vary significantly depending on your version and level of AIX.

ระบบกำลังรันแอ็พพลิเคชันที่เขียนไฟล์ที่มีขนาดใหญ่มาก (ใหญ่กว่าจำนวนของหน่วยความจำฟิสิคัลบนเครื่อง) ลงในระบบไฟล์ที่ประกอบเข้ากับ NFS ระบบไฟล์จะถูกประกอบเข้าโดยใช้ NFS V3 เซิร์ฟเวอร์ NFS และไคลเอ็นต์จะสื่อสารผ่านอีเทอร์เน็ตเน็ตเวิร์ก 100 MB ต่อวินาที เมื่อเขียนไฟล์ขนาดเล็กตามลำดับ ค่าเฉลี่ยของทรูพุตจะอยู่ที่ 10 MB ต่อวินาที อย่างไรก็ตาม เมื่อเขียนไฟล์ที่มีขนาดใหญ่มาก ค่าเฉลี่ยของทรูพุตจะตกลงต่ำกว่า 1 MB ต่อวินาที

การเขียนไฟล์ขนาดใหญ่ของแอ็พพลิเคชัน จะเติมหน่วยความจำของไคลเอ็นต์ทั้งหมด ซึ่งทำให้อัตราของการถ่ายโอนไปยังเซิร์ฟเวอร์ NFS จะลดน้อยลง เหตุการณ์นี้เกิดขึ้นเนื่องจากระบบไคลเอ็นต์ AIX ต้องเรียกใช้งาน LRUD **kproc** เพื่อปล่อยเพจบางเพจในหน่วยความจำ เพื่อให้เหมาะสมกับชุดของเพจถัดไปที่จะถูกเขียนลงโดยแอ็พพลิเคชัน

ใช้เมธอดต่อไปนี้ เพื่อตรวจสอบว่า คุณมีประสบการณ์กับปัญหานี้:

- ขณะที่ไฟล์กำลังถูกเขียนลงในเซิร์ฟเวอร์ NFS ให้รันคำสั่ง **nfsstat** เป็นระยะๆ (ทุกๆ 10 วินาที) โดยพิมพ์คำสั่ง:

  ```
  nfsstat
  ```

  ตรวจสอบเอาต์พุตคำสั่ง **nfsstat** ถ้าจำนวนของการ commit V3 มีปริมาณที่เพิ่มขึ้นซึ่งใกล้เคียงกับจำนวนของการเรียกเพื่อเขียน V3 เหตุการณ์นี้จึงดูเหมือนจะเป็นปัญหา
- ใช้คำสั่ง **topas** (อยู่ในชุดไฟล์ `bos.perf.tools`) เพื่อมอนิเตอร์จำนวนของข้อมูลต่อวินาทีที่ต้องส่งไปยังเซิร์ฟเวอร์ NFS โดยพิมพ์คำสั่งต่อไปนี้:

  ```
  topas -i 1
  ```

ถ้าเมธอดที่แสดงบ่งชี้ว่ามีปัญหาเกิดขึ้น โซลูชันคือ ใช้อ็อพชันคำสั่ง **mount** ที่เรียก **combehind** เมื่อประกอบเข้ากับระบบไฟล์เซิร์ฟเวอร์ NFS บนระบบไคลเอ็นต์ ปฏิบัติดังต่อไปนี้:

1. เมื่อระบบไฟล์ไม่แอ็คทีฟให้ถอดออกโดยพิมพ์คำสั่งต่อไปนี้ :

   ```
   unmount /mnt
   ```

   (สมมุติว่า /mnt คือจุดประกอบ)
2. ให้ประกอบระบบรีโมตไฟล์โดยใช้คำสั่ง **mount** ที่เรียกว่า **comebehind** ดังนี้:

   ```
   mount -o combehind server_hostname:/remote_mount_point /mnt
   ```

หลักการที่เกี่ยวข้อง:

“ประสิทธิภาพการทำงาน NFS” ในหน้า 362
AIX มี เครื่องมือและเมธอดสำหรับการมอนิเตอร์ Network File System (NFS) และการปรับแต่งทั้ง เซิร์ฟเวอร์ และไคลเอ็นต์

ข้อมูลที่เกี่ยวข้อง:

คำสั่ง mount

คำสั่ง nfsstat

คำสั่ง topas

รูทีนการรักษาความปลอดภัยด้วยการทำดัชนีรหัสผ่าน

ในสถานการณ์นี้ คุณจะตรวจสอบว่า คุณมีจำนวนของการประมวลผลรูทีนที่รักษาความปลอดภัยสูง จากนั้นลดจำนวนของเวลาที่ตัวประมวลผลใช้สำหรับรูทีนการรักษาความปลอดภัย โดยการทำดัชนีไฟล์รหัสผ่าน

สิ่งที่ควรนำมาพิจารณา

The information in this how-to scenario was tested using specific versions of AIX. The results you obtain might vary significantly depending on your version and level of AIX.

สภาวะแวดล้อมของสถานการณ์จำลองประกอบด้วยระบบแบบสองทิศทางที่ใช้เป็น เมลเซิร์ฟเวอร์ เมลที่ได้รับผ่าน POP3 (Post Office Protocol Version 3) แบบรีโมตและโลคัลไคลเอ็นต์ของเมลที่มีล็อกอินบนเซิร์ฟเวอร์เดียวกัน เมลที่ส่งโดยใช้ **sendmail** daemon เนื่องจากโดยการทำงานของเมลเซิร์ฟเวอร์ จำนวนของรูทีนความปลอดภัยสูงจะถูกเรียกสำหรับการพิสูจน์ตัวตนของผู้ใช้ หลังจากที่ย้ายจากเครื่องยูนิโพรเซสเซอร์ไปยังระบบแบบ 2 ทิศทาง คำสั่ง **uptime** จะส่งคืนการประมวลผล 200 กระบวนการ ซึ่งเปรียบเทียบน้อยกว่า 1 บนเครื่องยูนิโพรเซสเซอร์

หากต้องการพิจารณาสาเหตุของการลดระดับ ผลการทำงานและลดจำนวนของเวลาของตัวประมวลผลที่ใช้สำหรับรูทีนความปลอดภัย ให้ปฏิบัติดังนี้:

1. พิจารณาการประมวลผลที่ใช้เปอร์เซ็นต์ของเวลาตัวประมวลผลสูง และเวลาของตัวประมวลผลจะถูกใช้ในเคอร์เนล หรือโหมดผู้ใช้โดยรันคำสั่งต่อไปนี้ (อยู่ในชุดไฟล์ `bos.perf.tools`):

   ```
   topas -i 1
   ```

   เอาต์พุตคำสั่ง **topas** ในสถานการณ์จำลองของเราซึ่งบ่งชี้ว่า เวลาของตัวประมวลผล ประมาณ 90% จะถูกใช้ในโหมดผู้ใช้ และการประมวลผลที่ใช้เวลาของตัวประมวลผลส่วนใหญ่คือ **sendmail** and **pop3d** (มีการใช้ตัวประมวลผลเวลาเคอร์เนล การติดตามเคอร์เนลจะเป็นเครื่องมือที่เหมาะสม ในการดำเนินการต่อ)
2. พิจารณาว่า เวลาของตัวประมวลผลในโหมดผู้ใช้จะถูกใช้ในแอ็พพลิเคชันโค้ด (ผู้ใช้) หรือในไลบรารีที่แบ่งใช้ (แบ่งใช้) โดยรันคำสั่งต่อไปนี้ เพื่อรวบรวมข้อมูลสำหรับเวลา 60 วินาที:

   ```
   tprof -ske -x "sleep 60"
   ```

   คำสั่ง **tprof** จะแสดงชื่อของรูทีนย่อยที่เรียกใช้นอกไลบรารีที่แบ่งใช้ และเรียงลำดับตามจำนวนของขีดของตัวประมวลผลที่ใช้สำหรับรูทีนย่อยแต่ละตัว ข้อมูล **tprof** ที่แสดงเวลาของตัวประมวลผลในโหมดผู้ใช้จะใช้ในไลบรารีระบบ `libc.a` สำหรับรูทีนความปลอดภัย (และรูทีนย่อยเหล่านั้นที่เรียกโดยระบบ) (มีคำสั่ง **tprof** ที่แสดงเวลาของตัวประมวลผลในโหมดผู้ใช้ในโค้ดแอ็พพลิเคชัน (ผู้ใช้) จากนั้นการดีบักแอ็พพลิเคชันและการทำโปรไฟล์ควรเป็นสิ่งที่จำเป็น)
3. หากต้องการหลีกเลี่ยงไฟล์ `/etc/passwd` ที่สแกนด้วยรูทีนความปลอดภัยแต่ละตัว ให้สร้างดัชนีสำหรับไฟล์โดยรันคำสั่งต่อไปนี้ :

   ```
   mkpasswd -f
   ```

ด้วยการใช้ไฟล์รหัสผ่านที่ทำดัชนีแล้ว ค่าเฉลี่ยในการโหลดสำหรับสถานการณ์จำลองนี้ จะลดลงจากค่า 200 ไปเป็น 0.6

สำหรับข้อมูลเพิ่มเติม

- คำอธิบายคำสั่ง **topas**, **tprof** และ **uptime** ใน *Commands Reference, Volume 5*
- คำอธิบาย **pop3d** และ **sendmail** daemon ใน *Commands Reference, Volume 4*

หน่วยความจำแบบแบ่งใช้ BSR

barrier synchronization register (BSR) คือสิ่งอำนวยความสะดวก ฮาร์ดแวร์ที่แบ่งใช้พื้นที่หน่วยความจำขนาดเล็กหรือหนาแน่นได้อย่างมีประสิทธิภาพ พื้นที่ หน่วยความจำมีการอัพเดตในแบบขนานโดยหลายเธรด

หน่วยความจำ BSR อนุญาตให้เก็บสามารถขยายโดยใช้ระบบโดยมีอัตราความเร็วมากกว่าหน่วยความจำแคชแบบปกติ หน่วยความจำ BSR ใช้ความหมายการจัดการแคชที่ได้รับจัดสรรสำหรับแอ็พพลิเคชันแบบขนาน สูงซึ่งใช้หน่วยความจำขนาดเล็กจากหลายตัวประมวลผล ความหมายการจัดการแคชไม่ได้มีไว้ใช้สำหรับหน่วยความจำแบบแบ่งใช้สำหรับ วัตถุประสงค์ทั่วไป สิ่งอำนวยความสะดวกนี้มีประโยชน์สำหรับการนำโครงสร้าง การซิงโครไนซ์ barrier ซึ่งใช้ในเวิร์กโหลดแบบขนานสูงไปใช้อย่างมีประสิทธิภาพ

หน่วยความจำ BSR ต้องการการสนับสนุนตัวประมวลผลเฉพาะ และต้องกำหนด คอนฟิกรีซอร์สเป็นพาร์ติชันโลจิคัล (LPAR) เพื่อให้ LPAR สามารถใช้ สิ่งอำนวยความสะดวก BSR ได้

เมื่อต้องการจัดสรรหน่วยความจำแบบแบ่งใช้ BSR ให้ปฏิบัติตามขั้นตอนเหล่านี้:

1. จัดสรรพื้นที่หน่วยความจำแบบแบ่งใช้ของ system V สำหรับหน่วยความจำแบบแบ่งใช้ของ BSR โดยใช้รูทีนย่อย `shmctl()`
2. ร้องขอให้สำรองพื้นที่หน่วยความจำแบบแบ่งใช้ของ system V ที่ถูกจัดสรรโดยใช้หน่วยความจำ BSR โดยใช้รูทีนย่อย `shmctl()` และระบุคำสั่ง **SHM_BSR**

   หมายเหตุ: รูทีนย่อย `shmctl()` ที่ใช้กับคำสั่ง **SHM_BSR** จะถูกใช้บนหน่วยความจำแบบแบ่งใช้ ขั้นตอนนี้จะถูกดำเนินการทันทีหลังจากหน่วยความจำแบบแบ่งใช้ของ system V ถูกสร้างขึ้นโดยใช้รูทีนย่อย `shmget()` และก่อนกระบวนการใดๆ ที่เชื่อมกับหน่วยความจำแบบแบ่งใช้ เมื่อใช้คำสั่ง **SHM_BSR** รูทีนย่อย `shmctl()` จะพยายามใช้หน่วยความจำ BSR สำหรับพื้นที่หน่วยความจำแบบแบ่งใช้ที่ระบุ
3. แสดงข้อผิดพลาด ถ้าหน่วยความจำ BSR ที่มีอยู่ไม่เพียงพอ หรือ สิ่งอำนวยความสะดวก BSR ไม่ได้รับการสนับสนุนจากฮาร์ดแวร์แพล็ตฟอร์ม รูทีนย่อย `shmget()` ล้มเหลวด้วย `errno` ที่ตั้งค่าเป็น `ENOMEM`

   หมายเหตุ: ผู้ใช้ ที่ไม่ใช่ผู้ใช้รากต้องมีความสามารถ `CAP_BYPASS_RAC_VMM` เพื่อจัดสรรหน่วยความจำ BSR หากผู้ใช้ที่ไม่ใช่ผู้ใช้รากไม่มี ความสามารถนี้ รูทีนย่อย `shmctl()` ที่มีคำสั่ง **SHM_BSR** จะล้มเหลวด้วย `errno` ที่ตั้งค่าเป็น `EPERM`

เมื่อใช้หน่วยความจำแบบแบ่งใช้ของ BSR เฉพาะทำสั่งที่เก็บแบบ 1 ไบต์ และ 2 ไบต์ จะได้รับอนุญาตให้แบ่งใช้พื้นที่หน่วยความจำ คำสั่งที่เก็บที่ใช้มากกว่า 2 ไบต์จะทำงานไม่ถูกต้องกับหน่วยความจำแบบแบ่งใช้ของ BSR คำสั่งโหลดของทุกขนาดสามารถใช้ได้กับหน่วยความจำแบบแบ่งใช้ BSR

คำสั่ง **VMINFO** มีการรันบนรูทีนย่อย `vmgetinfo()` ซึ่งใช้เพื่อรวบรวมข้อมูลเกี่ยวกับการสนับสนุน BSR ที่มีอยู่ เมื่อมีการระบุคำสั่ง **VMINFO** ในรูทีนย่อย `vmgetinfo()` จะมีการส่งคืนโครงสร้าง `vminfo` ฟิลด์ `bsr_mem_total` รายงานจำนวนทั้งหมดของหน่วยความจำ BSR ที่กำหนดคอนฟิกใน LPAR ฟิลด์ `bsr_mem_free` รายงานจำนวนทั้งหมดของ BSR ที่มีอยู่ในปัจจุบันสำหรับการจัดสรร

พื้นที่หน่วยความจำแบบแบ่งใช้ BSR ไม่สามารถมีการปรับขนาดแบบไดนามิก ด้วยอ็อพชัน `SHM_SIZE` ในรูทีนย่อย `shmctl()` หากแอ็พพลิเคชันพยายามปรับขนาดพื้นที่หน่วยความจำแบบแบ่งใช้ BSR โดย การระบุพารามิเตอร์ `SHM_SIZE` ในรูทีนย่อย `shmctl()` `shmctl()` จะล้มเหลว ด้วย `errno` ที่ตั้งค่าเป็น `EINVAL` หน่วยความจำแบบแบ่งใช้ BSR ไม่ได้รับการสนับสนุนจากตัวแปรสภาวะแวดล้อม `EXTSHM` หากมีการตั้งค่าตัวแปรสภาวะแวดล้อม `EXTSHM` เมื่อเรียก `shmctl()` ด้วยแฟล็ก `SHM_BSR` `shmctl()` จะล้มเหลว ด้วย `EINVAL`

ตัวอย่าง

ตัวอย่างต่อไปนี้แสดงว่า แอ็พพลิเคชันสามารถเคียวรีจำนวนของหน่วยความจำ BSR ที่มีอยู่บนระบบ จากนั้นจัดสรรและแนบพื้นที่หน่วยความจำแบบแบ่งใช้ BSR ตัวอย่าง แสดงวิธีการที่แอ็พพลิเคชันแยกและลบพื้นที่หน่วยความจำแบบแบ่งใช้ BSR

```
#include <errno.h>
#include <stdio.h>
#include <sys/shm.h>
#include <sys/vminfo.h>

/* shm_rgn_size is the size of the shared memory region to
 * allocate. In this example, 4KB (PAGESIZE) is chosen.  4KB is the
 * smallest shared memory region size supported.  It is expected that
 * 4KB should be sufficient for most users of BSR memory.
 */
const size_t shm_rgn_size = PAGESIZE;

int main(int argc, char *argv[])
{
  struct vminfo my_info = { 0 };
  int           id;
  void          *ptr;

  /* Determine the amount of BSR memory available */
  if (vmgetinfo(&my_info, VMINFO, sizeof(my_info)) != 0)
 {
    perror("vmgetinfo() unexpectedly failed");
    return 1;
 }

  /* Check to see that sufficient BSR memory is available */
  if (my_info.bsr_mem_free < shm_rgn_size)
 {
    fprintf(stderr, "insufficient BSR memory\n");
    return 2;
 }

  /* Allocate a new shared memory region */
  id = shmget(IPC_PRIVATE, shm_rgn_size, IPC_CREAT|IPC_EXCL);

  if (id == -1)
 {
    perror("shmget() failed");
    return 3;
 }

  /* Request BSR memory for the shared memory region */
  if (shmctl(id, SHM_BSR, NULL))
 {
    perror("shmctl(SHM_BSR) failed");
    shmctl(id, IPC_RMID, 0);
    return 4;
 }

  /* Attach the shared memory region */
  ptr = shmat(id, NULL, 0);
  if ((int)ptr == -1)
 {
    perror("shmat() failed");
    shmctl(id, IPC_RMID, 0);
```

```
    return 5;
 }

  /* BSR memory can now be accessed starting at address - ptr */

  /* Detach shared memory region */
  if (shmdt(ptr))
 {
    perror("shmdt() failed");
    shmctl(id, IPC_RMID, 0);
    return 6;
 }

  /* Delete shared memory region */
  if (shmctl(id, IPC_RMID, 0))
 {
    perror("shmctl(IPC_RMID) failed");
    return 7;
 }

 return  0;
}
```

นโยบาย VMM fork

คุณสามารถเปลี่ยนวิธีการจัดการกับหน่วยความจำเฉพาะกิจสำหรับกระบวนการ เมื่อกระบวนการถูก fork

Virtual memory manager (VMM) ไม่ได้คัดลอก พื้นที่แอดเดรสทั้งหมดของกระบวนการเมื่อกระบวนการถูก fork หน้ามีการคัดลอกตามต้องการเมื่อมีการปรับเปลี่ยนโดยกระบวนการพาเรนต์หรือ ชายน์ ข้อมูลอ้างอิงโหลดของหน้าที่ยังไม่ได้ปรับเปลี่ยน มีการแก้ไขเป็นหน่วยความจำซึ่งแบ่งใช้ระหว่างกระบวนการพาเรนต์และชายน์ ถ้าหน้ามีการปรับเปลี่ยนในเวลาต่อมา หน้าจะถูกคัดลอกในเวลาที่ ปรับเปลี่ยน

ถ้ากระบวนการอ่านและเขียนหน่วยความจำในทันที เป็นเรื่องง่ายกว่าที่จะทำสำเนาของหน้าเมื่อมีการอ้างอิง ครั้งแรกหาก เปรียบเทียบกับเมื่อเขียนหน้าครั้งแรก ลักษณะการทำงานนี้สามารถ ใช้กับทั้งระบบได้โดยปรับเปลี่ยนพารามิเตอร์ที่ปรับได้ **vmm_fork_policy** แบบมีข้อจำกัด โดยใช้คำสั่ง **vmo** คุณสามารถลบล้างพารามิเตอร์ที่ปรับได้แบบโกลบอลด้วยกระบวนการเดียว โดยเอ็กซ์พอร์ตตัวแปรสภาวะแวดล้อม `VMM_CNTRL` และโดยการระบุคีย์เวิร์ด **vmm_fork_policy**

คำประกาศ

ข้อมูลนี้พัฒนาขึ้นสำหรับผลิตภัณฑ์และบริการที่มีในประเทศสหรัฐอเมริกาเท่านั้น

IBM อาจไม่นำเสนอผลิตภัณฑ์ เซอร์วิส หรือคุณลักษณะที่อธิบายในเอกสารนี้ในประเทศอื่น โปรดปรึกษาตัวแทน IBM ในท้องถิ่นของคุณสำหรับข้อมูลเกี่ยวกับผลิตภัณฑ์ และเซอร์วิส ที่มีอยู่ในพื้นที่ของคุณในปัจจุบัน การอ้างอิงใดๆ ถึงผลิตภัณฑ์ โปรแกรม หรือเซอร์วิสของ IBM ไม่ได้มีวัตถุประสงค์ที่จะระบุหรือตีความว่า สามารถใช้ได้เฉพาะผลิตภัณฑ์ โปรแกรม หรือเซอร์วิสของ IBM เพียงอย่างเดียว เท่านั้น ผลิตภัณฑ์ โปรแกรม หรือเซอร์วิสใดๆ ที่สามารถทำงานได้เท่าเทียมกัน และไม่ละเมิดสิทธิทรัพย์สินทางปัญญาของ IBM อาจนำมาใช้ แทนได้ อย่างไรก็ตาม ถือเป็นความรับผิดชอบของผู้ใช้ที่จะประเมินและตรวจสอบการดำเนินการของ ผลิตภัณฑ์ โปรแกรม หรือเซอร์วิสใดๆ ที่ไม่ใช่ของ IBM

IBM อาจมีสิทธิบัตร หรืออยู่ระหว่างดำเนินการขอ สิทธิบัตรที่ครอบคลุมถึงหัวข้อซึ่งอธิบายในเอกสารนี้ การนำเสนอเอกสารนี้ไม่ได้เป็นการให้ไลเซนส์ใดๆ ในสิทธิบัตรเหล่านี้แก่คุณ คุณสามารถส่งการสอบถามเกี่ยวกับไลเซนส์ เป็นลายลักษณ์อักษรไปยัง:

IBM Director of Licensing
IBM Corporation
North Castle Drive, MD-NC119
Armonk, NY 10504-1785
US

หากมีคำถามเกี่ยวกับข้อมูลชุดอักขระไบต์คู่ (DBCS) โปรดติดต่อแผนกทรัพย์สินทางปัญญาของ IBM ในประเทศของคุณ หรือส่งคำถาม เป็นลายลักษณ์อักษร ไปยัง:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan Ltd.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, Japan

INTERNATIONAL BUSINESS MACHINES CORPORATION จัดเตรียมเอกสาร "ตามสภาพที่เป็น" โดยไม่มีการรับประกันใดๆ ทั้งโดยชัดแจ้งหรือโดยนัย ซึ่งรวมถึง แต่ไม่จำกัดถึงการรับประกันโดยนัยที่ไม่ละเมิดความสามารถในการจัดจำหน่าย หรือตามความเหมาะสมสำหรับวัตถุประสงค์อย่างใดอย่างหนึ่ง ในบางรัฐไม่อนุญาตให้ ปฏิเสธการรับประกันทางตรงหรือทางอ้อมในธุรกรรมบางอย่าง ดังนั้น ข้อความนี้จึงอาจจะไม่ใช้กับคุณ

ข้อมูลนี้อาจมีความไม่ถูกต้องด้านเทคนิคหรือข้อผิดพลาดจากการพิมพ์ มีการเปลี่ยนแปลง ข้อมูลในเอกสารนี้เป็นระยะ และการเปลี่ยนแปลงเหล่านี้จะรวมอยู่ในเอดิชันใหม่ของ สิ่งพิมพ์ IBM อาจปรับปรุง และ/หรือเปลี่ยนแปลงในผลิตภัณฑ์ และ/หรือโปรแกรมที่อธิบายในสิ่งพิมพ์นี้ได้ตลอดเวลา โดยไม่ต้องแจ้งให้ทราบ

การอ้างอิงใดๆ ในข้อมูลนี้ถึงเว็บไซต์ไม่ใช่ของ IBM มีการจัดเตรียมเพื่อความสะดวกเท่านั้น และ ไม่ได้เป็นการรับรองเว็บไซต์เหล่านั้นในลักษณะใดๆ เอกสารประกอบที่เว็บไซต์เหล่านั้นไม่ได้เป็น ส่วนหนึ่งของเอกสารประกอบสำหรับผลิตภัณฑ์ IBM นี้ และการใช้เว็บไซต์เหล่านั้นถือเป็นความเสี่ยงของคุณเอง

IBM อาจใช้หรือแจกจ่ายข้อมูลใดๆ ที่คุณ ให้ในรูปแบบต่างๆ ซึ่ง IBM เชื่อว่ามีความเหมาะสมได้โดยไม่เกิดข้อผูกมัดใดๆ กับคุณ

ผู้รับไลเซนส์ของโปรแกรมนี้ที่ต้องการข้อมูลเกี่ยวกับโปรแกรมสำหรับวัตถุประสงค์ในการเปิดใช้งาน: (i) การแลกเปลี่ยนข้อมูลระหว่างโปรแกรมที่สร้างขึ้นอย่างอิสระกับโปรแกรมอื่น (รวมถึง โปรแกรมนี้) และ (ii) การใช้ข้อมูลซึ่งแลกเปลี่ยนร่วมกัน ควร ติดต่อ:

IBM Director of Licensing
IBM Corporation
North Castle Drive, MD-NC119
Armonk, NY 10504-1785
US

ข้อมูลดังกล่าวอาจพร้อมใช้งาน ภายใต้ข้อตกลงและเงื่อนไขที่เหมาะสม รวมถึง การชำระค่าธรรมเนียมในบางกรณี

โปรแกรมที่มีไลเซนส์ซึ่งอธิบายในเอกสารนี้ และเอกสารประกอบที่มีไลเซนส์ทั้งหมดสำหรับโปรแกรม นั้น มีการจัดเตรียมโดย IBM ภายใต้ข้อตกลงของข้อตกลงกับลูกค้าของ IBM, ข้อตกลงไลเซนส์โปรแกรมระหว่างประเทศของ IBM หรือข้อตกลงที่เท่าเทียมกันใดๆ ระหว่างเรา

ข้อมูลประสิทธิภาพ และตัวอย่างลูกค้าที่ระบุมีการนำเสนอสำหรับวัตถุประสงค์การสาธิตเท่านั้น ผลลัพธ์ของประสิทธิภาพการทำงานจริงอาจขึ้นอยู่กับคอนฟิกูเรชันและเกณฑ์การทำงานที่ระบุเฉพาะ

ข้อมูลเกี่ยวกับผลิตภัณฑ์ที่ไม่ใช่ของ IBM ได้รับมาจากซัพพลายเออร์ของผลิตภัณฑ์เหล่านั้น ประกาศที่เผยแพร่ หรือแหล่งข้อมูลที่เปิดเผยต่อสาธารณะ IBM ไม่ได้ทดสอบผลิตภัณฑ์ดังกล่าว และไม่สามารถยืนยันความถูกต้องของ ประสิทธิภาพ ความเข้ากันได้ หรือการเรียกร้องอื่นใดที่เกี่ยวข้องกับผลิตภัณฑ์ที่ไม่ใช่ของ IBM คำถามเกี่ยวกับ ความสามารถของผลิตภัณฑ์ที่ไม่ใช่ของ IBM ควรส่งไปยังซัพพลายเออร์ของผลิตภัณฑ์เหล่านั้น

ข้อความใดๆ ที่เกี่ยวข้องกับทิศทางในอนาคตและเจตจำนงค์ของ IBM อาจมีการเปลี่ยนแปลง หรือเพิกถอนได้โดยไม่ต้องแจ้งล่วงหน้า และนำเสนอเฉพาะเป้าหมาย และวัตถุประสงค์เท่านั้น

ราคาของ IBM ทั้งหมดที่แสดงเป็นราคาขายปลีกที่แนะนำของ IBM ซึ่งเป็นราคาปัจจุบัน และอาจเปลี่ยนแปลงได้โดยไม่ต้องแจ้งให้ทราบ ราคาของผู้แทนจำหน่ายอาจแตกต่างไป

ข้อมูลนี้ใช้สำหรับวัตถุประสงค์ของการวางแผนเท่านั้น ข้อมูลในเอกสารนี้อาจมีการเปลี่ยนแปลง ก่อนผลิตภัณฑ์ที่อธิบายจะวางจำหน่าย

ข้อมูลนี้มีตัวอย่างของข้อมูลและรายงานที่ใช้ในการดำเนินการทางธุรกิจรายวัน เพื่อ สาธิตข้อมูลให้สมบูรณ์ที่สุดเท่าที่จะเป็นไปได้ ตัวอย่างจึงมีชื่อของแต่ละบุคคล บริษัท ยี่ห้อ และผลิตภัณฑ์ ชื่อเหล่านี้ทั้งหมดเป็นชื่อสมมติ และความคล้ายคลึงใดๆ กับบุคคล หรือองค์กรธุรกิจที่มีอยู่จริง ถือเป็นเหตุบังเอิญ

ไลเซนส์ลิขสิทธิ์:

ข้อมูลนี้มีตัวอย่างแอ็พพลิเคชันโปรแกรมในภาษาต้นฉบับ ซึ่งแสดงถึง เทคนิคด้านโปรแกรมในหลากหลายแพล็ตฟอร์ม คุณอาจคัดลอก ปรับเปลี่ยน และแจกจ่าย โปรแกรมตัวอย่างเหล่านี้ในรูปแบบใดๆ โดยไม่ต้องชำระเงินให้แก่ IBM สำหรับวัตถุประสงค์ในการพัฒนา การใช้ การตลาด หรือการแจกจ่ายโปรแกรมแอ็พพลิเคชัน ที่สอดคล้องกับอินเตอร์เฟสการเขียน

โปรแกรมแอ็พพลิเคชันสำหรับแพล็ตฟอร์มปฏิบัติการ ซึ่งเขียน โปรแกรมตัวอย่าง ตัวอย่างเหล่านี้ยังไม่ได้ผ่านการทดสอบในทุกสภาพ ดังนั้น IBM จึงไม่สามารถรับประกัน หรือบอกเป็นนัยถึง ความน่าเชื่อถือ ความสามารถบริการได้ หรือฟังก์ชันของโปรแกรมเหล่านี้ โปรแกรมตัวอย่างมีการนำเสนอ "ตาม สภาพ" โดยไม่มีการรับประกันประเภทใดๆ IBM ไม่รับผิดชอบ ต่อความเสียหายใดๆ ที่เกิดขึ้นเนื่องจากการใช้โปรแกรมตัวอย่างของคุณ

แต่ละสำเนาหรือส่วนใดๆ ของโปรแกรมตัวอย่างเหล่านี้ หรืองานที่สืบเนื่องใดๆ ต้องมีคำประกาศ ลิขสิทธิ์ดังนี้:

© (ชื่อบริษัทของคุณ) (ปี)

ส่วนของโค้ดนี้ได้มาจากโปรแกรมตัวอย่างของ IBM Corp.

© Copyright IBM Corp. (C) ลิขสิทธิ์ IBM Corp. _ป้อน ปี_

ข้อความพิจารณาเกี่ยวกับนโยบายส่วนตัว

IBM Software products, including software as a service solutions, (“Software Offerings”) may use cookies or other technologies to collect product usage information, to help improve the end user experience, to tailor interactions with the end user or for other purposes. In many cases no personally identifiable information is collected by the Software Offerings. Some of our Software Offerings can help enable you to collect personally identifiable information. If this Software Offering uses cookies to collect personally identifiable information, specific information about this offering’s use of cookies is set forth below.

This Software Offering does not use cookies or other technologies to collect personally identifiable information.

If the configurations deployed for this Software Offering provide you as the customer the ability to collect personally identifiable information from end users via cookies and other technologies, you should seek your own legal advice about any laws applicable to such data collection, including any requirements for notice and consent.

For more information about the use of various technologies, including cookies, for these purposes, see IBM’s Privacy Policy at http://www.ibm.com/privacy and IBM’s Online Privacy Statement at http://www.ibm.com/privacy/details the section entitled “Cookies, Web Beacons and Other Technologies” and the “IBM Software Products and Software-as-a-Service Privacy Statement” at http://www.ibm.com/software/info/product-privacy.

เครื่องหมายการค้า

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the web at Copyright and trademark information at www.ibm.com/legal/copytrade.shtml.

INFINIBAND, InfiniBand Trade Association, and the INFINIBAND design marks are trademarks and/or service marks of the INFINIBAND Trade Association.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

ดัชนี

A

B

C

D

E

F

G

I

J

L

M

N

P

R

S

T

ข

ค

จ

ช

ซ

ด

ต

ท

ธ

น

บ

ป

ผ

พ

ฟ

ภ

ม

ร

ล

ว

ส

ห

อ

ฮ

พิมพ์ในสหรัฐอเมริกา