Linux on POWER versus Solaris 10, Part 1: A technical comparison

Chakarat Skawratananond
IBM eServer Solutions Enablement
June 2005

Table of contents

Abstract

This article is the first in a series of articles that compares the new features offered in Solaris 10 to the technologies available in Linux™ running on IBM® POWER™ processor-based servers.

Introduction

The goal of this series of articles is to provide a technical comparative analysis between the new features introduced in Solaris 10 and those available in Linux running on POWER processor-based systems (Linux on POWER).

With the advanced technology of the POWER architecture, the support of the Linux Open Source Community, and IBM's Linux Technology Center, Linux on POWER proven to be one of the most powerful computing platforms in the market. For more information about Linux on POWER, please visit Linux on POWER Resource Center at ibm.com/servers/enable/linux/power/tech.html.

Solaris 10 is the latest version of Sun's UNIX®-based operating system. Solaris 10 introduces many new features that, according to Sun, make it the most efficient, secure, and reliable operating system ever built[1]. Solaris 10 is running on both Sun's SPARC-based systems and the industry-standard x86 and x86-64 platforms. Sun also plans to release Solaris as open source software[2] under its own license structure. With all that, Solaris 10 poses a seemingly formidable challenge to the adoption of Linux in both commercial and technical computing environments.

In this first article, I'll compare the following features found in Solaris 10 with the technically equivalent tools and technologies available in Linux on POWER:

- DTrace
- Memory Placement Optimization
- Multithreading Enhancements

The next article in this series will cover these features: Containers, Predictive Self-Healing, Cryptographic Framework, and Process rights management.

For Linux on POWER, I base my discussion, as much as possible, on the tools available in the two latest distributions supported on the platform: SUSE LINUX Enterprise Server 9 (SLES9) and Red Hat Enterprise Linux 4 (RHEL4).

DTrace

DTrace[3] is a facility built into Solaris 10 that administrators and developers can use to trace and troubleshoot systemic problems in real time to quickly eliminate bottlenecks in live production systems. DTrace tracks down performance problems across many layers of software, or locates the cause of aberrant behavior by providing hooks into the kernel

[1] http://www.sun.com/software/solaris/features.jsp
[2] http://www.opensolaris.org/
[3] http://www.sun.com/bigadmin/content/dtrace/

through system calls where applications can be called when these system calls are encountered.

There are a number of powerful technologies available for Linux on POWER that provide some, if not all, of the features provided by DTrace. In the following, we provide a brief introduction to each of those tools.

SystemTap and Kprobes

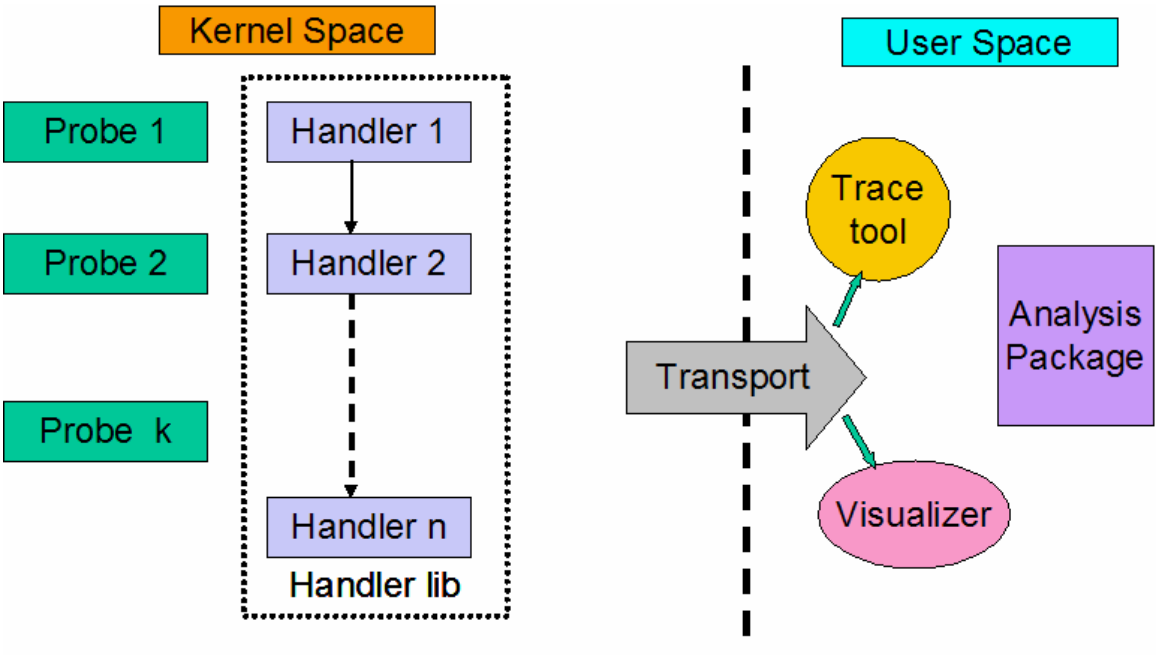
SystemTap (sources.redhat.com/systemtap) is a dynamic instrumentation system for Linux. The current members of this open source project include Red Hat, IBM, and Intel™. SystemTap is built on top of Kprobes. Kprobes is a facility that provides insight into the operation of the Linux kernel without recompiling or rebooting because it is built as a kernel module. Kprobes allows locations in the kernel to be instrumented with code which will be executed when the processor encounters that probe point. After the instrumentation code completes execution, the kernel resumes the normal operation. Kprobes is now included in the mainline 2.6 Linux kernel. The initial release is also targeted to support PPC64.

Although SystemTap is still a work-in-progress, its current design features are comparable to those offered in DTrace. SystemTap will be safe and lightweight enough to use with live production systems. It will be able to instrument both kernel and user space programs even in the absence of source code. SystemTap's probe language will be easy to use and users will be able to reuse general scripts written by others.

One of the differences between SystemTap and Dtrace is that Dtrace uses an in-kernel interpreter whereas SystemTap uses compiled native code. Compiled native code is faster than interpreted code. Therefore, using SystemTap will not affect the performance of the system while performing performance measurements. The in-kernel interpreter has to be completely bug free, otherwise problems in the interpreter itself can cause the system to crash. Moreover, the interpreter is newly developed and not as mature as the compiler, hence there is a higher possibility of encountering bugs. There are a few kernel debugging features that will be supported by SystemTap but not by Dtrace. These features include the ability to write arbitrary locations in kernel memory and the ability to invoke arbitrary kernel subroutines.

Figure 1 illustrates the relationship between the components of SystemTap.

Figure 1. SystemTap block diagram



Links to more information about Kprobes and SystemTap are provided in the Resources section.

OProfile

OProfile (oprofile.sf.net) is a systemic, low-overhead profiler capable of profiling all running code including hardware and software interrupt handlers, kernel modules, kernel, shared libraries, and user-space applications.

One of the most important features provided in OProfile is the capability of generating function-level or instruction-level reports. Users can request reports that have the source trees annotated with the profile information. The generated profiling report is essential in helping identify performance bottlenecks in the system.

OProfile for Linux on POWER uses a kernel module that has access to performance counter registers and a user-space daemon that runs in the background collecting the data from these registers. OProfile has kernel support for POWER4™, POWER5™, and PowerPC® 970 processors. OProfile is included in both RHEL4 and SLES9. Refer to the article, *Identify performance bottlenecks with OProfile for Linux on POWER* (see Resources) for detailed information about how to use OProfile specifically for Linux on POWER.

Performance Inspector

Performance Inspector (sourceforge.net/projects/perfinsp) is a suite of tools that allows you to identify performance problems in your applications and shows you how your application interacts with the Linux kernel. It consists of the following tools:

- **ITrace PPC64** is a software tracing mechanism that allows you to trace through both application and kernel code. ITrace is most useful in situations where you have located a specific performance hotspot in your code and would like to

optimize that hotspot at the assembly instruction level.

- **TProf** is a CPU profiling tool. TProf interrupts the system periodically by time or the hardware performance monitor counters, and then determines the address of the interrupted code along with its process id and thread id. The sampling information is then processed and used to report hotspots in your code.

- **PTT** collects per-thread statistics such as number of CPU cycles, number of interrupts and number of times that the thread was dispatched.

- **JLM** provides statistics on locks based in the Java™ 2 technology.

- **JProf** is a shared library that interfaces with Java jvmpi interface.

- **Hdump** is used to analyze the live objects in a JAVA heap. Hdump provides a live object heap usage summary by object class.

The majority of code within Performance Inspector is released under the GNU General Public license (GPL). Some shared libraries are under the GNU LGPL. At the time of this writing, there are three different packages for the different set of tools and methods for installation. They are:

1. **PerfInsp.ITrace.PPC64:** This package includes the version of ITrace for PPC64 that utilizes Kprobe kernel feature. The package contains the kernel patch for Kprobe, and therefore requires the kernel rebuild. This package has been tested on SLES9 and RHEL4.

2. **PerfInsp:** This package uses kernel patches and therefore requires the kernel rebuild. It is for the 2.4 kernel-based distributions.

3. **Dpiperf.beta:** This package is primarily intended for the 2.6 kernel-based distributions. In this package, TProf and java tools can be used without rebuilding the kernel. Other tools such as PTT and AI are dependent upon Kprobes. They therefore require a kernel patch until Kprobe is included in the distributions. At the time of this writing, this package is still in beta release and has not been fully tested on SLES9 and RHEL4 for PPC64.

IBM Performance Simulator for Linux on POWER

IBM Performance Simulator for Linux on POWER (alphaworks.ibm.com/tech/simppc) is a suite of performance models based on IBM POWER processors. The simulator enables you to examine how your code is executed on various IBM POWER processors so that you can identify and avoid performance bottlenecks on these processors. The supported processors include POWER4, POWER4+, POWER5, and PowerPC 970.

The Performance Simulator can also be used to analyze the *qtrace* output from ITrace. You can view a depiction of how the POWER processor pipeline is handling your instructions and then use this analysis to reorder those instructions to achieve better performance.

Post-Link Optimization for Linux on POWER

Post-Link Optimization (alphaworks.ibm.com/tech/fdprpro) optimizes the executable image of a program by collecting information on the behavior of the program while the program is running a typical workload. It then re-analyzes the program (together with the collected profile), applies global optimizations (including program restructuring), and creates a new version of the program executable that is optimized for that workload. The new program generated by the optimizer typically runs faster and uses less real memory than the original program. Particularly, the tool optimizes the program to achieve the following:

- Reduced number of memory accesses
- Reduced number of branches
- Better hit/miss I-cache ratio
- Reduced number of TLB misses
- Reduced number of page-faults

Post-Link Optimization for Linux on POWER was developed at IBM Haifa Research Lab; it currently accepts binaries produced by GCC 3.3, 3.4 and IBM XL C/C++ V7.0 for Linux on POWER. Post-Link Optimization runs on both SUSE LINUX Enterprise Server 9 and Red Hat Enterprise Linux 4.

The following table compares the features available in DTrace to those in the Linux performance analysis tools described above.

Table 1. Feature Comparison between DTrace and tools available for Linux on POWER

Features	DTrace	Linux
Dynamic Instrumentation (No impact to system performance when the tool is disabled).	**Yes**	**Yes** (SystemTap)
Work across kernel and user-space boundary	**Yes**	**Yes** (SystemTap)
Safe language and libraries for instrumentation	**Yes**	**Yes** (SystemTap)
Ability to speculatively trace data, and defer the decision to commit or discard at a later time	**Yes**	**Yes** (SystemTap)
Multiple handlers for the same probe	**Yes**	**Yes** (SystemTap)
Ability to leverage the hardware performance counters of the CPU	**No***	**Yes** (Oprofile, SystemTap[4])
Ability to view the flow of instructions through the processor models	**No**	**Yes** (IBM Performance Simulator)
Ability to recreate a new	**No**	**Yes** (Post-Link Optimization)

[4] Not currently available, but in plan for future releases.

version of a program that is specifically optimized for a particular set of workloads		
Ability to write arbitrary locations in kernel memory	**No**	**Yes** (SystemTap)
Ability to invoke arbitrary kernel subroutines.	**No**	**Yes** (SystemTap)

Memory Placement Optimization (MPO)

Solaris's MPO[5] provides performance improvements on systems in which each CPU accesses some area of memory more quickly than others. This architecture is also known as NUMA (Non-Uniform Memory Access). The essence of the NUMA architecture is the presence of multiple memory subsystems, as opposed to a single one on a SMP system. With MPO, Solaris is able to recognize the memory locality effects by ensuring that memory is as *close* as possible to the processors that access it, while still maintaining balance in the system to avoid bottlenecks. Solaris also provides several APIs[6] for developers who want to further optimize application performance through MPO.

The Linux community has made a tremendous effort to make the Linux kernel NUMA aware. The 2.6 kernel features NUMA awareness in the scheduler, so that the majority of processes execute in the *local* memory. For more information about how NUMA is supported in the kernel, please consult *Linux on NUMA Systems* (See Resources). Both SLES9 and RHEL4 are preconfigured with NUMA support. Applications for Linux on POWER do not require any code changes to take advantage of the NUMA support provided in the Linux kernel. If the application binds its processes to a processor, then the kernel will attempt to allocate memory *closest* to those processes. However, if programmers would like to take an extra step to further optimize the performance of their applications, they can also use the NUMA API to instruct the kernel where memory should be allocated and how. The Linux NUMA API enables applications to assign specific allocation behaviors (*policies*) to regions of their own virtual memory space. There is also a user-space `numactl` utility that controls NUMA policy for processes or shared memory. These user-space tool and APIs have been included in the RHEL4, and will be included in the second service pack for SLES9 on PPC64.

With the existence of multiple memory controllers in POWER5 multiprocessor-based systems, the performance of applications for Linux on POWER already benefit from the NUMA support in the Linux kernel without making any code changes. Note that some POWER5 processor-based systems such as the OpenPower™ 710 and the p5™ 510 have only one memory controller. As a result, NUMA support is irrelevant to these types of systems.

Table 2. NUMA support comparison

Attribute	Solaris 10	Linux
NUMA kernel support	**yes**	**yes**
NUMA APIs	**yes**	**yes**

[5] http://www.sun.com/software/solaris/performance.jsp
[6] http://iforce.sun.com/protected/solaris10/adoptionkit/tech/mpo/mpo_man.html

Multithreading Enhancements

In the last few Solaris releases, the performance of multithreaded applications have improved due to the enhancement of the threading library[7]. Beginning in the Solaris 9 operating system, the 1-on-1 thread model was adopted in preference to the historic M-on-N implementation. Both 1-on-1 and M-on-N represent the relationship between the kernel threads and the user-level threads. In the 1-on-1 model, each user-level thread is associated to an underlying kernel thread.

The thread-local storage (TLS) is also supported for the first time by the Sun Studio compilers for Solaris 10 running on x86 platforms. TLS is a mechanism by which variables are allocated such that there is one instance of the variable per extant thread. According to Sun[8], the combination of the new threads model and the latest Java Virtual Machine (JVM) technology has significantly improved SPECjbb2000[9] performance.

The Native POSIX Thread Library (NPTL) for Linux also uses the 1-on-1 thread model. Both SLES9 and RHEL4 for POWER processor-based systems include support for NPTL. And, the SPECjbb2000 benchmark for the 4-way and 2-way systems indicates a much better result[10] for Linux running on POWER5 processor-based systems.

The Linux 2.6 kernel also incorporates several thread-related improvements. The Linux kernel is preemptive, that is, some kernel-space operations can be interrupted to yield to user processes. This significantly benefits GUI applications that require maximum responsiveness. Also, the O(1)-scheduler allows the kernel to efficiently handle a greater number of threads than previous versions. Thread creation and destruction are now faster and less costly. The PPC64 enabling for thread-local storage and NPTL has been fully supported in GNU C library since the version 2.3.3[11]. Both GNU GCC and IBM XL Compilers for Linux on POWER also support thread-local storage as an extension to the C Language Family.

The following tables compare the results of the SPECjbb2000 benchmark, at the time of this writing, between the 4-way Solaris 10 systems using the 64-bit JVM and the 4-way Linux on POWER5 processor-based systems using the 32-bit JVM.

Table 3. The SPECjbb2000 benchmark comparison (4-way systems)

System	JVM	#CPU	Published	Result (ops/s)
Sun Fire V40z	Java HotSpot 64–bit Server VM on Solaris 10 /AMD64, version 1.5.0_02	4 cores, 4 chips	Mar. 2005	**116142**
IBM eServer OpenPower 720	IBM J2RE 1.4.2 (32-bit) RHEL4/POWER5	4 cores, 2 chips (SMT on)	Mar. 2005	**136261**

[7] http://www.sun.com/software/solaris/performance.jsp
[8] http://www.sun.com/software/solaris/benchmarks.jsp
[9] The SPECjbb2000 benchmark is for evaluating the performance of servers running typical Java business applications. It can be used to evaluate performance of hardware and software aspects of Java Virtual Machine (JVM) servers.
[10] http://www.spec.org/jbb2000/results/
[11] SLES9 uses glibc 2.3.3 and RHEL4 uses glibc 2.3.4.

IBM eServer p5 570	IBM J2RE 1.4.2 (32-bit) SLES9/POWER5	4 cores, 2 chips (SMT on)	Sep. 2004	**160995**

The following table provides the comparison for the 2-way systems.

Table 4. The SPECjbb2000 benchmark comparison (2-way systems)

System	JVM	#CPU	Published	Result (ops/s)
Sun Fire V20z	Java HotSpot 64–bit Server VM on Solaris 10 /AMD64, version 1.5.0_02	2 cores, 2 chips	Mar,05	**65840**
IBM eServer p5 570	IBM J2RE 1.4.2 (32-bit) SLES9/POWER5	2 cores, 1 chip (SMT on)	Sep, 04	**82615**

The results have clearly shown that Linux on POWER outperforms Solaris 10 in the SPECjbb2000 benchmark.

Summary

In this article, I've introduced three Solaris 10 features: DTrace, Memory Placement Optimization and Multithreading enhancements. For each feature, I've provided a technical analysis and comparison of what Solaris 10 is offering to similar features available for Linux on POWER.

The conclusion is clear; Linux provides a wide range of tools and technologies that are technically comparable, or better, than those offered in Solaris 10. In addition, the Linux global community has demonstrated its ability to continually enhance the quality of Linux and accomplish it in a timely fashion. The combination of Linux and the IBM Power architecture has undoubtedly proven to be one of the best platforms for business-critical applications.

In the next article in the series, I will compare the Solaris 10 Container technology with the virtualization functions available in the Linux operating system on IBM POWER5 processor-based systems. I will also compare the Predictive Self-Healing capabilities in Solaris 10 with those available in Linux on POWER.

Resources

- OProfile's Home page: http://oprofile.sf.net/about/
- Identify performance bottlenecks with OProfile for Linux on POWER: http://www-128.ibm.com/developerworks/linux/library/l-pow-oprofile/
- Kernel debugging with Kprobes: http://www-106.ibm.com/developerworks/library/l-kprobes.html?ca=dgr-lnxw42Kprobe
- Gaining insight into the Linux kernel with Kprobes: http://www.redhat.com/magazine/005mar05/features/kprobes/
- Dynamic Instrumentation of Production Systems (DTrace): http://www.sun.com/bigadmin/content/dtrace/dtrace_usenix.pdf
- SystemTap's Home Page: http://sourceware.org/systemtap/

- Architecture of systemtap: a Linux trace/probe tool : http://sourceware.org/systemtap/archpaper-0505.pdf
- Locating System Problems Using Dynamic Instrumentation by Vara Prasad et al. *Proceeding of Linux Symposium.* July, 2005, Ottawa Canada.
- Linux on NUMA systems: http://www.kernel.org/pub/linux/kernel/people/mbligh/presentations/OLS2004-numa_paper.pdf
- Anton Blanchard's Patch for Hugetlb Documentation: http://lkml.org/lkml/2005/1/1/50
- Linux on POWER: Distribution migration and binary compatibility considerations: http://www-128.ibm.com/developerworks/eserver/library/es-bincomp/
- Big Servers—2.6 compared to 2.4, *Proceedings of the Linux Symposium,* July, 2004: http://www.linuxsymposium.org/proceedings/reprints/Reprint-Coekaerts-OLS2004.pdf
- Guide to porting from Solaris to Linux on POWER: http://www-106.ibm.com/developerworks/linux/library/l-pow-portsolaris/
- Five easy-to-use performance tools for Linux on PowerPC: http://www-128.ibm.com/developerworks/eserver/library/es-PerformanceInspectoronpLinux.html

About the author

Chakarat Skawratananond is a Linux technical consultant for the eServer Solutions Enablement organization at IBM based in Austin, Texas. Chakarat's main role is to help solution developers bring their applications to Linux on POWER. Chakarat received his B. Eng in Electrical Engineering from Chulalongkorn University, Bangkok, Thailand. He earned a MS and a PhD in Electrical and Computer Engineering from the University of Texas at Austin.

Acknowledgements

I would like to thank the Linux on POWER technical competency team for their review and Linda Kinnunen for the document template and editorial review. My special thanks go to Bill Buros, David A. Desrosiers, Gordon Haubenschild, Steve Dibbell, Nam Keung, and Vara Prasad. Their detailed comments have helped improve this document tremendously.

Trademarks and special notices

References in this document to IBM products or services do not imply that IBM intends to make them available in every country.

The following terms are trademarks of International Business Machines Corporation in the United States, other countries, or both:
eServer
IBM
OpenPower
POWER4
POWER4+
POWER5
Power Architecture
PowerPC

Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Intel, Intel Inside (logos), MMX and Pentium are trademarks of Intel Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

Information is provided "AS IS" without warranty of any kind.

All customer examples described are presented as illustrations of how those customers have used IBM products and the results they may have achieved. Actual environmental costs and performance characteristics may vary by customer.

Information concerning non-IBM products was obtained from a supplier of these products, published announcement material, or other publicly available sources and does not constitute an endorsement of such products by IBM. Sources for non-IBM list prices and performance numbers are taken from publicly available information, including vendor announcements and vendor worldwide homepages. IBM has not tested these products and cannot confirm the accuracy of performance, capability, or any other claims related to non-IBM products. Questions on the capability of non-IBM products should be addressed to the supplier of those products.

All statements regarding IBM future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only. Contact your local IBM office or IBM authorized reseller for the full text of the specific Statement of Direction.

Some information addresses anticipated future capabilities. Such information is not intended as a definitive statement of a commitment to specific levels of performance, function or delivery schedules with respect to any future products. Such commitments are only made in IBM product announcements. The information is presented here to communicate IBM's current investment and development activities as a good faith effort to help with our customers' future planning.

Performance is based on measurements and projections using standard IBM benchmarks in a controlled environment. The actual throughput or performance that any user will experience will vary depending upon considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve throughput or performance improvements equivalent to the ratios stated here.