IBM WebSphere Real Time for Linux
Versión 3

Guía del usuario

IBM WebSphere Real Time for Linux
Versión 3

Guía del usuario

Nota

Antes de utilizar esta información y el producto al que da soporte, lea la información del apartado “Avisos” en la página 79.

Quinta edición (febrero de 2014)

Esta edición de la guía del usuario se aplica a IBM WebSphere Real Time for Linux, versión 3, y a todos los releases y modificaciones posteriores, hasta que se indique otra cosa en ediciones nuevas.

Contenido

Figuras

Tablas

Prefacio

Esta guía del usuario ofrece información general sobre IBM® WebSphere Real Time for Linux.

Capítulo 1. Introducción

Aquí se proporciona información sobre IBM WebSphere Real Time for Linux.

Las nuevas modificaciones realizadas en esta guía del usuario se indican mediante barras verticales a la izquierda de los cambios.

Otra información más reciente sobre IBM WebSphere Real Time for Linux que no esté disponible en la guía de usuario puede consultarse aquí: http://www.ibm.com/support/docview.wss?uid=swg21501145

Visión general de WebSphere Real Time for Linux

Funciones en tiempo real de paquetes de WebSphere Real Time for Linux con IBM J9 virtual machine (JVM).

WebSphere Real Time for Linux es un entorno de ejecución Java™ con un kit de desarrollo de software que amplía el SDK de IBM para Java con prestaciones en tiempo real. Las aplicaciones que dependen de tiempos de respuesta precisos pueden aprovechar las funciones de tiempo real que se proporcionan con WebSphere Real Time for Linux en la tecnología Java estándar.

Características

Las aplicaciones de tiempo real necesitan tiempo de ejecución coherente en lugar de una velocidad absoluta.

Las principales preocupaciones a la hora de desplegar aplicaciones de tiempo real con las JVM tradicionales son las siguientes:
- Retardos imprevisible (potencialmente largos) por la actividad de recogida de basura.
- Retardos en el tiempo de ejecución del método mientras se producen la compilación JIT (Just-In-Time) y la recompilación, con variabilidad en el tiempo de ejecución.
- Planificación arbitraria del sistema operativo.

WebSphere Real Time for Linux elimina estos obstáculos proporcionando:
- Recopilador de basura Metronome, un recopilador de basura incremental y determinante, con tiempos de pausa muy breves.

Novedades

En este tema se presentan los cambios para IBM WebSphere Real Time for Linux.

WebSphere Real Time for Linux V3

WebSphere Real Time for Linux V3 es una ampliación al SDK para Java V7 de IBM, construcción sobre las características y funciones disponibles con este release

para incluir prestaciones en tiempo real. Las versiones anteriores de WebSphere Real Time for Linux estaban basadas en releases anteriores del SDK para Java de IBM.

Para obtener más información sobre las novedades de SDK para Java 7 de IBM, consulte: Novedades en el Information Center del SDK para Java 7 de IBM.

Las nuevas modificaciones realizadas en esta guía del usuario se indican mediante barras verticales a la izquierda de los cambios.

Planificación de hebras de Java mediante políticas de planificación de Linux

Desde la renovación de servicio 1, puede planificar hebras regulares de Java con la política de programación **SCHED_RR** para ajustar aplicaciones en tiempo real. Para obtener más información, consulte "Planificación de hebras" en la página 6.

Control de los tiempos de pausa para la Recogida de basura Metronome

De forma predeterminada, la Recogida de basura Metronome se pausa durante 3 milisegundos entre los ciclos de recogida de basura. Puede cambiar este valor para controlar el tiempo de pausa usando una nueva opción de línea de mandatos. Para obtener más información sobre esta opción, consulte "Control de los tiempos de pausa" en la página 23.

Referencias comprimidas

La Recogida de basura Metronome tiene ahora soporte tanto para referencias no comprimidas como para referencias comprimidas en plataformas de 64 bits. Para implicaciones de rendimiento, consulte Capítulo 7, "Rendimiento", en la página 31.

Ventajas

Las ventajas del entorno de tiempo real son que las aplicaciones Java se ejecutan con un mayor grado de predicción que con la JVM estándar y ofrecen un comportamiento de temporización coherente para su aplicación Java. Las actividades de fondo, como la compilación y la recogida de basura, se producen en momentos determinados y eliminan cualquier pico inesperado de actividad de fondo durante la ejecución de la aplicación.

Estas ventajas se obtienen ampliando la JVM con la tecnología de recogida de basura en tiempo real de Metronome.

Accesibilidad

Las características de accesibilidad permiten que los usuarios con discapacidades, por ejemplo, con limitaciones en la movilidad o la vista, puedan utilizar productos de tecnologías de la información sin problemas.

IBM se esfuerza en proporcionar productos con acceso apto para todos los usuarios, independientemente de la edad o capacidad.

Por ejemplo, puede utilizar WebSphere Real Time for Linux sin ratón, utilizando solo el teclado.

Si desea leer acerca de estas cuestiones que afectan a la accesibilidad de IBM SDK para Java V7 subyacente, consulte IBM Information Center. No hay problemas de accesibilidad que afecten a una determinada prestación o función en WebSphere Real Time for Linux.

Navegación del teclado

Este producto utiliza las teclas de navegación estándar de Microsoft Windows.

Los usuarios que necesiten el desplazamiento mediante teclas pueden consultar la descripción de las pulsaciones más útiles para las aplicaciones Swing en Swing Key Bindings.

IBM y accesibilidad

Consulte el apartado IBM Human Ability and Accessibility Center para obtener más información sobre el compromiso que IBM tiene con la accesibilidad.

Capítulo 2. Información acerca de IBM WebSphere Real Time for Linux

En esta sección se presentan componentes clave de IBM WebSphere Real Time for Linux.

Introducción al recopilador de basura Metronome

El recopilador de basura Metronome sustituye al recopilador de basura estándar en WebSphere Real Time for Linux.

La diferencia principal entre la recogida de basura Metronome y la recogida de basura estándar es que la recogida de basura Metronome se produce en pequeños pasos que se pueden interrumpir, mientras que la recogida de basura estándar detiene la aplicación mientras marca y recoge la basura.

Por ejemplo:

```
java -Xgcpolicy:metronome -Xgc:targetUtilization=80 su_aplicación
```

El ejemplo especifica que su aplicación se ejecuta durante el 80% de cada 60 ms. El 20% del tiempo se utiliza para la recogida de basura, si hay basura que recoger. El recopilador de basura Metronome garantiza la utilización de los niveles siempre que se le hayan otorgado recursos suficientes. La recogida de basura empieza cuando el espacio libre del almacenamiento dinámico pasa a estar por debajo de un umbral determinado de forma dinámica.

Recogida de basura y descarga de clases Metronome

Metronome admite la descarga de clases de la misma manera que un Java Developer Kit estándar. Sin embargo, debido al trabajo implicado, al descargar las clases se pueden producir valores de tiempo poco habituales durante las actividades de recogida de basura.

Hebras del recopilador de basura Metronome

El recopilador de basura Metronome consta de dos tipos de hebras: una hebra de alarma simple y varias hebras de la recogida de basura. De forma predeterminada, la recogida de basura utiliza una hebra para cada procesador lógico a disposición del sistema operativo. Esto permite el procesamiento paralelo más eficaz durante los ciclos de recogida de basura. Un ciclo de recogida de basura se refiere al tiempo transcurrido desde que se desencadena el recopilador de basura y la finalización de la liberación de basura. Según el tamaño de almacenamiento dinámico de Java, el tiempo transcurrido para un ciclo completo de recogida de basura puede ser de unos segundos. Un ciclo de recogida de basura suele contener cientos de cantidades del recopilador de basura. Estas cantidades son pausas muy breves en el código de aplicación, normalmente de unos 3 milisegundos. Utilice **-verbose:gc** para obtener los informes de resumen de ciclos y cantidades. Para obtener más información, consulte el apartado: "Utilización de la información de verbose:gc" en la página 66. Puede definir la cantidad de hebras de recogida de basura de la JVM utilizando la opción **-Xgcthreads**.

No existe ninguna ventaja derivada de incrementar **-Xgcthreads** por encima del valor predeterminado. La reducción de **-Xgcthreads** puede reducir la carga global de la CPU durante los ciclos de recogida de basura, aunque los ciclos de recogida de basura se prolongarán.

Nota: Los objetivos de duración de las cantidades de recogida de basura se mantendrán constantes en 3 milisegundos.

No puede cambiar el número de hebras de alarma de la JVM.

El recopilador de basura Metronome comprueba periódicamente la JVM para comprobar si el almacenamiento dinámico tiene suficiente espacio libre. Cuando la cantidad de espacio libre pase a estar por debajo del límite, el recopilador de basura Metronome desencadenará la JVM para que se inicie la recogida de basura.

Hebra de alarma
La hebra de alarma sencilla garantiza la utilización de los mínimos recursos. Se "activa" en intervalos regulares y realiza estas comprobaciones:
- La cantidad de espacio libre de la memoria de almacenamiento dinámico
- Si la recogida de basura se está realizando actualmente

Si no hay suficiente espacio libre disponible y no se está realizando la recogida de basura, la hebra de alarma desencadena las hebras de recogida para iniciar la recogida de basura. La hebra de alarma no realiza ninguna acción hasta la próxima vez que compruebe la JVM, según lo planificado.

Hebras de recogida
Las hebras de recogida realizan la recogida de basura.

Una vez completado el ciclo de recogida de basura, el recopilador de basura Metronome comprueba el espacio de almacenamiento dinámico liberado. Si sigue sin ser suficiente espacio de almacenamiento dinámico libre, se inicia otro ciclo de recogida de basura que utiliza el mismo ID de desencadenante. Si hay suficiente espacio de almacenamiento dinámico libre, el desencadenante finaliza y se detienen las hebras de recogida de basura. La hebra de alarma sigue supervisando el espacio de almacenamiento dinámico libre y desencadenará otro ciclo de recogida de basura cuando sea necesario.

Para obtener más información sobre el uso de la Recogida de basura metronome, consulte "Utilización del recopilador de basura Metronome" en la página 23.

Planificación de hebras

Las políticas de planificación de Linux se pueden utilizar con hebras regulares de Java para ajustar aplicaciones de tiempo real.

Con WebSphere Real Time for Linux, puede ejecutar las hebras regulares de Java con la política de programación de SCHED_RR. La utilización de la política SCHED_RR le ofrece un mayor control sobre su aplicación, lo que puede mejorar el rendimiento en tiempo real de las hebras de Java. La máquina virtual Java detecta la prioridad y la política de la hebra principal cuando Java se inicia con la política SCHED_RR. La JVM altera las correlaciones de política y prioridad en consecuencia. Para obtener más información sobre la modificación de políticas y prioridades de hebras regulares de Java, consulte "Asignación y planificación de hebras" en la página 19.

Entre las políticas de planificación de Linux se incluyen las siguientes:

SCHED_OTHER
Política de planificación de compartición de tiempo universal predeterminada que utilizan la mayoría de las hebras. Estas hebras se deben asignar con una prioridad de cero.

SCHED_OTHER utiliza la distribución en porciones de tiempo, lo que significa que cada hebra se ejecuta durante un período de tiempo limitado, tras el cual se puede ejecutar la siguiente hebra.

SCHED_FIFO
Se puede utilizar solo con prioridades mayores que cero. Cuando una hebra SCHED_FIFO pasa a estar disponible, esta tiene prioridad sobre cualquier hebra SCHED_OTHER normal.

Si una hebra SCHED_FIFO con una prioridad más alta pasa a estar disponible, esta tiene prioridad sobre las demás hebras SCHED_FIFO existentes con prioridad más baja. Entonces, esta hebra se mantiene en lo alto de la cola según su prioridad.

No hay distribución en porciones de tiempo.

Nota: WebSphere Real Time for Linux no utiliza SCHED_FIFO.

SCHED_RR
Es una mejora de SCHED_FIFO. La diferencia es que cada hebra se puede ejecutar solo durante un período de tiempo limitado. Si la hebra supera dicho tiempo, se devuelve a la lista según su prioridad. SCHED_RR puede ser utilizado por WebSphere Real Time for Linux V3.

Para obtener más detalles sobre estas políticas de planificación de Linux, consulte la página **man** para **sched_setscheduler**.

Para obtener más información sobre el uso de las políticas de planificación de Linux con WebSphere Real Time for Linux, consulte “Asignación y planificación de hebras” en la página 19.

Capítulo 3. Planificación

Lea esta sección antes de instalar WebSphere Real Time for Linux.

-
- “Entornos soportados”
-
- “Consideraciones” en la página 10

Migración

Puede ejecutar sus aplicaciones Java estándar en WebSphere Real Time for Linux sin modificación.

Entornos soportados

IBM WebSphere Real Time for Linux tiene soporte en determinadas plataformas de hardware y sistemas operativos.

IBM WebSphere Real Time for Linux

Están soportados las arquitecturas de las plataformas siguientes:

- Intel Architecture, 32 bits (IA-32)
  - Pentium 4
  - Pentium Xeon
  - Pentium M
  - Pentium D y equivalentes
- AMD64/EM64T
- IBM POWER 32
- IBM POWER 64

Nota: Ya no hay soporte para el hardware de Pentium 3.

Están soportados los sistemas operativos siguientes:

Tabla 1. Entornos de Linux comprobados

Hardware	**IA-32 32 bits**	**AMD64/EM64T 64 bits**	
Espacio de direcciones SDK	**32 bits**	**32 bits**	**64 bits**
RHEL 5 Update 7	Sí	Sí	Sí
RHEL 6 Update 1	Sí	Sí	Sí
SLES 11 Service Pack 2	Sí	Sí	Sí
Ubuntu 8.04	Sí	Sí	Sí
Ubuntu 10.04	Sí	Sí	Sí

Nota: No hay soporte para SLES 9, SLES 10 y RHEL 4.

Consideraciones

Debe tener en cuenta algunos factores cuando utilice WebSphere Real Time for Linux.

- Cuando sea posible, no ejecute más de una JVM en tiempo real en el mismo sistema. La razón es que tendría varios recopiladores de basura. Una JVM no conoce las áreas de memoria de las demás. Un efecto es que los ciclos y tiempos de pausa de GC no se pueden coordinar entre las JVM, lo que significa que es posible que una JVM afecte negativamente al rendimiento de GC de otra JVM. Si tiene que utilizar varias JVM, asegúrese de que cada JVM esté vinculada a un subconjunto específico de procesadores mediante el mandato **taskset**.
- Las memorias caché compartidas usadas por releases anteriores de WebSphere Real Time for Linux para almacenar clases y código precompilado no son compatibles con las memorias caché utilizadas por este release de WebSphere Real Time for Linux. Debe volver a generar el contenido de las memorias caché anteriores.
- Si utiliza memorias caché de clase compartida, el nombre de la memoria caché no debe superar los 53 caracteres.

Capítulo 4. Instalación de WebSphere Real Time for Linux

Siga estos pasos para instalar el producto.

Archivos de instalación

Necesita estos archivos de instalación

IBM WebSphere Real Time for Linux se proporciona en dos tipos de paquetes InstallAnywhere.

Paquetes instalables
Los Paquetes instalables configuran su sistema. Por ejemplo, los programas podrían establecer variables de entorno.

- `wrt-3.0-0.0-linux-<arch>-sdk.bin`
- `wrt-3.0-0.0-linux-<arch>-jre.bin`

Paquetes de archivo
Estos paquetes extraen los archivos en su sistema, pero no realizan configuración alguna.

- `wrt-3.0-0.0-linux-<arch>-sdk-archive.bin`
- `wrt-3.0-0.0-linux-<arch>-jre-archive.bin`

Nota: <arch> es la arquitectura de su plataforma; x86_32 o x86_64.

Instalación desde un paquete InstallAnywhere

Estos paquetes proporcionan un programa interactivo que le guía durante las opciones de instalación. Puede ejecutar el programa como una interfaz gráfica de usuario o desde una consola del sistema.

Antes de empezar

El sistema debe tener las siguientes bibliotecas compartidas:

- GNU C Library V2.3 (glibc)
- `libstdc++.so.5`

Si no tiene la biblioteca compartida `libstdc++.so.5`, es posible que se produzca un volcado Java core al realizar la instalación, con los siguientes errores:

```
JVMJ9VM011W Unable to load j9dmp24: libstdc++.so.5: cannot open shared object file:
No such file or directory
JVMJ9VM011W Unable to load j9gc24: libstdc++.so.5: cannot open shared object file:
No such file or directory
JVMJ9VM011W Unable to load j9vrb24: libstdc++.so.5: cannot open shared object file:
No such file or directory
```

Si está instalando un paquete instalable, debe tener la herramienta rpm-build instalada en el sistema, de lo contrario el programa de instalación no podrá registrar el nuevo paquete en la base de datos de RPM. Para averiguar si la herramienta rpm-build está instalada, especifique el mandato siguiente:

```
rpm -q rpm-build
```

Acerca de esta tarea

Los paquetes InstallAnywhere tienen una extensión de archivo `.bin`.

Existen dos tipos de paquetes:

Instalable
Al instalar estos paquetes también se configura el sistema, por ejemplo estableciendo las variables de entorno.

De archivo
Al instalar estos paquetes se extraen los archivos del sistema, pero no se realiza ninguna configuración.

Procedimiento

- Para instalar el paquete de una forma interactiva, complete una instalación atendida.
- Para instalar el paquete sin ninguna interacción adicional del usuario, complete una instalación desatendida. Puede seleccionar esta opción si desea instalar muchos sistemas.
- Cuando se completa el proceso de instalación, siga los pasos de configuración de esta sección, como el establecimiento de las variables de entorno path y classpath.

Resultados

El producto se instala.

Nota: No interrumpa el proceso de instalación, por ejemplo pulsando Ctrl+C. Si interrumpe el proceso, puede que tenga que reinstalar el producto. Para obtener más información, consulte “Instalación interrumpida” en la página 15.

Si utiliza un paquete instalable, podrá ver mensajes que le notificarán en caso de encontrar algún problema. La instalación de los paquetes de archivo no genera ningún mensaje. Algunos de los mensajes que podrá ver si utiliza un paquete instalable se muestran en la siguiente lista:

`El instalador no se puede ejecutar en la configuración. Va a abandonar.`
Este mensaje de error se genera cuando ni ID de usuario no está autorizado para ejecutar el proceso de instalación. Como no puede continuar, el programa de instalación finaliza. Para arreglar el problema, inicie de nuevo la instalación pero con un ID de usuario que tenga autorización de usuario root.

Ya hay un paquete RPM instalado. Desinstale el paquete antes de continuar.
Este mensaje indica que ya hay un paquete RPM instalado. Como no puede continuar, el programa de instalación finaliza. Para solucionar el problema, desinstale el paquete RPM antes de continuar.

Finalización de una instalación atendida

Instale el producto desde un paquete InstallAnywhere, de forma interactiva.

Antes de empezar

Compruebe las condiciones siguientes antes de empezar el proceso de instalación:

- Si ha instalado con anterioridad WebSphere Real Time for Linux desde un paquete RPM, deberá desinstalar este paquete antes de continuar.
- Deberá tener un ID de usuario con autorización de usuario root.

Procedimiento

1. Descargue el archivo del paquete de instalación en un directorio temporal.
2. Vaya al directorio temporal.
3. Inicie el proceso de instalación escribiendo `./package` en un indicador de shell, donde *paquete* es el nombre del paquete que está instalando.
4. Seleccione un idioma de la lista que aparece en la ventana del instalador y, a continuación, pulse **Siguiente**. La lista de idiomas disponibles se basa en el valor de entorno local de su sistema.
5. Lea el acuerdo de licencia, utilice la barra de desplazamiento para leer el texto de licencia hasta el final. Para proseguir con la instalación deberá aceptar los términos del acuerdo de licencia. Para aceptar los términos, seleccione el botón de selección y, a continuación, pulse **Aceptar**.

   Nota: No seleccione el botón de selección para aceptar el acuerdo de licencia hasta que haya leído todo el texto de licencia.
6. Se le solicitará que elija un directorio de destino para la instalación. Si no desea realizar la instalación en el directorio predeterminado, pulse **Elegir** para seleccionar un directorio alternativo utilizando la ventana de navegador. Cuando haya seleccionado el directorio de instalación, pulse **Siguiente** para continuar.
7. Se le solicitará que revise las opciones que hay realizado. Para cambiar su selección, pulse **Anterior**. Si sus opciones son correctas, pulse **Instalar** para continuar con la instalación.
8. Cuando el proceso de instalación haya finalizado, pulse **Hecho** para terminar.

Finalización de una instalación desatendida

Si tiene que instalar más de un sistema y ya sabe las opciones de instalación que desea utilizar, tal vez quiera utilizar un proceso de instalación desatendida. Instale uno mediante el proceso de instalación atendida y, a continuación, utilice el archivo de respuestas resultante para completar las demás instalaciones sin ninguna interacción adicional del usuario.

Procedimiento

1. Cree un archivo de respuestas completando una instalación atendida. Utilice una de las opciones siguientes:

- Utilice la GUI y especifique que el programa de instalación cree un archivo de respuestas. El archivo de respuestas se denomina `installer.properties` y se crea en el directorio de instalación.
- Utilice la línea de comandos y añada la opción -r al mandato de instalación atendida, especificando la vía de acceso completa al archivo de respuestas. Por ejemplo:

  `./paquete -r /vía de acceso/installer.properties`

Contenido del archivo de respuestas de ejemplo:

```
INSTALLER_UI=silent
USER_INSTALL_DIR=/mi_directorio
```

En este ejemplo, */mi_directorio* es el directorio de instalación de destino que seleccionó para la instalación.

2. Opcional: En caso necesario, edite el archivo de respuestas para cambiar opciones.

   Nota: Los paquetes de archivo presentan el siguiente problema conocido: las instalaciones que utilizan un archivo de respuestas usan el directorio predeterminado incluso aunque cambie el directorio en el archivo de respuestas. Si existe una instalación anterior en el directorio predeterminado, se sobrescribe.

   Si crea más de un archivo de respuestas, cada uno con diferentes opciones de instalación, especifique un nombre único para cada archivo de respuesta, con el formato *myfile*`.properties`.
3. Opcional: Genere un archivo de registro. Como está instalando en modalidad silenciosa, no se muestra ningún mensaje de estado al final del proceso de instalación. Para generar un archivo de registro que contenga el estado de la estado, complete los pasos siguientes:
   a. Defina las propiedades del sistema necesarias mediante el mandato siguiente.

      ```
      export _JAVA_OPTIONS="-Dlax.debug.level=3 -Dlax.debug.all=true"
      ```
   b. Defina la siguiente variable de entorno para enviar la salida de registro a la consola:

      ```
      export LAX_DEBUG=1
      ```
4. Inicie una instalación desatendida mediante la ejecución del instalador del paquete con la opción silenciosa **-i** y la opción **-f** para especificar el archivo de respuestas. Por ejemplo:

   ```
   ./package -i silent -f /path/installer.properties 1>console.txt 2>&1
   ./package -i silent -f /path/myfile.properties 1>console.txt 2>&1
   ```

   Puede utilizar una vía de acceso completa o relativa para el archivo de propiedades. En estos ejemplos, la cadena `1>console.txt 2>&1` redirecciona la información del proceso de instalación desde las secuencias stderr y stdout al archivo de registro `console.txt` en el directorio actual. Revise este archivo de registro si cree que ha habido un problema con la instalación.

   Nota: Si el directorio de instalación contiene varios archivos de respuestas, se utiliza el archivo de respuestas predeterminado, `installer.properties`.

Instalación interrumpida

Si el instalador de paquete se detiene inesperadamente durante la instalación, por ejemplo si pulsa Ctrl+C, la instalación será errónea y no podrá desinstalar o reinstalar el producto. Si intenta desinstalar o volver a instalar es posible que vea el mensaje `Error de aplicación fatal`.

Acerca de esta tarea

Para resolver este problema, suprima los archivos y vuelva a instalarlos, tal como se describe en los pasos siguientes.

Procedimiento

1. Suprima el archivo de registro `/var/.com.zerog.registry.xml`.
2. Suprima el directorio que contiene la instalación, en el caso de que se haya creado. Por ejemplo, /opt/IBM/javawrt3[_64]/.
3. Ejecute el programa de instalación de nuevo.

Problemas y limitaciones conocidos

Los paquetes InstallAnywhere tienen algunos problemas y limitaciones conocidas.

- Si no tiene la biblioteca compartida `libstdc++.so.5` en su sistema, la instalación fallará, produciendo un volcado Java core. Para obtener más información, consulte "Instalación desde un paquete InstallAnywhere" en la página 11.
- La GUI de paquete de instalación no da soporte al programa de lector de pantalla Orca. Puede utilizar la instalación desatendida como una alternativa a la GUI.
- Si, tras la instalación, especifica `./package` para iniciar de nuevo el programa, el programa muestra el siguiente mensaje:

  ```
  ENTER THE NUMBER OF THE DESIRED CHOICE, OR PRESS <ENTER> TO ACCEPT THE DEFAULT:
  ```

  Si pulsa la tecla Intro para aceptar el valor predeterminado, el programa no responde. Escriba un número y, a continuación, pulse Intro.
- Si instala el paquete; a continuación, instálelo de nuevo en una modalidad diferente, por ejemplo, consola o silencio, es posible que vea el mensaje de error siguiente:

  ```
  Invocation of this Java Application has caused an InvocationTargetException.
  This application will now exit
  ```

  No debería ver este mensaje si ha realizado la instalación con la modalidad GUI y está ejecutando el programa de instalación de nuevo en la modalidad de consola. Si ve este error y está ejecutando el programa para seleccionar la opción de desinstalación (solo paquetes instalables), utilice en su lugar el mandato **`./_uninstall/uninstall`**, tal como se describe en "Desinstalación de WebSphere Real Time for Linux" en la página 18.

Sólo paquetes instalables

- No puede actualizar una instalación existente mediante los paquetes InstallAnywhere. Para actualizar WebSphere Real Time for Linux, primero debe desinstalar cualquier versión anterior.
- No puede instalar dos instancias distintas de la misma versión de WebSphere Real Time for Linux en el mismo sistema y al mismo tiempo, incluso si utiliza directorios de instalación diferentes. Por ejemplo, no puede tener simultáneamente WebSphere Real Time for Linux V3 en el directorio `/previous`, y una instalación actualizada del servicio WebSphere Real Time for Linux en el

directorio /current. El programa de instalación comprueba el número de versión. Si el programa encuentra un paquete existente con el mismo número de versión, se le solicita que desinstale el paquete existente.

- Si el paquete está instalado y ejecuta el instalador del paquete de nuevo mediante la GUI, puede seleccionar desinstalar el paquete. Esta opción de desinstalación no está disponible en la modalidad desatendida. Si ejecuta el instalador del paquete en modalidad desatendida, el programa se ejecuta pero no realiza ninguna acción.

Sólo paquetes de archivo

- Si cambia el directorio de instalación en un archivo de respuestas y, a continuación, ejecuta una instalación desatendida mediante ese archivo de respuesta, el programa de instalación ignora el nuevo directorio de instalación y utiliza el directorio predeterminado. Si existe una instalación anterior en el directorio predeterminado, se sobrescribe.

Configuración de la vía de acceso

Cuando haya definido la variable de entorno **PATH**, podrá ejecutar una aplicación o un programa escribiendo su nombre en un indicador de shell.

Acerca de esta tarea

Nota: Si altera la variable de entorno **PATH** como se ha descrito en esta sección, alterará temporalmente los archivos Java ejecutables existentes en su vía de acceso.

Puede especificar la vía de acceso a una herramienta escribiendo siempre la vía de acceso antes del nombre de la herramienta. Por ejemplo, si el SDK está instalado en /opt/IBM/javawrt3[_64]/, puede compilar un archivo denominado *myfile.java* escribiendo el mandato siguiente en el indicador de shell:

```
/opt/IBM/javawrt3[_64]/bin/javac myfile.java
```

Para no tener que escribir cada vez la vía de acceso completa:

1. Edite el archivo de arranque de shell en el directorio local (normalmente **.bashrc**, dependiendo del shell) y añada las vías de acceso absolutas a la variable de entorno **PATH**; por ejemplo:

   ```
   export PATH=/opt/IBM/javawrt3[_64]/bin:/opt/IBM/javawrt3[_64]/jre/bin:$PATH
   ```

2. Vuelva a iniciar una sesión o ejecute el script de shell actualizado para activar el nuevo valor de **PATH**.
3. Compile el archivo con la herramienta **javac**. Por ejemplo, para compilar el archivo *myfile.java*, en un indicador de shell, escriba:

   ```
   javac -Xgcpolicy:metronome myfile.java
   ```

   La variable de entorno **PATH** permite que Linux busque archivos ejecutables, como, por ejemplo, **javac**, **java** y la herramienta **javadoc**, desde cualquier directorio actual. Para visualizar el valor actual de la vía de acceso, escriba el mandato siguiente en un indicador de mandatos:

   ```
   echo $PATH
   ```

Qué hacer a continuación

Consulte el apartado “Establecimiento de classpath” en la página 17 para determinar si necesita definir su variable de entorno CLASSPATH.

Establecimiento de classpath

La variable de entorno **CLASSPATH** indica las herramientas de del SDK, como **java**, **javac** y **javadoc**, donde se encuentran las bibliotecas de clases Java.

Acerca de esta tarea

Defina la variable de entorno **CLASSPATH** explícitamente solo si se da una de estas condiciones:

- Si necesita un archivo de clase o de biblioteca diferente, por ejemplo, uno que haya desarrollado previamente, y no está en el directorio activo.
- Si cambia la ubicación de los directorios `bin` y `lib` de forma que ya no tienen el mismo directorio padre.
- Si tiene previsto desarrollar o ejecutar aplicaciones que utilicen entornos de ejecución diferentes en el mismo sistema.

Para visualizar el valor actual de **CLASSPATH**, escriba el mandato siguiente en un indicador del intérprete de mandatos:

```
echo $CLASSPATH
```

Si desarrolla y ejecuta aplicaciones que utilicen entornos de ejecución diferentes, incluyendo otras versiones que haya instalado por separado, debe establecer **CLASSPATH** y **PATH** de forma explícita para cada aplicación. Si ejecuta varias aplicaciones simultáneamente y utiliza entornos de ejecución diferentes, cada aplicación debe ejecutarse en su propio shell.

Si solo ejecuta una versión de Java en cada momento, puede utilizar un script de shell para pasar de un entorno de tiempo de ejecución a otro.

Qué hacer a continuación

Consulte el apartado "Prueba de la instalación" para verificar que ha realizado su instalación correctamente.

Prueba de la instalación

Utilice la opción **-version** para comprobar si su instalación es correcta.

Acerca de esta tarea

La instalación de Java consta de una JVM en tiempo real.

Procedimiento

Pruebe su instalación siguiendo estos pasos:

1. Para ver la información de la versión para la JVM en tiempo real, especifique el mandato siguiente en un indicador de shell:

   ```
   java -Xgcpolicy:metronome -version
   ```

   Este mandato devuelve los mensajes siguientes si es correcto:

   ```
   java versión "1.7.0"
   WebSphere Real Time V3(build pxi3270-20110428_04)
   IBM J9 VM (build 2.6, JRE 1.7.0 Linux x86-32 20110427_81014 (JIT enabled, AOT
   enabled)
   J9VM - R26_head_20110426_2022_B81001
   ```

```
JIT  - r11_20110426_19388
GC   - R26_head_20110426_1548_B80973
J9CL - 20110427_81014)
JCL - 20110427_03 based on Oracle 7b145
```

Nota: La información de la versión es correcta pero las fechas y arquitectura de la plataforma podrían ser distintas del ejemplo. El formato de la cadena de fecha es el siguiente: `yyyymmdd`, posiblemente seguido de información adicional específica del componente.

Desinstalación de WebSphere Real Time for Linux

El proceso que utiliza para eliminar WebSphere Real Time for Linux depende del tipo de instalación que haya empleado.

Antes de empezar

Para los paquetes instalables InstallAnywhere, deberá tener un ID de usuario con autorización de usuario root.

Acerca de esta tarea

No existe ningún proceso de desinstalación para los paquetes de archivo InstallAnywhere. Para eliminar un paquete de archivo desde el sistema, suprima el directorio de destino que seleccionó al instalar el paquete. Para los paquetes instalables InstallAnywhere, desinstale el producto mediante un mandato o bien ejecutando de nuevo el programa de instalación, como se describe en los pasos siguientes.

Procedimiento

- Opcional: Desinstálelo manualmente mediante el mandato **`uninstall`**.
  1. Acceda al directorio que contiene la instalación. Por ejemplo:
     ```
     cd /opt/IBM/javawrt3
     ```
  2. Inicie el proceso de desinstalación mediante el mandato siguiente:
     ```
     ./_uninstall/uninstall
     ```
- Opcional: Si no puede ubicar fácilmente el programa de desinstalación, puede ejecutar otra instalación atendida como alternativa. El programa de instalación detecta que el producto ya está instalado y, a continuación, le ofrece la oportunidad de desinstalar la instalación anterior.

Capítulo 5. Ejecución de aplicaciones de IBM WebSphere Real Time for Linux

Información importante para ayudarle en la ejecución de aplicaciones en tiempo real.

Asignación y planificación de hebras

El sistema operativo Linux admite varias políticas de planificación. La política de planificación de compartición de tiempo universal es SCHED_OTHER, que utilizan la mayoría de las hebras. Las hebras pueden utilizar SCHED_RR y SCHED_FIFO en aplicaciones en tiempo real. WebSphere Real Time for Linux solo utiliza SCHED_OTHER y SCHED_RR.

El kernel decide cuál es la siguiente hebra ejecutable que puede ejecutar el procesador. El kernel mantiene una lista de hebras ejecutables. Busca la hebra con la prioridad más alta y selecciona dicha hebra como la hebra que se va a ejecutar a continuación.

Las políticas y prioridades de hebras se pueden listar utilizando el mandato siguiente:

```
ps -emo pid,ppid,policy,tid,comm,rtprio,cputime
```

donde la política:

- `TS` es SCHED_OTHER
- `RR` es SCHED_RR
- `FF` es SCHED_FIFO
- `-` no tiene ninguna política notificada

La salida es similar a la de este ejemplo:

```
  PID  PPID POL   TID COMMAND         RTPRIO     TIME
31531 30800 -       - java                 - 00:00:13
    -     - RR  31531 -                    6 00:00:00
    -     - RR  31532 -                    6 00:00:13
    -     - RR  31533 -                    6 00:00:00
    -     - RR  31538 -                    6 00:00:00
    -     - RR  31539 -                    6 00:00:00
    -     - RR  31540 -                    6 00:00:00
    -     - RR  31541 -                    6 00:00:00
    -     - RR  31542 -                    6 00:00:00
    -     - RR  31543 -                    6 00:00:00
    -     - RR  31544 -                    6 00:00:00
    -     - RR  31545 -                    6 00:00:00
    -     - RR  31546 -                    6 00:00:00
```

Esta salida muestra el proceso Java y varias hebras con la política SCHED_RR y la prioridad 6.

Para consultar la política de planificación actual, utilice **sched_getscheduler**, o bien el mandato **ps** que se muestra en el ejemplo.

Para obtener más información sobre los procesos, consulte el apartado “Técnicas de depuración generales” en la página 36.

Políticas y prioridades de las hebras de Java común

Las hebras de Java común, es decir, las hebras asignadas como objetos `java.lang.Thread`, utilizan la política de programación predeterminada de SCHED_OTHER. Desde WebSphere Real Time for Linux V3 renovación de servicio 1, puede ejecutar las hebras regulares de Java con la política de programación de SCHED_RR.

De forma predeterminada, las hebras de Java se ejecutan utilizando la política SCHED_OTHER predeterminada. Esta política asigna las hebras de Java con la prioridad 0 del sistema operativo.

La utilización de la política SCHED_RR le ofrece un mayor control sobre su aplicación, lo que puede mejorar el rendimiento en tiempo real de las hebras de Java. La máquina virtual Java detecta la prioridad y la política de la hebra principal cuando Java se inicia con la política SCHED_RR. La JVM altera las correlaciones de política y prioridad en consecuencia. Todas las hebras de Java se ejecutan con la misma prioridad de sistema operativo que la hebra principal. Aunque a SCHED_RR se les pueden asignar las prioridades 1 - 99, las prioridades de SCHED_RR que se pueden utilizar para WebSphere Real Time for Linux V3 son las prioridades 1-10. Si la prioridad se define más alta que 10, la prioridad de la hebra principal baja a 10 y la correlación aplicada se basa en un valor de 10.

Una manera de modificar la propiedad de programación en tiempo real de un proceso en la línea de mandatos es utilizar el mandato **chrt**. En el ejemplo siguiente, la prioridad de la hebra principal de Java se ha definido para utilizar la política de programación SCHED_RR, con una prioridad de sistema operativo de 6.

```
chrt -r 6 java
```

Es posible que necesite configurar su sistema para permitir que se cambien las prioridades. Consulte el apartado “Configuración del sistema para permitir los cambios de prioridades” en la página 21 para obtener más información.

Tabla 2. Prioridades del sistema operativo y Java

Prioridad de Java	Valor de prioridad de Java para la hebra	Prioridad del sistema operativo
1	MIN_PRIORITY	6
2		6
3		6
4		6
5	NORM_PRIORITY (valor predeterminado)	6
6		6
7		6
8		6
9		6
10	MAX_PRIORITY	6

Todas las hebras asociadas a la hebra principal de Java se ejecutan con la misma prioridad de sistema operativo.

Si ejecuta el mandato `chrt -r 11 java`, el resultado es el mismo que si ejecuta `chrt -r 10 java`. Esto se debe a que no puede aplicar una prioridad por encima de 10 a la correlación de prioridades utilizada por las hebras de la JVM, aunque la hebra que inicia la JVM y espera la finalización de la JVM se mantenga con la prioridad 11.

La JVM genera un mensaje de error si se intenta utilizar el mandato `chrt -f 6 java`, porque SCHED_FIFO no se admite en WebSphere Real Time for Linux V3.

Para obtener más información sobre el mandato **chrt**, consulte http://publib.boulder.ibm.com/infocenter/lnxinfo/v3r0m0/index.jsp?topic=/liaai/realtime/liaairtchrt.htm.

Configuración del sistema para permitir los cambios de prioridades

De forma predeterminada, los usuarios no root en Linux no pueden elevar la prioridad de una hebra o un proceso. Puede cambiar la configuración del sistema para permitir los cambios de prioridades usando el módulo `pam_limits` de PAM (Pluggable Authentication Modules) para Linux.

Si no puede cambiar la prioridad de una hebra o un proceso usando el programa de utilidad **chrt**, verá este mensaje:

```
sched_setscheduler: Operation not permitted
```

En kernels Linux más recientes, puede cambiar la configuración del sistema para permitir cambios de prioridades usando el módulo `pam_limits`. Este módulo le permite configurar los límites de los recursos del sistema en el archivo de configuración de límites. El archivo predeterminado es `/etc/security/limits.conf`.

Una entrada en el archivo `/etc/security/limits.conf` tiene el formato siguiente:

```
<dominio> <tipo> <elemento> <valor>
```

donde:

<dominio> puede ser:

- un nombre de usuario del sistema que puede modificar los límites de un recurso.
- un nombre de grupo, con la sintaxis `@group`, cuyos miembros pueden modificar los límites de un recurso.
- un comodín "`*`", que indica que cualquier usuario o grupo puede modificar los límites de un recurso.

<tipo> puede ser:

- `hard`: el kernel impone límites estrictos.
- `soft`: se aplican límites intermedios, que se pueden modificar dentro del rango indicado por los límites estrictos.
- un guión "`-`": indica los límites estrictos e intermedios.

<elemento> es:

- un recurso. Utilice `rtprio` para las prioridades en tiempo real.

<valor> es:

- un límite. Utilice un valor del rango 1 - 100 para indicar el límite máximo para el valor de la prioridad en tiempo real.

Por ejemplo,

```
* - rtprio 100
```

permite a todos los usuarios cambiar la prioridad de los procesos en tiempo real, utilizando **chrt** u otros mecanismos.

De forma predeterminada, el usuario root puede aumentar las prioridades en tiempo real sin límites. Para aplicar un límite a una raíz, el usuario root se debe haber especificado explícitamente. Los límites de grupos y comodines del archivo de configuración no se aplican al usuario root.

Si especifica límites de usuario individuales en el archivo, estos límites tendrán prioridad sobre los límites de grupos.

Los cambios realizados en `limits.conf` no entran en vigor de inmediato. Debe reiniciar los servicios afectados o reiniciar el sistema para que el cambio realizado en la configuración entre en vigor.

La capacidad para aumentar la prioridad en tiempo real de una máquina virtual Java no está disponible en los kernel Linux 2.6.12 y posteriores. La tabla indica si hay soporte disponible para esta característica en algunas de las distribuciones comunes de Linux.

Tabla 3. Soporte para cambios de prioridad en tiempo real

Distribución Linux	Versión de kernel de Linux	Soporte para cambios de prioridad en tiempo real (sí/no)
Red Hat Enterprise Linux (RHEL) 4	2.6.9	no
RHEL 5 y posterior	2.6.18 y posterior	sí
SUSE Linux Enterprise Server (SLES) 9	2.6.5-7	no
SLES 10 y posterior	2.6.16 y posterior	sí
Red Hat Enterprise MRG - todas las versiones	2.6.24 y posterior	sí
SUSE Linux Enterprise Real Time (SLERT) - todas las versiones	2.6.16 y posterior	sí
Ubuntu 5.10	2.6.12	no
Ubuntu 6.06 y posterior	2.6.15 y posterior	sí

Para habilitar los cambios de prioridad en un sistema Linux en tiempo real, puede añadir un usuario al grupo `realtime`, que aparece en el archivo `limits.conf`.

Inicio de procesos secundarios

Los métodos java.lang.Runtime.exec de la API de la máquina virtual Java dan a su aplicación Java la capacidad de ejecutar un mandato en un proceso independiente.

Desde dicha llamada a método, se crea un nuevo objeto java.lang.Process. El objeto se puede usar para controlar el nuevo proceso o para obtener información acerca de él.

Los métodos exec crean varias hebras con esta finalidad. En IBM WebSphere Real Time for Linux, las modificaciones del procedimiento habilitan un comportamiento más determinante en un entorno de tiempo real.

La llamada Runtime.exec crea una hebra "recolectora" para cada subproceso bifurcado. La hebra recolectora es la única que espera a que el subproceso termine. Cuando el subproceso termina, la hebra recolectora registra el estado de salida del subproceso. La hebra recolectora genera el nuevo proceso y le da los mismos de prioridad que a la hebra que llamó en un principio a Runtime.exec.

Si el proceso generado es otra máquina virtual Java de WebSphere Real Time for Linux y el método Runtime.exec ha sido llamado por otro método que ejecuta una prioridad y una política en tiempo real de Linux, la hebra principal de la nueva máquina virtual correlaciona su política y su prioridad con la misma política y prioridad en tiempo real de Linux. Esta prioridad de hebra Java está entre 1 y 10.

La hebra recolectora también crea dos nuevas hebras que escuchan las secuencias `stdout` y `stderr` del nuevo proceso. Los datos `stdout` y `stderr` se guardan en almacenamientos intermedios utilizados por estas hebras. Los almacenamientos intermedios persisten más allá del tiempo de vida del proceso generado. Esta persistencia permite que los recursos que mantiene el proceso generado se borren de inmediato cuando el proceso termina.

Utilización del recopilador de basura Metronome

El recopilador de basura Metronome sustituye al recopilador de basura estándar en WebSphere Real Time for Linux.

Control de los tiempos de pausa

El tiempo de pausa del recopilador de basura (GC) Metronome puede ajustarse para cada proceso Java.

De forma predeterminada, Metronome GC realiza una pausa de 3 milisegundos en cada pausa individual, lo que se denomina cantidad. Un ciclo de recopilación de basura completo requiere muchas de estas pausas, que se dispersan para darle tiempo suficiente a la aplicación para ejecutarse. Puede cambiar este valor de tiempo de pausa individual máximo con la opción **-Xgc:targetPauseTime**. Por ejemplo, con **-Xgc:targetPauseTime=20** el recopilador de basura funciona con pausas individuales que no son superiores a 20 milisegundos.

IBM Monitoring and Diagnostics Tools for Java - Garbage Collection and Memory Visualizer (GCMV) puede utilizarse para supervisar los tiempos de pausa del recopilador de basura de la aplicación, además de ayudar a diagnosticar y ajustar los problemas de rendimiento de la aplicación Java. La herramienta analiza y traza datos desde distintos tipos de registro, entre los que se incluyen:

- Registros cronológicos detallados de la recopilación de basura
- Registros de recopilación de basura de rastreo, generados mediante el parámetro **-Xtgc**.
- Registros de memoria nativa, generados mediante los mandatos del sistema **ps**, **svmon** o **perfmon**.

GCMV genera los gráficos de esta sección y muestra el efecto de cambiar el tiempo de pausa de destino en ciclos de recopilación de basura. Cada gráfico traza los tiempos de pausa actuales entre los ciclos de recopilación de basura Metronome (eje Y) frente al tiempo de ejecución de una aplicación (eje X).

Nota: GCMV tiene soporte para el formato de recogida de basura detallado de versiones anteriores. Si desea analizar la salida detallada de la recogida de basura con GCMV, genere la salida con la opción **-Xgc:verboseFormat=deprecated**. Para obtener más información, consulte Opciones de línea de mandatos de GC.

Con el tiempo de pausa de destino predeterminado establecido, el gráfico de tiempo de pausa de recopilación de basura detallada muestra que los tiempos de pausa se mantienen alrededor de la marca de 3 milisegundos:

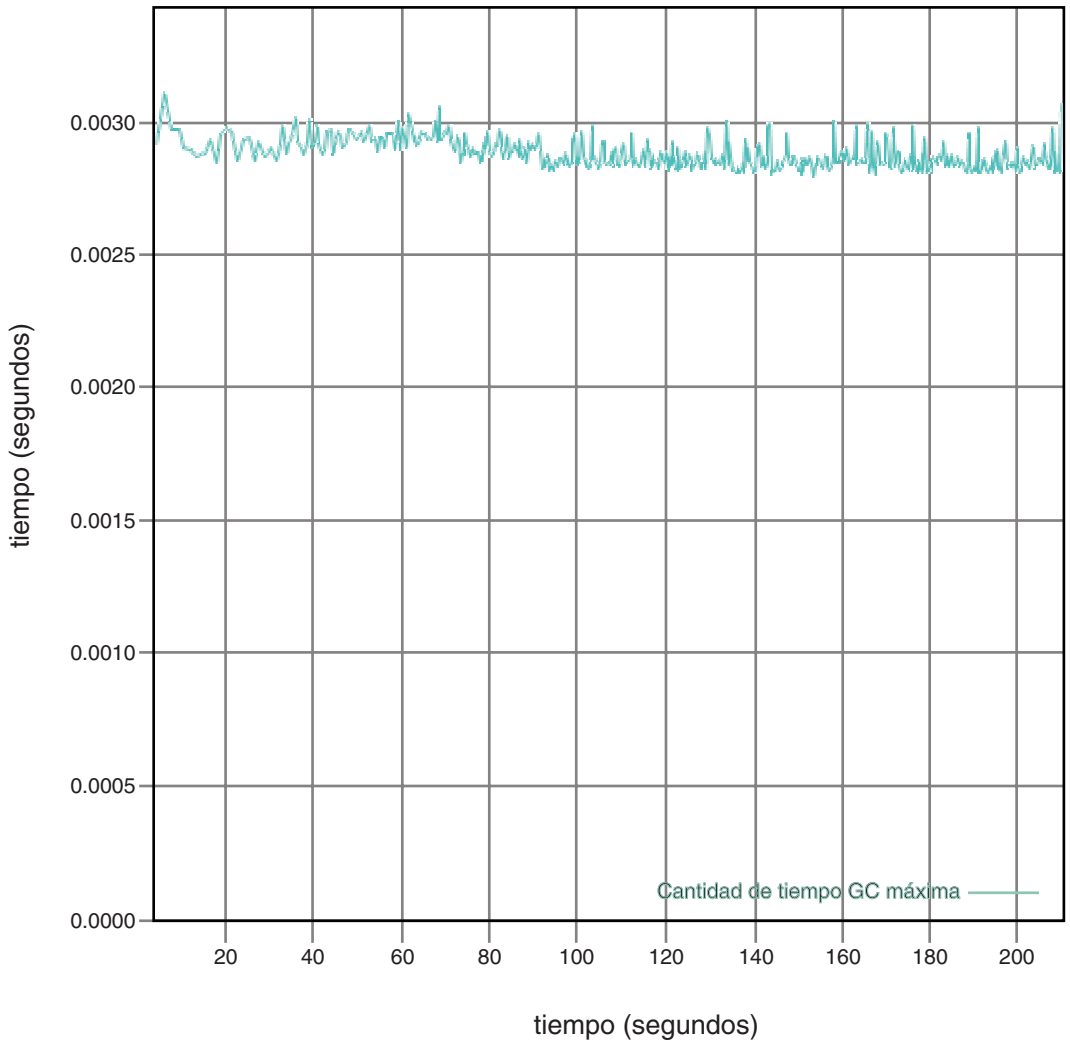

Figura 1. Tiempos de pausa reales de la recopilación de basura cuando el tiempo de pausa de destino se establece en el valor predeterminado (3 milisegundos)

Con un tiempo de pausa establecido en 6 milisegundos, el gráfico de tiempo de pausa del recopilador de basura detallado muestra que los tiempos de pausa se mantienen alrededor de la marca de 6 milisegundos:

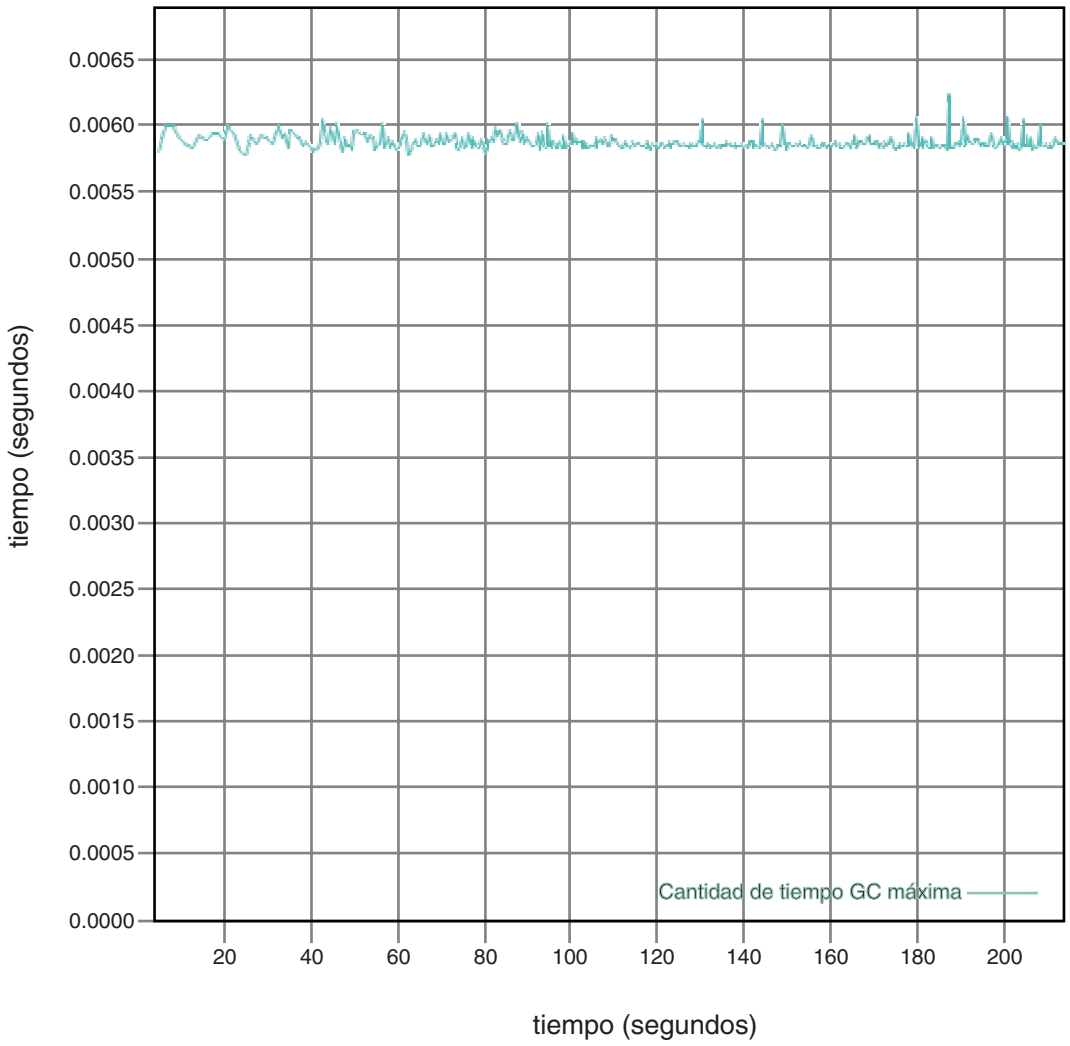

Figura 2. Tiempos de pausa reales cuando el tiempo de pausa de destino se establece en 6 milisegundos

Con un tiempo de pausa establecido en 10 milisegundos, el gráfico de tiempo de pausa del recopilador de basura detallado muestra que los tiempos de pausa se mantienen alrededor de la marca de 10 milisegundos:

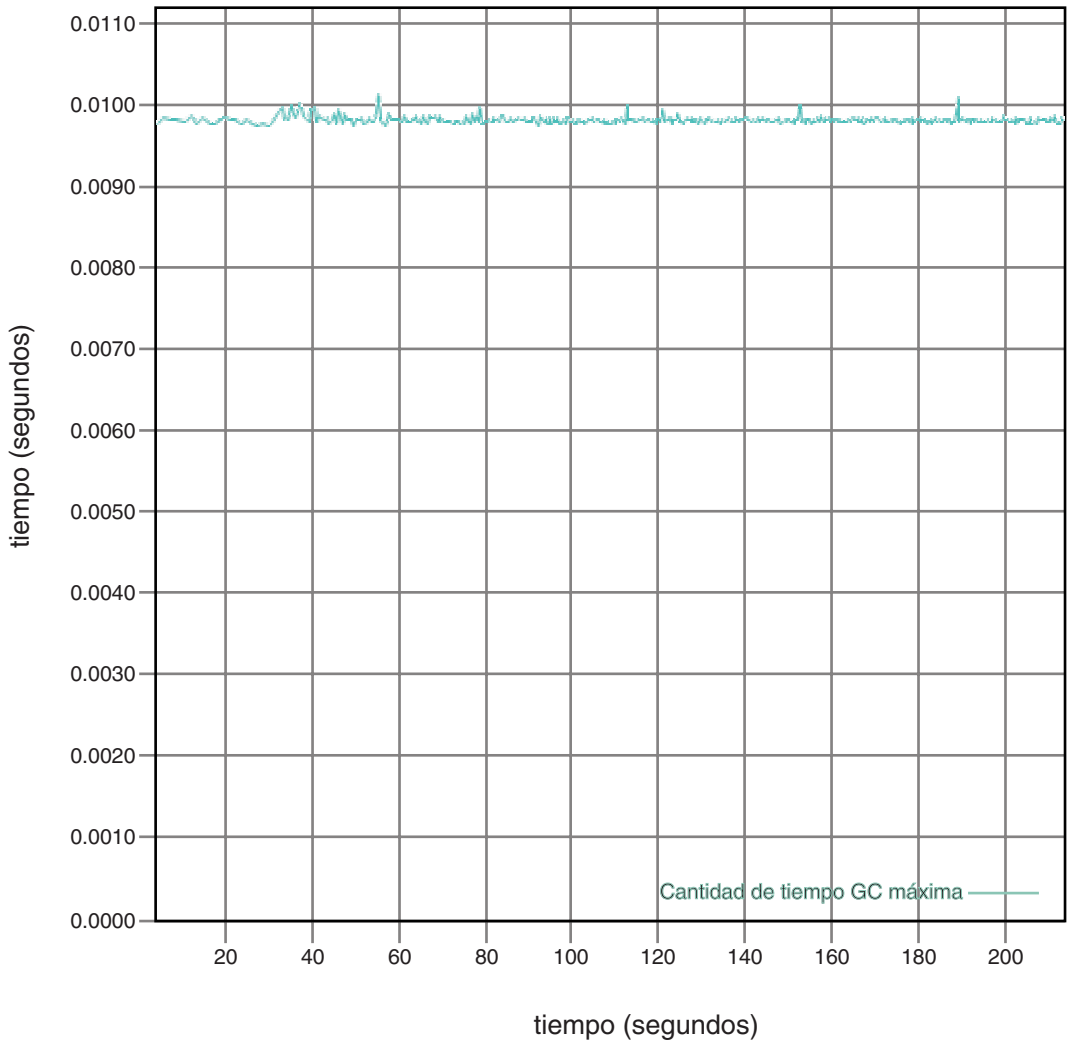

Figura 3. Tiempos de pausa reales cuando el tiempo de pausa de destino se establece en 10 milisegundos

Con un tiempo de pausa establecido en 15 milisegundos, el gráfico de tiempo de pausa del recopilador de basura detallado muestra que los tiempos de pausa se mantienen alrededor de la marca de 15 milisegundos:

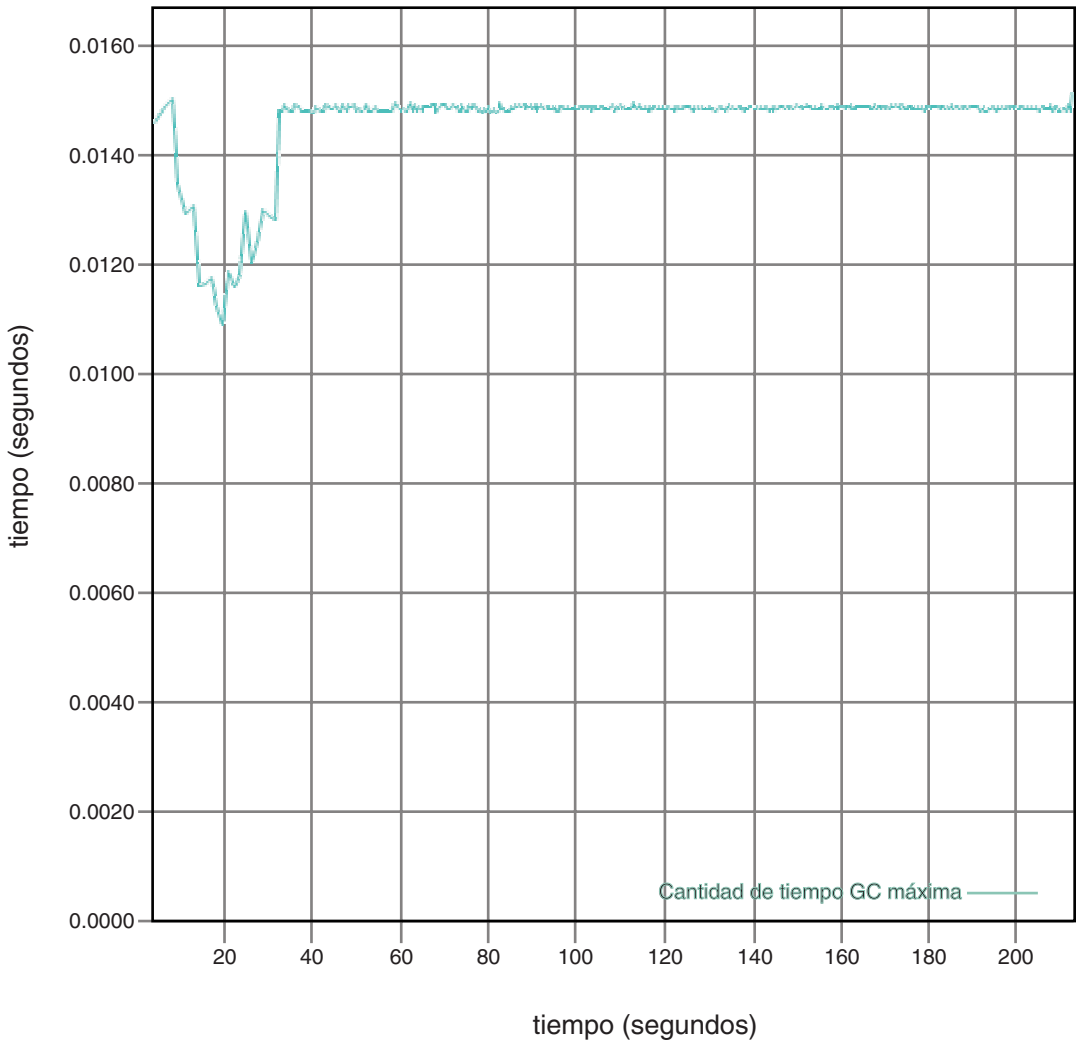

Figura 4. Tiempos de pausa reales cuando el tiempo de pausa de destino se establece en 15 milisegundos

Control de la utilización del procesador

Puede limitar la cantidad de energía de proceso disponible para el recopilador de basura Metronome.

Puede controlar la recogida de basura con el recopilador de basura Metronome utilizando la opción **-Xgc:targetUtilization=N** para limitar la cantidad de CPU utilizada por el recopilador de basura.

Por ejemplo:

```
java -Xgcpolicy:metronome -Xgc:targetUtilization=80 su_aplicación
```

El ejemplo especifica que la aplicación se ejecuta durante el 80% de cada 60 milisegundos. El 20% del tiempo se utiliza para la recogida de basura. El recopilador de basura Metronome garantiza la utilización de los niveles siempre que se le hayan otorgado recursos suficientes. La recogida de basura empieza cuando el espacio libre del almacenamiento dinámico pasa a estar por debajo de un umbral determinado de forma dinámica.

Limitaciones del recopilador de basura Metronome

En este tema se describen las limitaciones o problemas conocidos que afectan a la política de recopilación de basura de Metronome.

Soporte de AESNI en plataformas x86

Actualmente, con la política de recopilación de basura de Metronome, no es posible el aprovechamiento del software de las instrucciones AESNI en arquitecturas de x86.

Largos tiempos de pausa durante la recogida de basura

Bajo excepcionales circunstancias, puede sufrir pausas más largas de lo esperado durante la recopilación de basura.Durante la recogida de basura, se utiliza un proceso de exploración de la raíz. El recopilador de basura recorre el almacenamiento dinámico, empezando por las referencias activas conocidas. Entre estas referencias se incluyen las siguientes:

- Variables de referencias activas en las pilas de llamadas de hebra activas.
- Referencias estáticas.

Para encontrar todas las referencias a objetos activas en la pila de hebras de una aplicación, el recopilador de basura explora todos los marcos de la pila de esta pila de llamadas a la hebra. Cada pila de hebras activas se explora en un paso que no se puede interrumpir. Por tanto, la exploración debe realizarse en una pausa del recopilador de basura individual.

El efecto es que el rendimiento del sistema podría ser peor de lo esperado si tiene algunas hebras con pilas muy profundas, porque la recogida de basura ampliada se detiene al principio de cada ciclo de recogida.

Capítulo 6. Desarrollo de aplicaciones

Información importante sobre la escritura de aplicaciones en tiempo real, incluyendo muestras de código.

Correlación hash en tiempo real de muestra

WebSphere Real Time for Linux incluye implementaciones HashMap y HashSet que proporcionan rendimiento más coherente para el método put que el HashMap estándar en el SDK de IBM para Java 7.

La java.util.HashMap estándar que proporciona IBM funciona correctamente con aplicaciones de alto rendimiento. También ayuda con las aplicaciones que conocen el tamaño máximo hasta el que necesita crecer su correlación hash. Para las aplicaciones que necesiten una correlación hash que pueda crecer hasta distintos tamaños, según el uso, hay un problema potencial de rendimiento con la correlación hash estándar. La correlación hash estándar proporciona buenos tiempos de respuesta para añadir nuevas entradas a la correlación hash utilizando el método put. Sin embargo, cuando se completa la primera correlación hash, se debe asignar un almacenamiento de copia de seguridad más grande. Esto significa que las entradas del almacenamiento de copia de seguridad actual se deben migrar. Si la correlación hash es grande, el período de tiempo necesario para realizar un método put también podría ser grande. Por ejemplo, la operación podría llevar varios milisegundos.

WebSphere Real Time for Linux incluye una correlación hash de tiempo real de muestra. Proporciona la misma interfaz funcional que la java.util.HashMap estándar, pero habilita un rendimiento mucho más sólido para el método put. En lugar de crear un almacenamiento de copia de seguridad y de migrar todas las entradas cuando la correlación hash se completa, la correlación hash de muestra crea un almacenamiento de copia de seguridad adicional. El nuevo almacenamiento de copia de seguridad se encadena a los otros almacenamientos de copia de seguridad en la correlación hash. El encadenamiento, provoca inicialmente una leve reducción en el rendimiento, mientras que el almacenamiento de copia de seguridad vacío se asigna y se encadena a otros almacenamientos de copia de seguridad. Una vez actualizada la correlación hash de copia de seguridad, resulta más rápido que migrar todas las entradas. Una desventaja de la correlación hash en tiempo real es que las operaciones get, put y remove son algo más lentas. Las operaciones son más lentas porque cada búsqueda se debe realizar mediante un conjunto de correlaciones hash de copia de seguridad, en lugar de solo una.

Para probar la correlación hash de tiempo real, añada el archivo `RTHashMap.jar` al inicio de su classpath de arranque. Si ha instalado WebSphere Real Time for Linux en el directorio`$WRT_ROOT`, añada la opción siguiente para utilizar la correlación hash de tiempo real con su aplicación, en lugar de la correlación hash estándar:

```
-Xbootclasspath/p:$WRT_ROOT/demo/realtime/RTHashMap.jar
```

Los archivos de origen y de clase de la implementación de correlación hash de tiempo real están incluidos en el archivo `demo/realtime/RTHashMap.jar`. Además, se proporcionan también implementaciones java.util.LinkedHashMap y java.util.HashSet en tiempo real.

Capítulo 7. Rendimiento

WebSphere Real Time for Linux se ha optimizado para pausas breves en la recogida de basura, en lugar de para el rendimiento más alto en la salida o la ocupación de memoria más baja.

Rendimiento en configuraciones de hardware certificadas

Los sistemas certificados tienen suficiente granularidad de reloj y velocidad de procesador como para admitir los objetivos de rendimiento de WebSphere Real Time for Linux. Por ejemplo, una aplicación bien escrita que se ejecute en un sistema no sobrecargado, y con un tamaño de almacenamiento dinámico adecuado, experimentará normalmente tiempos de pausa por recogida de basura de unos 3 milisegundos, y no más de 3,2 milisegundos. Durante los ciclos de recogida de basura, una aplicación con la configuración de entorno predeterminada no se pausa durante más del 30% del tiempo transcurrido durante cualquier ventana deslizante de 60 milisegundos. El tiempo colectivo empleado en la recogida de basura se pausa por encima de cualquier período de 60 milisegundos suele ser en total de unos 18 milisegundos.

Reducción de la variabilidad de temporización

Los dos principales orígenes de variabilidad en una JVM estándar son las pausas en la recogida de basura. En WebSphere Real Time for Linux, las pausas potencialmente largas de las modalidades estándar del recopilador de basura se pueden evitar utilizando Recopilador de basura Metronome. Consulte el apartado “Utilización del recopilador de basura Metronome” en la página 23.

Compartimiento de datos de clases entre varias JVM

Compartir datos de clases proporciona un método transparente de reducir el espacio de memoria y mejorar el tiempo de inicio de JVM. Para obtener más información sobre el compartimiento de datos de clases, consulte el apartado “Datos de clases compartidos entre las JVM” en la página 32

Referencias comprimidas

Metronome GC soporta las referencias comprimidas y descomprimidas en plataformas de 64 bits. Cuando la máquina virtual Java (JVM) utiliza las referencias comprimidas, almacena todas las referencias a objetos, clases, hebras y supervisores como valores de 32 bits. La utilización de referencias comprimidas mejora el rendimiento de muchas aplicaciones porque los objetos son más pequeños, y como resultado se obtiene una recopilación de basura menos frecuente y una mejor utilización de memoria caché.

Nota: El tamaño de almacenamiento dinámico disponible para referencias comprimidas está limitado a unos 28 GB.
Para obtener más información sobre las referencias comprimidas, consulte http://publib.boulder.ibm.com/infocenter/java7sdk/v7r0/topic/com.ibm.java.lnx.70.doc/diag/understanding/mm_compressed_references.html.

Datos de clases compartidos entre las JVM

El soporte para clases compartidas es el mismo cuando se ejecuta con o sin la opción **-Xrealtime**.

Puede compartir datos de clases entre máquinas virtuales Java (JVM) si los almacena en un archivo en disco de caché correlacionada con memoria. Compartir reduce el consumo de almacenamiento virtual global cuando más de una JVM comparte una memoria caché. También se disminuye el tiempo de arranque de una JVM después de que se ha creado la memoria caché. La memoria caché de clase compartida es independiente de cualquier JVM en ejecución y se mantiene hasta que se suprime.

Una memoria caché compartida puede contener:

- Clases de arranque
- Clases de aplicación
- Metadatos que describen las clases
- Código compilado de manera anticipada (AOT)

Nota: Una JVM no de tiempo real no puede eliminar una memoria caché de clases compartidas en tiempo real.

Capítulo 8. Seguridad

Esta sección contiene información importante sobre seguridad.

Consideraciones de seguridad para la memoria caché de clase compartida

La memoria caché de clase compartida se ha diseñado para facilitar el uso y la gestión de la memoria caché, pero la política de seguridad predeterminada puede no ser la adecuada.

Cuando utilice la memoria caché de clase compartida, debe tener en cuenta los permisos predeterminados de los nuevos archivos, a fin de poder mejorar la seguridad restringiendo el acceso.

Archivo	**Permisos predeterminados**
nuevas memorias caché compartidas	permisos de lectura de grupos y otros
directorio `javasharedresources`	permiso universal de ejecución, escritura y lectura

Se requiere permiso de escritura en el archivo de memoria caché y el directorio de memoria caché, para destruir o desarrollar una memoria caché.

Cambio de los permisos de archivos en el archivo de memoria caché

Para limitar el acceso a una memoria caché de clase compartida, puede utilizar el mandato **`chmod`**.

Cambio necesario	**Mandato**
Limitar acceso al usuario y al grupo	`chmod 770 /tmp/javasharedresources`
Limitar acceso al usuario	`chmod 700 /tmp/javasharedresources`
Limitar el acceso de lectura y escritura del usuario a solo una memoria caché concreta	`chmod 600 /tmp/javasharedresources/<`*`archivo para memoria caché compartida`*`>`
Limitar el acceso de lectura y escritura del usuario y el grupo a solo una memoria caché concreta	`chmod 660 /tmp/javasharedresources/<`*`archivo para memoria caché compartida`*`>`

Cómo conectarse a una memoria caché a la que no tiene permiso para acceder

Si intenta conectarse a una memoria caché para la que no tiene los permisos de acceso adecuados, verá un mensaje de error:

```
JVMSHRC226E Error opening shared class cache file
JVMSHRC220E Port layer error code = -302
JVMSHRC221E Platform error message: Permission denied
JVMJ9VM015W Initialization error for library j9shr25(11): JVMJ9VM009E J9VMDllMain
failed
Could not create the Java virtual machine.
```

Capítulo 9. Resolución de problemas y soporte

Resolución de problemas y soporte para WebSphere Real Time for Linux

Métodos para determinación de problemas generales

La determinación de problemas le ayuda a comprender el tipo de error que tiene y las acciones que debe realizar.

Cuando sepa que tipo de problema tiene, puede realizar una o más de las siguientes tareas:

- Arreglar el problema.
- Buscar un buen método alternativo.
- Recopilar los datos necesarios con los que generar un informe de errores para IBM.

Determinación de problemas en Linux

Este apartado describe la determinación de problemas en Linux.

La Guía de usuario de IBM SDK para Java V7 contiene instrucciones útiles sobre el diagnóstico de problema en Linux, y cubre:

- Configuración y comprobación del entorno de Linux
- Técnicas de depuración generales
- Diagnóstico de bloqueos
- Depuración de cuelgues
- Depuración de fugas de memoria
- Depuración de problemas de rendimiento

Puede encontrar esta información en: IBM SDK para Java 7 - Determinación de problemas en Linux .

La información siguiente es adicional para IBM WebSphere Real Time for Linux

Configuración y comprobación del entorno de Linux

En IBM WebSphere Real Time for Linux, compruebe que la JVM está configurada correctamente para generar un volcado del sistema.

volcados del sistema Linux (archivos clase Core)

Cuando se produce un bloqueo, los datos de diagnóstico más importantes que deben obtenerse son el volcado del sistema de Linux (archivo core). Para asegurarse de que se genera este archivo, debe comprobar los valores del sistema operativo y su espacio de disco disponible, según se muestra en la Guía del usuario de IBM SDK para Java V7.

Valores de la máquina virtual Java

La JVM se debe configurar para generar archivos core cuando se produzca un bloqueo. Ejecute `java -Xdump:what` en la línea de mandatos. La salida de esta opción es:

```
-Xdump:system:
    events=gpf+abort+traceassert+corruptcache,
    label=/mysdk/sdk/jre/bin/core.%Y%m%d.%H%M%S.%pid.dmp,
    range=1..0,
    priority=999,
    request=serial
```

Los valores mostrados son los valores predeterminados. Al menos debe establecerse `events=gpf` para general un archivo clase Core cuando se produzca un bloqueo. Puede cambiar y establecer las opciones con la opción de línea de mandatos **-Xdump:system[:name1=value1,name2=value2 ...]**

Técnicas de depuración generales

Como los nombres de hebra de Java son visibles en el sistema operativo, puede utilizar el mandato ps como ayuda para la depuración. Cuando utilice herramientas de rastreo, debe utilizar los mandatos correctos para IBM WebSphere Real Time for Linux.

Examinar la información sobre procesos

Cuando ejecute el mandato ps en IBM WebSphere Real Time for Linux, la salida será algo parecido a lo siguiente:

```
ps -eLo pid,tid,rtprio,comm,cmd
13654 13654      - java            jre/bin/java -Xgcpolicy:metronome -jar example.jar
13654 13655      - main            jre/bin/java -Xgcpolicy:metronome -jar example.jar
13654 13656      - Signal Reporter jre/bin/java -Xgcpolicy:metronome -jar example.jar
13654 13661      - JIT Compilation jre/bin/java -Xgcpolicy:metronome -jar example.jar
13654 13662      - JIT Sampler     jre/bin/java -Xgcpolicy:metronome -jar example.jar
13654 13666      - Signal Dispatch jre/bin/java -Xgcpolicy:metronome -jar example.jar
13654 13667      - Finalizer maste jre/bin/java -Xgcpolicy:metronome -jar example.jar
13654 13668      - Gc Slave Thread jre/bin/java -Xgcpolicy:metronome -jar example.jar
13654 13669      - Gc Slave Thread jre/bin/java -Xgcpolicy:metronome -jar example.jar
13654 13670      - Gc Slave Thread jre/bin/java -Xgcpolicy:metronome -jar example.jar
13654 13671      - Gc Slave Thread jre/bin/java -Xgcpolicy:metronome -jar example.jar
13654 13672      - Metronome GC Al jre/bin/java -Xgcpolicy:metronome -jar example.jar
13654 13673      - Thread-2        jre/bin/java -Xgcpolicy:metronome -jar example.jar
13654 13698      - process reaper  jre/bin/java -Xgcpolicy:metronome -jar example.jar
13654 13700      - stdout reader j jre/bin/java -Xgcpolicy:metronome -jar example.jar
13654 13701      - stderr reader j jre/bin/java -Xgcpolicy:metronome -jar example.jar
```

e Selecciona todos los procesos.

L Muestra las hebras.

o Proporciona un formato predefinido de las columnas que se deben visualizar. Las columnas especificadas son el ID de proceso, el ID de hebra, la política de planificación, la prioridad de hebra en tiempo real y el mandato asociado al proceso. Esta información es útil para entender qué hebras de la aplicación así como de la máquina virtual se están ejecutando en un momento dado.

Herramientas de rastreo

Tres herramientas de rastreo de Linux son **strace**, **ltrace** y **mtrace**. El mandato `man strace` muestra un conjunto completo de opciones disponibles.

strace
La herramienta strace rastrea las llamadas al sistema. Puede utilizarla en un proceso que ya está disponible, o bien iniciarla con un nuevo proceso. strace registra las llamadas al sistema que realiza un programa y las señales que recibe un proceso. Para cada llamada al sistema, se utiliza el nombre, los argumentos y el valor de retorno. strace le permite rastrear un programa sin requerir el código fuente (no es necesario volver a compilar). Si utiliza strace con la opción **-f**, se rastrearán los procesos hijo que se hayan creado como resultado de una llamada al sistema bifurcada. Puede utilizar strace para investigar problemas de plug-in o intentar entender por qué los programas no se inician correctamente.

Para utilizar strace con una aplicación Java, escriba `strace java -Xgcpolicy:metronome <class-name>`.

Puede dirigir la salida de rastreo desde la herramienta strace a un archivo utilizando la opción `-o`.

ltrace
La herramienta ltrace depende de la distribución. Es muy parecida a strace. Esta herramienta intercepta y registra las llamadas dinámicas a la biblioteca que realiza el proceso en ejecución. strace hace lo mismo para las señales que recibe el proceso en ejecución.

Para utilizar ltrace con una aplicación Java, escriba `ltrace java -Xgcpolicy:metronome <class-name>`

mtrace
mtrace se incluye en el conjunto de herramientas GNU. Instala manejadores especiales para malloc, realloc y free, y permite que todos los usos de estas funciones se rastreen y se registren en un archivo. Este rastreo disminuye la eficiencia del programa y no debe habilitarse durante el uso normal. Para utilizar mtrace, establezca **IBM_MALLOCTRACE** en 1, y establezca **MALLOC_TRACE** para que apunte a un archivo válido donde se almacenará la información de rastreo. Debe tener acceso de escritura a este archivo.

Para utilizar mtrace con una aplicación Java, escriba:

```
export IBM_MALLOCTRACE=1
export MALLOC_TRACE=/tmp/file
java -Xgcpolicy:metronome <class-name>
mtrace /tmp/file
```

Diagnóstico de bloqueos

Cuando recopile información sobre la ejecución de procesos y el entorno de Java antes de un bloqueo, siga estas directrices.

Recopilación de información sobre procesos

Cuando investigue qué ha pasado antes de que se produjera el bloqueo, use los mandatos **gdb** y **bt** para mostrar el seguimiento de la pila de la hebra errónea, en vez de analizar el archivo principal (core).

Obtención de información sobre el entorno Java

Utilice el volcado Java para determinar qué estaba haciendo cada hebra y qué métodos Java se estaban ejecutando. Relacione las direcciones de función con las direcciones de biblioteca para determinar el origen del código que se ejecuta en diversos puntos.

Use la opción **-verbose:gc** para ver el estado del almacenamiento dinámico de Java . Hágase estas preguntas:

- ¿Ha habido una falta de memoria en una de las áreas de memoria que haya podido causar el bloqueo?
- ¿Se ha producido el bloqueo durante la recopilación de basura, lo que apuntaría a un posible error en la recopilación de basura?
- ¿Se ha producido el bloqueo después de la recopilación de basura, lo que indicaría posibles daños en la memoria?

Depuración de problemas de rendimiento

Cuando depure problemas de rendimiento, tenga en cuenta estos elementos específicos para IBM WebSphere Real Time for Linux además de los temas en la Guía del usuario de IBM SDK para Java V7.

Asignación de tamaño de áreas de memoria

El tamaño del almacenamiento dinámico de Java es uno de los parámetros de ajuste más importantes de la JVM. Elija el tamaño adecuado para optimizar el rendimiento. Al utilizar el tamaño adecuado el colector de basura puede proporcionar más fácilmente la utilización necesaria.

Para obtener más información sobre cómo cambiar el tamaño de las áreas de memoria, consulte “Resolución de problemas del recopilador de basura Metronome” en la página 66.

Compilación JIT y rendimiento

Al usar el JIT, debe considerar las implicaciones para el comportamiento en tiempo real.

Limitaciones conocidas en Linux

Linux ha tenido un desarrollo rápido y se han observado varios problemas con la interacción de la JVM y el sistema operativo, especialmente en el área de las hebras.

Tenga en cuenta las siguientes limitaciones que podrían afectar a su sistema Linux.

Hebras como procesos

Si el número de hebras Java sobrepasa el número máximo de procesos permitidos, el programa podría:

- Obtener un mensaje de error
- Obtener un error **SIGSEGV**
- Detenerse

Para obtener más información, consulte *El informe Volano* en http://www.volano.com/report/index.html.

Limitaciones de las pilas flotantes

Si está ejecutando sin pilas flotantes, independientemente de lo que se establezca para **-Xss**, se proporciona un tamaño mínimo de pilas nativas de 256 KB para cada hebra.

En un sistema Linux de pilas flotantes, se utilizan los valores **-Xss**. Si realiza la migración desde un sistema Linux sin pilas flotantes, asegúrese de que los valores **-Xss** son suficientemente grandes y que no se basan en el mínimo de 256 KB.

limitaciones de glibc

Si recibe un mensaje que indica que la biblioteca `libjava.so` no se ha podido cargar porque no se ha encontrado un símbolo (como por ejemplo __bzero), es posible que tenga instalada una versión anterior de la biblioteca GNU C Runtime Library, glibc. El SDK para la implementación de la hebra de Linux requiere glibc versión 2.3.2 o superior.

Limitaciones de font

Cuando se instala en un sistema Red Hat, para permitir que el servidor encuentre los fonts TrueType de Java, ejecute (en Linux IA32, por ejemplo):

```
/usr/sbin/chkfontpath --add /opt/IBM/javawrt3[_64]/jre/lib/fonts
```

Debe hacerlo durante la instalación y debe haber iniciado sesión como usuario "root" para ejecutar el mandato. Para ver problemas de font más detallados, consulte el *SDK Linux y la Guía del usuario del entorno de tiempo de ejecución*..

El planificador Linux CFS (Completely Fair Scheduler) tiene un impacto sobre el rendimiento de Java

Las aplicaciones Java que utilizan exhaustivamente la sincronización pueden tener un rendimiento bajo en las distribuciones Linux que incluyen el planificador CFS (Completely Fair Scheduler). El planificador CSF (Completely Fair Scheduler) se adoptó en el kernel de gama principal de Linux a partir del release 2.6.23. El algoritmo CFS es diferente de los algoritmos de planificación de los releases anteriores de Linux. Podría cambiar las propiedades de rendimiento de algunas aplicaciones. En especial, CFS implementa sched_yield() de forma diferente, lo cual hace que sea más probable que a una hebra resultantes se le asigne tiempo de CPU.

Si tiene este problema, tal vez detecte un uso elevado de CPU por parte de la aplicación Java y un progreso lento a través de los bloques sincronizados. Podría parecer que la aplicación se detiene debido al lento progreso.

Hay dos posibles soluciones alternativas:

- Iniciar la JVM con el argumento adicional **-Xthr:minimizeUserCPU**.
- Configurar el kernel de Linux para que utilice una implementación de sched_yield() que sea más compatible con las versiones anteriores. Para ello, establezca la propiedad de kernel ajustable de sched_compat_yield en **1**. Por ejemplo:

  ```
  echo "1" > /proc/sys/kernel/sched_compat_yield
  ```

No utilice estas soluciones alternativas a menos que detecte que el rendimiento es bajo.

Este problema podría afectar a IBM Developer Kit and Runtime Environment para Linux 5.0 (todas las versiones) y 6.0 (todas las versiones hasta SR 4 inclusive) que se ejecutan en kernels de Linux que incluyen el planificador CFS (Completely Fair Scheduler). Para IBM Developer Kit and Runtime Environment para Linux versión 6.0 posterior a SR 4, se detecta el uso de CFS en el kernel y la opción

-Xthr:minimizeUserCPU se habilita automáticamente. Algunas distribuciones Linux que incluyen el planificador CFS (Completely Fair Scheduler) son Ubuntu 8.04 y SUSE Linux Enterprise Server 11.

Puede obtener más información sobre el planificador CFS en Multiprocessing with the Completely Fair Scheduler.

Problemas de rendimiento en los kernels de Linux Red Hat MRG

Un problema de configuración con los kernels de Red Hat MRG puede producir pausas inesperadas en la hebras de aplicación cuando WebSphere Real Time se inicia con la recopilación de basura detallada habilitada. Estas pausas no se notifican en la salida de la recopilación de basura (GC) detallada, pero pueden durar varios milisegundos, en función de la configuración de red. Las JVM iniciadas desde usuarios LDAP definidos remotamente son las que se ven más afectadas, ya que el daemon de la memoria caché de servicio de nombres (`nscd`) no inicia y produce retardos en la red. Solucione el problema iniciando `nscd`. Siga los pasos que se indican a continuación para comprobar el estado del servicio `nscd` y corrija el problema:

1. Para comprobar que el daemon `nscd` se está ejecutando, escriba el siguiente mandato:

   ```
   /sbin/service nscd status
   ```

   Si el daemon no se está ejecutando, verá el siguiente mensaje:

   ```
   nscd is stopped
   ```

2. Como usuario root, inicie el servicio `nscd` con el siguiente mandato:

   ```
   /sbin/service nscd start
   ```

3. Como usuario root, cambie la información de inicio para el servicio `nscd` con el siguiente mandato:

   ```
   /sbin/chkconfig nscd on
   ```

Ahora el proceso `nscd` se está ejecutando y se inicia automáticamente después de rearrancar.

Determinación de problemas de NLS

La JVM contiene soporte integrado para diferentes entornos locales.

La Guía de usuario de IBM SDK para Java V7 contiene instrucciones útiles sobre el diagnóstico de problema con NLS, y cubre:

- Visión general de fonts
- Programas de utilidad de fonts
- Problemas comunes con NLS y causas posibles

Esta información está en: IBM SDK para Java 7 - Determinación de problemas de NLS.

Determinación de problemas de ORB

Una de las primeras tareas al depurar un problema de ORB es determinar si el problema está en el lado del cliente o en el lado del servidor de la aplicación distribuida. Piense en una sesión de RMI-IIOP normal como una sencilla comunicación síncrona entre un cliente que solicita acceso a un objeto y un servidor que lo suministra.

La Guía de usuario de IBM SDK para Java V7 contiene instrucciones útiles sobre el diagnóstico de problema con ORB, y cubre:

- Identificación de un problema de ORB
- Interpretación del rastreo de la pila
- Interpretación de rastreos de ORB
- Problemas comunes
- Servicio ORB de IBM: recopilación de datos

Esta información está en: IBM SDK para Java 7 - Determinación de problemas de ORB.

La información siguiente es adicional para IBM WebSphere Real Time for Linux.

Servicio ORB de IBM: recopilación de datos

Cuando recopile la salida de la versión de Java para el servicio, ejecute el mandato siguiente:

```
java -Xgcpolicy:metronome -version
```

Pruebas preliminares

Cuando se produzca un problema, el ORB podría generar una excepción `org.omg.CORBA.*` que incluya:

- texto para indicar la causa
- un código menor
- Un estado de terminación

Antes de dar por supuesto que el ORB es la causa del problema, compruebe estos elementos:

- El escenario se puede reproducir en una configuración similar.
- JIT está inhabilitado.
- NO se está utilizando ningún código compilado por AOT

Otras acciones serían:

- Apagar los procesadores adicionales.
- Apagar la multihebra simultánea (SMT) cuando sea posible.
- Elimine dependencias de memoria con el cliente o el servidor. La falta de memoria física puede ser la causa de un rendimiento lento, de bloqueos o cuelgues aparentes. Para eliminar estos problemas, asegúrese de disponer de suficiente espacio de memoria.
- Compruebe los problemas de red física, como cortafuegos, enlaces de comunicaciones, direccionadores y servidores de nombres DNS. Éstas son las causas principales de las excepciones de COMM_FAILURE de CORBA. Como prueba, haga ping en el nombre de su propia estación de trabajo.
- Si la aplicación utiliza una base de datos como, por ejemplo, DB2, cambie al controlador más fiable. Por ejemplo, para aislar DB2 AppDriver, cambie a Net Driver, que es más lento y utiliza sockets, pero es más fiable.

Resolución de problemas de tipo OutOfMemory

Tratamiento de excepciones OutOfMemoryError.

Para información de resolución de problemas general sobre el uso de la Recogida de basura metronome, consulte “Resolución de problemas del recopilador de basura Metronome” en la página 66.

Diagnóstico de excepciones OutOfMemoryError

Diagnosticar excepciones de tipo OutOfMemoryError en el recopilador de basura Metronome puede resultar más complejo que en una JVM estándar, debido a la naturaleza periódica del recopilador de basura.

Por lo general, una aplicación de Real Time requiere aproximadamente un 20% más de espacio de almacenamiento dinámico que una aplicación Java estándar.

De forma predeterminada, la JVM produce la siguiente salida de diagnóstico cuando se genera una excepción de tipo OutOfMemoryError no detectada:

- Un volcado breve; consulte el apartado “Uso de agentes de volcado” en la página 47.
- Un volcado de almacenamiento dinámico; consulte el apartado “Utilizar el vuelco de almacenamiento dinámico” en la página 55.
- Un volcado Java; consulte el apartado “Uso del volcado Java” en la página 50
- Un volcado del sistema; consulte “Uso de volcados del sistema y el visor de volcados” en la página 58.

Los nombres de los archivos de volcado se otorgan en la salida de la consola:

```
JVMDUMP006I Processing dump event "systhrow", detail "java/lang/OutOfMemoryError" - please wait.
JVMDUMP007I JVM Requesting Snap dump using 'Snap.20081017.104217.13161.0001.trc'
JVMDUMP010I Snap dump written to Snap.20081017.104217.13161.0001.trc
JVMDUMP007I JVM Requesting Heap dump using 'heapdump.20081017.104217.13161.0002.phd'
JVMDUMP010I Heap dump written to heapdump.20081017.104217.13161.0002.phd
JVMDUMP007I JVM Requesting Java dump using 'javacore.20081017.104217.13161.0003.txt'
JVMDUMP010I Java dump written to javacore.20081017.104217.13161.0003.txt
JVMDUMP013I Processed dump event "systhrow", detail "java/lang/OutOfMemoryError".
```

El rastreo inverso de Java, mostrado en la salida de la consola, también disponible en el volcado Java, indica en qué parte de la aplicación Java se ha producido la excepción OutOfMemoryError. El componente de gestión de memoria de la JVM emite un punto de rastreo con el tamaño, la dirección de bloque de clases y el nombre del espacio de memoria de la asignación que falla. Encontrará este punto de rastreo en el volcado breve:

```
<< lines omitted... >>
09:42:17.563258000 *0xf2888e00       j9mm.101  Event       J9AllocateIndexableObject() returning NULL! 80
bytes requested for object of class 0xf1632d80 from memory space 'Metronome' id=0xf288b584
```

Los campos de datos e ID de punto de rastreo pueden variar con respecto a los mostrados, en función del tipo de objeto que se esté asignando. En este ejemplo, el punto de rastreo muestra que el error en la asignación se ha producido cuando la asignación trató de asignar un objeto de 33,6 MB de tipo `class 0x81312d8` en `Metronome heap`, en el segmento de memoria `id=0x809c5f0`.

Puede determinar qué área de memoria de está afectada observando la información de gestión de memoria del volcado Java:

```
NULL           ------------------------------------------------------------
0SECTION       MEMINFO subcomponent dump routine
NULL           ================================
NULL
```

```
1STMEMTYPE     Object Memory
NULL           region     start      end        size       name
1STHEAP        0xF288B584 0xF2A1C000 0xF6A1C000 0x04000000 Default
NULL
1STMEMUSAGE    Total memory available: 67108864 (0x04000000)
1STMEMUSAGE    Total memory in use:    66676824 (0x03F96858)
1STMEMUSAGE    Total memory free:      00432040 (0x000697A8)

<< lines removed for clarity >>
```

Puede determinar el tipo de objeto que se está asignando examinando la sección de clases del volcado Java:

```
NULL           ------------------------------------------------------------------------
0SECTION       CLASSES subcomponent dump routine
NULL           ================================
<< lines omitted... >>
1CLTEXTCLLOD    ClassLoader loaded classes
2CLTEXTCLLOAD    Loader *System*(0xF182BB80)
<< lines omitted... >>
3CLTEXTCLASS      [C(0xF1632D80)
```

La información del volcado Java confirma que se ha intentado una asignación para una matriz de caracteres, en el almacenamiento dinámico normal (`ID=0xF288B584`) y que el tamaño total asignado del almacenamiento dinámico, indicado por la línea `1STHEAP` correspondiente, es de 67108864 bytes decimales o 0x04000000 bytes hexadecimales, o bien 64 MB.

En este ejemplo, la asignación que falla es más grande en relación con el tamaño total del almacenamiento dinámico. Si se espera que su aplicación cree objetos de 33 MB, el paso siguiente será aumentar el tamaño del almacenamiento dinámico mediante la opción **-Xmx**.

Es más habitual que la asignación que falla sea pequeña en relación con el tamaño total del almacenamiento dinámico. Esto se debe a que las asignaciones anteriores llenan el almacenamiento dinámico. En estos casos, el paso siguiente pasa por utilizar el volcado de almacenamiento dinámico para investigar la memoria asignada a los objetos existentes.

El volcado de almacenamiento dinámico es un archivo binario comprimido que contiene una lista con todos los objetos con su clase de objeto, su tamaño y sus referencias. Analice el volcado de almacenamiento dinámico utilizando la herramienta IBM Monitoring and Diagnostic Tools for Java - Memory Analyzer, disponible para su descarga desde IBM Support Assistant (ISA).

Mediante MDD4J, puede cargar un volcado de almacenamiento dinámico y buscar tres estructuras de objetos sospechosos de consumir una gran cantidad de espacio en el almacenamiento dinámico. La herramienta proporciona varias vistas para los objetos del almacenamiento dinámico. Por ejemplo, MDD4J puede mostrar una vista que detalla los sospechosos probables de las fugas de memoria, además de mostrarse los cinco principales objetos y paquetes que contribuyen al tamaño del almacenamiento dinámico. Al seleccionar la vista de árbol, obtendrá más información sobre la naturaleza del objeto contenedor con la fuga de memoria.

Cómo gestiona la memoria la JVM de IBM

La JVM de IBM requiere memoria para varios componentes distintos, incluidas regiones de memoria para clases, código compilado, objetos Java, pilas Java y pilas

JNI. Algunas de estas regiones de memoria deben estar en la memoria contigua. Otras regiones de memoria se pueden segmentar en regiones de memoria más pequeñas y enlazarse juntas.

Las clases cargadas de forma dinámica y el código compilado se almacenan en regiones de memoria segmentadas para clases cargadas dinámicamente. Las clases se subdividen a continuación en regiones de memoria de escritura (clases RAM) y regiones de memoria de solo lectura (clases ROM). Durante el tiempo de ejecución, las clases ROM y el código AOT de la memoria caché de clase se correlacionan mediante la memoria, pero no se cargan, en una región de memoria del inicio de la aplicación. Dado que la aplicación hace referencia a las clases, estas clases y el código compilado en la memoria caché de clase se correlacionan en el almacenamiento. Varios procesos que hacen referencia a esta clase comparten el componente ROM de la clase. La primera vez que la JVM hace referencia a la clase, se crea el componente RAM de la clase en las regiones de memoria segmentadas para las clases cargadas dinámicamente. El código compilado mediante AOT para los métodos de una clase de la memoria caché de clase se copian en una región de memoria de código dinámico ejecutable, porque los procesos no comparten este código. Las clases que no se cargan desde la memoria caché de clase son similares a las clases almacenadas en memoria caché, salvo que la información de la clase ROM se crea en regiones de memoria segmentadas para las clases cargadas de forma dinámica. El código generado de forma dinámica se almacena en las mismas regiones de memoria de código dinámico que contienen el código AOT para clases almacenadas en memoria caché.

La pila para cada hebra de Java puede abarcar una región de memoria segmentada. La pila JNI de cada hebra ocupa una región de memoria contigua.

Para determinar la configuración de la JVM, ejecútela con la opción **-verbose:sizes**. Esta opción imprime información sobre las regiones de memoria en las que puede gestionar el tamaño. Para las regiones de memoria que no son contiguas, se imprime un incremento que describe cuánta memoria se obtiene cada vez que la región necesita crecer.

Esta es una salida de ejemplo que utiliza las opciones **-Xrealtime -verbose:sizes**:

```
-Xmca32K                    RAM class segment increment
-Xmco128K                   ROM class segment increment
-Xms64M                     initial memory size

-Xmx64M                     memory maximum
-Xmso256K                   operating system thread stack size
-Xiss2K                     java thread stack initial size
-Xssi16K                    java thread stack increment
-Xss256K                    java thread stack maximum size
```

En este ejemplo se indica que el segmento de la clase RAM es inicialmente 0, pero crece en bloques de 32 KB cuando es necesario. El segmento de la clase ROM es inicialmente 0, y crece en bloques de 128 KB cuando es necesario. Puede utilizar las opciones **-Xmca** y **-Xmco** para controlar estos tamaños. Los segmentos de la clase RAM y la clase ROM crecen cuando es necesario, por lo que normalmente no será necesario que cambie estas opciones.

Utilice la opción **-Xshareclasses** para determinar el tamaño que tendrá su región de memoria correlacionada si utiliza la memoria caché de clase. Esta es una muestra de la salida del mandato `java -Xgcpolicy:metronome -Xshareclasses:printStats`.

```
Current statistics for cache "sharedcc_chamlain":


base address = 0xF1BBD000
end address = 0xF2BAF000
allocation pointer = 0xF1CA95A0

cache size = 16776852
free bytes = 15499564
ROMClass bytes = 1198572
AOT bytes = 0
Data bytes = 57300
Metadata bytes = 21416
Metadata % used  = 1%

# ROMClasses = 368
# AOT Methods = 0
# Classpaths                          = 1
# URLs           = 0
# Tokens         = 0
# Stale classes = 0
% Stale classes  = 0%

Cache is 7% full
```

Durante el tiempo de ejecución, se copian unos 3 MB de bytes AOT y bytes de metadatos en la región segmentada de código dinámico, a medida que se hace referencia a las clases. Los bytes de datos se copian en la región segmentada de la clase RAM, a medida que se hace referencia a las clases.

Utilización de las herramientas de diagnóstico

Hay una serie de herramientas de diagnóstico disponibles que ayudan a diagnosticar problemas con la JVM de IBM WebSphere Real Time for Linux.

El SDK de IBM para Java 7 proporciona una serie de herramientas de diagnóstico que se pueden usar para diagnosticar problemas con la JVM de IBM WebSphere Real Time for Linux. En esta sección se presentan las herramientas que hay disponibles, y se proporcionan enlaces a más información sobre el uso de las herramientas.

Hay un punto importante que recordar cuando se utilizan las herramientas de diagnóstico de SDK. Cuando invoque la JVM en tiempo real, utilice la opción siguiente:

```
java  -Xgcpolicy:metronome
```

Esta opción se debe utilizar cuando se ejecutan herramientas de diagnóstico para la JVM en tiempo real. Por ejemplo, para mostrar los agentes de volcado registrados para la JVM de IBM WebSphere Real Time for Linux, escriba:

```
java -Xgcpolicy:metronome -Xdump:what
```

El resto de diferencias en el uso de estas herramientas con IBM WebSphere Real Time for Linux se proporciona aquí como información suplementaria, junto con salida de ejemplo, como ayuda al diagnóstico.

Para un resumen de la información de diagnóstico generada por el SDK de IBM para Java 7, consulte Resumen de información de diagnóstico.

Uso de IBM Monitoring and Diagnostic Tools for Java

IBM proporciona herramientas y documentación como ayuda para comprender, supervisar y diagnosticar problemas con aplicaciones que utilizan IBM JRE.

Están disponibles las herramientas siguientes:

- Health Center
- Garbage Collection and Memory Visualizer
- Interactive Diagnostic Data Explorer
- Memory Analyzer

Garbage Collection and Memory Visualizer

Garbage Collection and Memory Visualizer (GCMV) le ayuda a conocer el uso de memoria, el comportamiento de la recogida de basura y el rendimiento de las aplicaciones Java.

GCMV analiza y traza datos desde distintos tipos de registro, entre los que se incluyen:

- Registros cronológicos detallados de la recopilación de basura
- Registros de recopilación de basura de rastreo, generados mediante el parámetro -Xtgc.
- Registros de memoria nativa, generados mediante los mandatos del sistema ps, **svmon** o **perfmon**.

La herramienta ayuda a diagnosticar problemas como fugas de memoria y a analizar los datos en distintos formatos visuales, y proporciona recomendaciones de ajuste.

GCMV se proporciona como un complemento de IBM Support Assistant (ISA). Para obtener información sobre cómo instalar y cómo empezar con el complemento, consulte: http://www.ibm.com/developerworks/java/jdk/tools/gcmv/.

En el IBM Information Center hay disponible más información sobre GCMV.

Health Center

Health Center es una herramienta de diagnóstico para supervisar el estado de una Java Virtual Machine (JVM) en ejecución.

La herramienta contiene dos partes:

- El agente de Health Center que recopila datos de una aplicación en ejecución.
- Un cliente basado en la plataforma Eclipse que se conecta al agente. El cliente interpreta los datos y realiza recomendaciones para mejorar el rendimiento de la aplicación supervisada.

Health Center se proporciona como un complemento de IBM Support Assistant (ISA). Para obtener información sobre cómo instalar y cómo empezar con el complemento, consulte: http://www.ibm.com/developerworks/java/jdk/tools/healthcenter/.

En el IBM Information Center hay disponible más información sobre Health Center.

Interactive Diagnostic Data Explorer

Interactive Diagnostic Data Explorer (IDDE) es una alternativa basada en GUI al visor de volcado (mandato **jdmpview**). IDDE proporciona la misma funcionalidad que el visor de volcado, pero con soporte adicional, como la posibilidad de guardar la salida del mandato.

Utilice IDDE para explorar y examinar más fácilmente archivos de volcado generados por la JVM. En IDDE, se especifican mandatos en un registro de investigación para explorar el archivo de volcado. El soporte que se proporciona con el registro de investigación incluye los elementos siguientes:

- Asistencia de mandatos
- Cumplimentación automática del texto y algunos parámetros como nombres de clase
- La posibilidad de guardar mandatos y salida, que podrá enviar entonces a otras personas.
- Texto resaltado y señalización de problemas
- La posibilidad de añadir sus propios comentarios
- Soporte para utilizar el analizador de memoria desde IDDE

IDDE se proporciona como un complemento de IBM Support Assistant (ISA). Para obtener más información sobre cómo instalar el complemento e iniciarse en él, consulte la visión general de IDDE en developerWorks.

Hay disponible más información sobre IDDE en el Information Center de IBM.

Memory Analyzer

Memory Analyzer le ayuda a analizar el almacenamiento dinámico de Java mediante volcados de almacenamiento dinámico portátil (PHD, Portable Heap Dump) y volcados de nivel de sistema operativo.

Esta herramienta puede analizar volcados que contienen millones de objetos y proporciona la siguiente información:

- Los tamaños retenidos de los objetos.
- Los procesos que están impidiendo que el recopilador de basura recopile los objetos.
- Un informe para extraer automáticamente los sucesos que pueden estar causando fugas de memoria.

Esta herramienta se basa en el proyecto Eclipse Memory Analyzer (MAT) y utiliza la característica IBM Diagnostic Tool Framework for Java (DTFJ) para permitir el proceso de volcados de las JVM de IBM.

Memory Analyzer se proporciona como un complemento de IBM Support Assistant (ISA). Para obtener información sobre cómo instalar y cómo empezar con el complemento, consulte: http://www.ibm.com/developerworks/java/jdk/tools/memoryanalyzer/.

En el IBM Information Center hay disponible más información sobre Memory Analyzer.

Uso de agentes de volcado

Los agentes de volcado se configuran durante la inicialización de la JVM. Le permiten utilizar sucesos que se producen en la JVM como, por ejemplo, la

recopilación de basura, el inicio de hebras o la terminación de la JVM, para iniciar volcados o iniciar una herramienta externa.

La Guía de usuario de IBM SDK para Java V7 contiene instrucciones útiles sobre agentes de volcado, y cubre:

- Uso de la opción **-Xdump**
- Agentes de volcado
- Sucesos de volcado
- Control avanzado de agentes de volcado
- Símbolos de agente de volcado
- Agentes de volcado predeterminados
- Eliminación de agentes de volcado
- Variables de entorno de agente de volcado
- Correlaciones de señales
- Ubicaciones predeterminadas del agente de volcado

Puede encontrar esta información aquí: IBM SDK for Java 7 - Using dump agents.

Aquí se proporciona información adicional para IBM WebSphere Real Time for Linux:

Sucesos de volcado

Los agentes de volcado son desencadenados por sucesos que se producen durante el funcionamiento de la JVM. Para IBM WebSphere Real Time for Linux, el valor predeterminado del suceso lento es de 5 milisegundos.

Algunos sucesos se pueden filtrar para mejorar la relevancia de la salida. Consulte “Opción filter” en la página 49 para obtener más información.

Nota: Los sucesos unload y expand actualmente no se producen en WebSphere Real Time. Las clases están en memoria inmortal y no se pueden descargar.

Nota: Los sucesos gpf y abort no pueden desencadenar un volcado de almacenamiento dinámico, prepare el almacenamiento dinámico (request=prepwalk), o compáctelo (request=compact).

La siguiente tabla muestra sucesos disponibles cuando el agente de volcado se desencadena:

Suceso	Desencadenado cuando...	Operación de filtro
gpf	Se produce un error de protección general.	
user	La JVM recibe la señal SIGQUIT del sistema operativo.	
abort	La JVM recibe la señal SIGABRT del sistema operativo.	
vmstart	La máquina virtual se inicia.	
vmstop	La máquina virtual se detiene.	Filtra código de salida; por ejemplo, **filter=#129..#192#-42#255**
load	Se carga una clase.	Filtra nombre de clase; por ejemplo, **filter=java/lang/String**
unload	Se descarga una clase.	

Suceso	Desencadenado cuando...	Operación de filtro
throw	Se emite una excepción.	Filtra nombre de clase de excepción; por ejemplo, **filter=java/lang/OutOfMem***
catch	Se detecta una excepción.	Filtra nombre de clase de excepción; por ejemplo, **filter=*Memory***
uncaught	La aplicación no detecta una excepción de Java.	Filtra nombre de clase de excepción; por ejemplo, **filter=*MemoryError**
systhrow	La JVM está a punto de emitir una excepción de Java. Es diferente del suceso "throw" porque solo se desencadena para condiciones de error detectadas internamente en la JVM.	Filtra nombre de clase de excepción; por ejemplo, **filter=java/lang/OutOfMem***
thrstart	Se inicia una nueva hebra.	
blocked	Se bloquea una hebra.	
thrstop	Se detiene una hebra.	
fullgc	Se inicia un ciclo de recopilación de basura.	
slow	Una hebra tarda más de 5 ms en responder a una solicitud interna de la JVM.	Cambia el tiempo que tarda un suceso en considerarse lento; por ejemplo, **filter=#300ms** se desencadenará cuando una hebra tarde más de 300 ms en responder a una solicitud interna de la JVM.
allocation	Un objeto Java se asigna con un tamaño que coincide con una especificación de filtro determinada.	Filtra tamaño de objeto; debe suministrarse un filtro. Por ejemplo, se desencadenará **filter=#5m** en objetos superiores a 5 Mb. También se da soporte a rangos; por ejemplo, se desencadenará **filter=#256k..512k** en objetos entre 256 Kb y 512 Kb en tamaño.
traceassert	Se ha producido un error interno en la JVM	No aplicable.
corruptcache	La JVM encuentra que la memoria caché de clases compartidas está dañada.	No aplicable.

Opción filter

Algunos sucesos de la JVM se producen miles de veces durante el tiempo de vida de una aplicación. Los agentes de volcado pueden utilizar filtros y rangos para evitar que se produzcan excesivos volcados.

Comodines

Puede utilizar un comodín en su filtro de sucesos de excepción colocando un asterisco sólo al principio o al final del filtro. El siguiente mandato no funciona porque el segundo asterisco no está al final:

```
-Xdump:java:events=vmstop,filter=*InvalidArgumentException#*.myVirtualMethod
```

Para que funcione este filtro, debe cambiarlo a:

```
-Xdump:java:events=vmstop,filter=*InvalidArgumentException#MyApplication.*
```

Sucesos de carga de clases y excepción

Puede filtrar los sucesos de carga de clase (load) y excepción (throw, catch, uncaught, systhrow) por nombre de clase Java:

```
-Xdump:java:events=throw,filter=java/lang/OutOfMem*
-Xdump:java:events=throw,filter=*MemoryError
-Xdump:java:events=throw,filter=*Memory*
```

Puede filtrar los sucesos de excepción throw, uncaught y systhrow por nombre de método Java:

```
-Xdump:java:events=throw,filter=ExceptionClassName[#ThrowingClassName.
throwingMethodName[#stackFrameOffset]]
```

Las partes opcionales aparecen entre paréntesis.

Puede filtrar los sucesos de excepción catch por nombre de método Java:

```
-Xdump:java:events=catch,filter=ExceptionClassName[#CatchingClassName.
catchingMethodName]
```

Las partes opcionales aparecen entre corchetes.

Suceso vmstop

Puede filtrar el suceso JVM shut down utilizando uno o varios códigos de salida:

```
-Xdump:java:events=vmstop,filter=#129..192#-42#255
```

Suceso slow

Puede filtrar el suceso slow para cambiar el umbral de tiempo del valor predeterminado de 5 ms:

```
-Xdump:java:events=slow,filter=#300ms
```

No puede establecer el filtro en un tiempo inferior al tiempo predeterminado.

Suceso allocation

Debe filtrar el suceso allocation para especificar el tamaño de los objetos que causan un desencadenante. Puede establecer el tamaño de filtro de cero al máximo valor de un puntero de 32 bits en plataformas de 32 bits, o al valor máximo de un puntero de 64 bits en plataformas de 64 bits. Si establece el valor de filtro inferior en cero desencadenará un volcado en todas las ubicaciones.

Por ejemplo, para desencadenar volcados en asignaciones superiores a 5 Mb de tamaño, utilice:

```
-Xdump:stack:events=allocation,filter=#5m
```

Para desencadenar volcados en asignaciones entre 256 Kb y 512 Kb en tamaño, utilice:

```
-Xdump:stack:events=allocation,filter=#256k..512k
```

Otros sucesos

Si aplica un filtro a un suceso que no soporta filtrado, el filtro se ignorará.

Uso del volcado Java

El volcado Java produce archivos que contienen información de diagnóstico relacionada con la JVM y una aplicación Java capturada en un punto durante la ejecución. Por ejemplo, la información puede ser sobre el sistema operativo, el entorno de aplicación, hebras, pilas, bloqueos y memoria.

La Guía de usuario de IBM SDK para Java V7 contiene instrucciones útiles sobre volcados Java, y cubre:

- Habilitación de un volcado Java

- Cómo se desencadena un volcado Java
- Interpretación de un volcado Java
- Variables de entorno y volcado Java

Puede encontrar dicha información aquí: IBM SDK for Java 7 - Using Javadump.

En los temas siguientes se proporciona información suplementaria y salida de ejemplo para IBM WebSphere Real Time for Linux.

Gestión del almacenamiento (MEMINFO)

La sección `MEMINFO` proporciona información sobre el Gestor de memoria, incluyendo las áreas de memoria de almacenamiento dinámico, inmortal y con ámbito.

La sección `MEMINFO` de un volcado Java proporciona información sobre el gestor de memoria. Consulte Uso del recopilador de basura Metronome para obtener detalles sobre cómo funciona el componente gestor de memoria.

Esta parte del volcados Java proporciona varios valores de gestión de almacenamiento, incluyendo:

- cantidad de memoria libre
- cantidad de memoria usada
- tamaño actual del almacenamiento dinámico
- tamaño actual de las áreas de memoria inmortal
- tamaño actual de las áreas de memoria con ámbito

Esta sección también contiene datos históricos sobre la recogida de basura. Los datos aparecen como secuencia de puntos de rastreo, cada uno con indicación de fecha y hora, ordenados con el punto de rastreo más reciente primero.

Los volcados Java producidos por la JVM estándar contienen una sección "GC History". Esta información no está contenida en los volcados Java producidos cuando se utiliza la JVM de tiempo real. Utilice la opción **`-verbose:gc`** o el rastreo breve de la JVM para obtener información sobre el comportamiento de la recopilación de basura (GC). Para obtener más detalles, consulte "Utilización de la información de verbose:gc" en la página 66 y la sección de agentes de volcado de la Guía de usuario de IBM SDK para Java V7.

En un volcado Java, los segmentos son bloques de memoria asignados por el tiempo de ejecución de Java para que las tareas utilicen más cantidad de memoria. Estas son algunas tareas de ejemplo:

- mantenimiento de memorias caché JIT
- almacenamiento de clases Java

El entorno de tiempo de ejecución Java también asigna otra memoria nativa que no aparece en la lista de la sección `MEMINFO`. El total de la memoria utilizada por los segmentos de Java Runtime no representa necesariamente toda la ocupación de memoria de Java Runtime. Un segmento de Java Runtime consiste en la estructura de datos del segmento y un bloque de memoria nativa asociado.

El siguiente ejemplo muestra una salida normal. Todos los valores se proporcionan como valores hexadecimales. Las cabeceras de columna en la sección `MEMINFO` significan lo siguiente:

- Sección de memoria de objeto (`HEAPTYPE`):

id El ID del espacio o región.

start La dirección inicial de esta región del almacenamiento dinámico.

end La dirección final de esta región del almacenamiento dinámico.

size El tamaño de esta región del almacenamiento dinámico.

space/region
Para una línea que contiene solo un `id` y un nombre, esta columna muestra el nombre del espacio de memoria. De lo contrario, la columna muestra el nombre del espacio de memoria seguido del nombre de una región determinada incluida dentro de ese espacio de memoria.

- Sección de memoria interna (`SEGTYPE`), incluidas memoria de clase, memoria caché de código JIT y memoria caché de datos JIT:

segment
La dirección de la estructura de datos de control del segmento.

start La dirección inicial del segmento de memoria nativa.

alloc La dirección de asignación actual dentro del segmento de memoria nativa.

end La dirección final del segmento de memoria nativa.

type Un campo de bits interno que describe las características del segmento de memoria nativa.

size El tamaño del segmento de memoria nativa.

```
0SECTION       MEMINFO subcomponent dump routine
NULL           ================================
NULL
1STHEAPTYPE    Object Memory
NULL           id         start      end        size       space/region
1STHEAPSPACE   0x00497030     --         --         --     Generational
1STHEAPREGION  0x004A24F0 0x02850000 0x05850000 0x03000000 Generational/Tenured Region
1STHEAPREGION  0x004A2468 0x05850000 0x06050000 0x00800000 Generational/Nursery Region
1STHEAPREGION  0x004A23E0 0x06050000 0x06850000 0x00800000 Generational/Nursery Region
NULL
1STHEAPTOTAL   Total memory:           67108864 (0x04000000)
1STHEAPINUSE   Total memory in use:    33973024 (0x02066320)
1STHEAPFREE    Total memory free:      33135840 (0x01F99CE0)
NULL
1STSEGTYPE     Internal Memory
NULL           segment    start      alloc      end        type       size
1STSEGMENT     0x073DFC9C 0x0761B090 0x0761B090 0x0762B090 0x01000040 0x00010000
       (líneas eliminadas para facilitar la lectura)
1STSEGMENT     0x00497238 0x004FA220 0x004FA220 0x0050A220 0x00800040 0x00010000
NULL
1STSEGTOTAL    Total memory:           873412 (0x000D53C4)
1STSEGINUSE    Total memory in use:         0 (0x00000000)
1STSEGFREE     Total memory free:      873412 (0x000D53C4)
NULL
1STSEGTYPE     Class Memory
NULL           segment    start      alloc      end        type       size
1STSEGMENT     0x0731C858 0x0745C098 0x07464098 0x07464098 0x00010040 0x00008000
        (líneas eliminadas para facilitar la lectura)
1STSEGMENT     0x00498470 0x070079C8 0x07026DC0 0x070279C8 0x00020040 0x00020000
NULL
1STSEGTOTAL    Total memory:          2067100 (0x001F8A9C)
1STSEGINUSE    Total memory in use:   1839596 (0x001C11EC)
1STSEGFREE     Total memory free:     227504 (0x000378B0)
NULL
1STSEGTYPE     JIT Code Cache
NULL           segment    start      alloc      end        type       size
1STSEGMENT     0x004F9168 0x06960000 0x069E0000 0x069E0000 0x00000068 0x00080000
NULL
1STSEGTOTAL    Total memory:           524288 (0x00080000)
1STSEGINUSE    Total memory in use:    524288 (0x00080000)
1STSEGFREE     Total memory free:           0 (0x00000000)
NULL
1STSEGTYPE     JIT Data Cache
```

```
NULL           segment    start      alloc      end        type       size
1STSEGMENT     0x004F92E0 0x06A60038 0x06A6839C 0x06AE0038 0x00000048 0x00080000
NULL
1STSEGTOTAL    Total memory:           524288 (0x00080000)
1STSEGINUSE    Total memory in use:     33636 (0x00008364)
1STSEGFREE     Total memory free:      490652 (0x00077C9C)
NULL
1STGCHTYPE     GC History
3STHSTTYPE     15:18:14:901108829 GMT j9mm.134 -   Allocation failure end: newspace=7356368/8388608
oldspace=32038168/50331648 loa=3523072/3523072
3STHSTTYPE     15:18:14:901104380 GMT j9mm.470 -   Allocation failure cycle end: newspace=7356416/8388608
oldspace=32038168/50331648 loa=3523072/3523072
3STHSTTYPE     15:18:14:901097193 GMT j9mm.65 -   LocalGC end: rememberedsetoverflow=0
causedrememberedsetoverflow=0 scancacheoverflow=0 failedflipcount=0 failedflipbytes=0 failedtenurecount=0
failedtenurebytes=0 flipcount=11454 flipbytes=991056 newspace=7356416/8388608 oldspace=32038168/50331648
loa=3523072/3523072 tenureage=1
3STHSTTYPE     15:18:14:901081108 GMT j9mm.140 -   Tilt ratio: 50
3STHSTTYPE     15:18:14:893358658 GMT j9mm.64 -   LocalGC start: globalcount=3 scavengecount=24 weakrefs=0
soft=0 phantom=0 finalizers=0
3STHSTTYPE     15:18:14:893354551 GMT j9mm.63 -   Set scavenger backout flag=false
3STHSTTYPE     15:18:14:893348733 GMT j9mm.135 -   Exclusive access: exclusiveaccessms=0.002
meanexclusiveaccessms=0.002 threads=0 lastthreadtid=0x00495F00 beatenbyotherthread=0
3STHSTTYPE     15:18:14:893348391 GMT j9mm.469 -   Allocation failure cycle start: newspace=0/8388608
oldspace=38199368/50331648 loa=3523072/3523072 requestedbytes=48
3STHSTTYPE     15:18:14:893347364 GMT j9mm.133 -   Allocation failure start: newspace=0/8388608
oldspace=38199368/50331648 loa=3523072/3523072 requestedbytes=48
3STHSTTYPE     15:18:14:866523613 GMT j9mm.134 -   Allocation failure end: newspace=2359064/8388608
oldspace=38199368/50331648 loa=3523072/3523072
3STHSTTYPE     15:18:14:866519507 GMT j9mm.470 -   Allocation failure cycle end: newspace=2359296/8388608
oldspace=38199368/50331648 loa=3523072/3523072
3STHSTTYPE     15:18:14:866513004 GMT j9mm.65 -   LocalGC end: rememberedsetoverflow=0
causedrememberedsetoverflow=0 scancacheoverflow=0 failedflipcount=5056 failedflipbytes=445632
failedtenurecount=0 failedtenurebytes=0 flipcount=9212 flipbytes=6017148 newspace=2359296/8388608
oldspace=38199368/50331648 loa=3523072/3523072 tenureage=1
3STHSTTYPE     15:18:14:866493839 GMT j9mm.140 -   Tilt ratio: 64
3STHSTTYPE     15:18:14:859814852 GMT j9mm.64 -   LocalGC start: globalcount=3 scavengecount=23 weakrefs=0
soft=0 phantom=0 finalizers=0
3STHSTTYPE     15:18:14:859808692 GMT j9mm.63 -   Set scavenger backout flag=false
3STHSTTYPE     15:18:14:859801848 GMT j9mm.135 -   Exclusive access: exclusiveaccessms=0.004
meanexclusiveaccessms=0.004 threads=0 lastthreadtid=0x00495F00 beatenbyotherthread=0
3STHSTTYPE     15:18:14:859801163 GMT j9mm.469 -   Allocation failure cycle start: newspace=0/10747904
oldspace=38985800/50331648 loa=3523072/3523072 requestedbytes=232
3STHSTTYPE     15:18:14:859800479 GMT j9mm.133 -   Allocation failure start: newspace=0/10747904
oldspace=38985800/50331648 loa=3523072/3523072 requestedbytes=232
3STHSTTYPE     15:18:14:652219028 GMT j9mm.134 -   Allocation failure end: newspace=2868224/10747904
oldspace=38985800/50331648 loa=3523072/3523072
3STHSTTYPE     15:18:14:650796714 GMT j9mm.470 -   Allocation failure cycle end: newspace=2868224/10747904
oldspace=38985800/50331648 loa=3523072/3523072
3STHSTTYPE     15:18:14:650792607 GMT j9mm.475 -   GlobalGC end: workstackoverflow=0 overflowcount=0
memory=41854024/61079552
3STHSTTYPE     15:18:14:650784052 GMT j9mm.90 -   GlobalGC collect complete
3STHSTTYPE     15:18:14:650780971 GMT j9mm.57 -   Sweep end
3STHSTTYPE     15:18:14:650611567 GMT j9mm.56 -   Sweep start
3STHSTTYPE     15:18:14:650610540 GMT j9mm.55 -   Mark end
3STHSTTYPE     15:18:14:645222792 GMT j9mm.54 -   Mark start
3STHSTTYPE     15:18:14:645216632 GMT j9mm.474 -   GlobalGC start: globalcount=2
           (líneas eliminadas para facilitar la lectura)
NULL
NULL           ------------------------------------------------------------------------
```

Hebras y rastreo de la pila (THREADS)

Para el programador de aplicaciones, una de las partes más útiles de un volcado Java es la sección THREADS. En esta sección se muestra una lista de hebras de Java, hebras nativas y seguimientos de la pila.

Una hebra Java es implementada por una hebra nativa del sistema operativo. Cada hebra se representa por un conjunto de líneas como las siguientes:

```
"main" J9VMThread:0x41D11D00, j9thread_t:0x003C65D8, java/lang/Thread:0x40BD6070, state:CW, prio=5
 (native thread ID:0xA98, native priority:0x5, native policy:UNKNOWN)
 Java callstack:
 at java/lang/Thread.sleep(Native Method)
 at java/lang/Thread.sleep(Thread.java:862)
 at mySleep.main(mySleep.java:31)
```

Los nombres de hebras de Java se pueden ver en el sistema operativo cuando se utiliza el mandato **ps**. Para obtener más información sobre cómo utilizar el mandato **ps**, consulte "Técnicas de depuración generales" en la página 36.

Las propiedades de la primera línea son el nombre de hebra, la dirección de las estructuras de la JVM y del objeto, el estado y la prioridad de la hebra de Java thread object, thread state, and Java. Las propiedades de la segunda línea son el ID de hebra de sistema operativo nativo, la prioridad de hebra de sistema operativo nativo y la política de planificación del sistema operativo nativo.

Los nombres de hebra son visibles de tres maneras:

- Listados en archivos javacore. No todas las hebras aparecen en archivos javacore.
- Al listar hebras del sistema operativo utilizando el mandato **ps**.
- Al utilizar el método java.lang.Thread.getName().

La tabla siguiente proporciona información sobre los nombres de hebras de IBM WebSphere Real Time for Linux.

Tabla 4. Nombres de hebra en IBM WebSphere Real Time for Linux

Detalle de la hebra	Nombre de hebra
Una hebra interna de la JVM que utiliza la recopilación de basura para asignar la finalización de objetos por hebras secundarias.	`Maestro finalizador`
La hebra de alarma que utiliza el recopilador de basura (GC).	`Alarma del GC`
Las hebras esclavas que se utilizan para la recopilación de basura.	`Esclava del GC`
Una hebra interna de la JVM que utiliza el módulo del compilador JIT (just-in-time) para obtener muestras del uso de los métodos en la aplicación.	`IProfiler`
Una hebra que utiliza la máquina virtual (VM) para gestionar las señales que recibe la aplicación, tanto si se generan interna como externamente.	`Informador de señales`
Una hebra de la JVM interna que se utiliza para compilar código Java.	`JIT Compilation Thread`
Una hebra de la JVM interna que se utiliza para permitir que los agentes JVMTI se conecten a una JVM en ejecución.	`Attach API wait loop`

La prioridad de hebra Java se correlaciona con un valor de prioridad de sistema operativo en función de la plataforma. Un valor grande para la prioridad de hebra Java significa que la hebra tiene una prioridad alta. Es decir, la hebra se ejecuta con más frecuencia que las hebras de prioridad inferior.

Los valores de estado pueden ser:

- R (Runnable): Ejecutable; la hebra puede ejecutarse cuando tenga oportunidad.
- CW (Condition Wait): Condición de espera; la hebra está esperando. Por ejemplo, porque:
  - Se ha realizado una llamada sleep()

- Se ha bloqueado la hebra para E/S
- Se ha llamado al método wait() para que espere en un supervisor que se está notificando
- La hebra está sincronizándose con otra hebra con una llamada join()

- S (Suspended): Suspendida; la hebra ha sido suspendida por otra hebra.
- Z (Zombie): Inerte; la hebra se ha detenido.
- P (Parked): Aparcada; la hebra ha sido aparcada por la nueva API de simultaneidad (java.util.concurrent).
- B (Blocked): Bloqueada; la hebra está esperando para obtener un bloqueo que posee actualmente otra cosa.

Si una hebra está aparcada o bloquead, la información de salida contendrá una línea correspondiente a dicha hebra, comenzando por 3XMTHREADBLOCK, indicando el recurso que esa hebra está esperando y, cuando sea posible, indicando la hebra que actualmente posee dicho recurso. Para obtener más información, consulte el tema sobre hebras bloqueadas en la Guía de usuario IBM SDK para Java V7.

Cuando se inicia un volcado de Java para obtener información sobre un diagnóstico, la JVM desactiva temporalmente las hebras de Java antes de generar el archivo javacore. Un estado de preparación de exclusive_vm_access se muestra en la línea 1TIPREPSTATE del apartado TITLE.

```
1TIPREPSTATE Prep State: 0x4 (exclusive_vm_access)
```

Las hebras que están ejecutando código Java cuando se desencadenó el javacore están es estado CW (Condición de espera).

```
3XMTHREADINFO      "main" J9VMThread:0x41481900, j9thread_t:0x002A54A4, java/lang/Thread:0x004316B8,
state:CW, prio=5
3XMTHREADINFO1            (native thread ID:0x904, native priority:0x5, native policy:UNKNOWN)
3XMTHREADINFO3           Java callstack:
4XESTACKTRACE                at java/lang/String.getChars(String.java:667)
4XESTACKTRACE                at java/lang/StringBuilder.append(StringBuilder.java:207)
```

El apartado The LOCKS del javacore muestra que están hebras están esperando en un bloqueo interno de la JVM.

```
2LKREGMON          Thread public flags mutex lock (0x002A5234): <unowned>
3LKNOTIFYQ            Waiting to be notified:
3LKWAITNOTIFY            "main" (0x41481900)
```

Utilizar el vuelco de almacenamiento dinámico

El término Heapdump describe el mecanismo de máquina virtual de IBM para Java que genera un volcado de todos los objetos activos que están en el almacenamiento dinámico de Java; es decir, los que se utilizan ejecutando la aplicación Java.

La Guía de usuario de IBM SDK para Java V7 contiene instrucciones útiles sobre volcados de almacenamiento dinámico, y cubre:

- Obtención de volcados de almacenamiento dinámico
- Herramientas para procesar volcados de almacenamiento dinámico
- Uso de **-Xverbose:gc** para obtener información de almacenamiento dinámico
- Variables de entorno y Heapdump
- Formato de archivo Heapdump de texto (clásico)
- Formato de archivo de vuelco de almacenamiento dinámico portátil (PHD)

Puede encontrar esta información en: IBM SDK for Java 7 - Using Heapdump.

Información suplementaria para IBM WebSphere Real Time for Linux:

Formato de archivo Heapdump de texto (clásico)

El Heapdump de texto o clásico es una lista de todas las instancias de objeto del almacenamiento dinámico, que incluyen el tipo de objeto, el tamaño y las referencias entre objetos.

Registro de cabecera

El registro de cabecera es un único registro que contiene una serie de información de la versión.

```
// Version: <serie de versión que contiene el nivel del SDK, nivel de build de la JVM y la plataforma
```

Ejemplo:

```
// Version: J2RE 7.0 IBM J9 2.6 Linux x86-32 build 20101016_024574_lHdRSr
```

Registros de objetos

Los registros de objetos son varios registros, uno para cada instancia de objeto del almacenamiento dinámico, que proporcionan la dirección de objeto, el tamaño, el tipo y las referencias del objeto.

```
<object address, in hexadecimal> [<length in bytes of object instance, in decimal>]
OBJ <object type> <class block reference, in hexadecimal>
<heap reference, in hexadecimal <heap reference, in hexadecimal> ...
```

Las referencias de dirección de objeto y almacenamiento dinámico están en el almacenamiento dinámico pero la dirección de bloque de clases está fuera del almacenamiento dinámico. Todas las referencias encontradas en la instancia de objeto aparecen, incluidas las referencias de valores nulos. El tipo de objeto es un nombre de clase que incluye paquete o una matriz primitiva o tipo de matriz de clase, mostrada mediante su firma de tipo de JVM estándar; consulte “Firmas de tipo de la máquina virtual Java” en la página 58. Los registros de objetos también pueden contener referencias de bloque de clases adicionales, normalmente en el caso de las instancias de clase de reflejo.

Ejemplos:

Una instancia de objeto, longitud de 28 bytes, de tipo java/lang/String:

```
0x00436E90 [28] OBJ java/lang/String
```

Una dirección de bloque de clases de java/lang/String, seguida de una referencia a la instancia de matriz de caracteres:

```
0x415319D8 0x00436EB0
```

Una instancia de objeto, longitud de 44 bytes, de tipo matriz de caracteres:

```
0x00436EB0 [44] OBJ [C
```

Una dirección de bloque de clases de matriz de caracteres:

```
0x41530F20
```

Un objeto de tipo matriz de java/util/Hashtable Entry inner class:

```
0x004380C0 [108] OBJ [Ljava/util/Hashtable$Entry;
```

Un objeto de tipo java/util/Hashtable Entry inner class:

```
0x4158CD80 0x00000000 0x00000000 0x00000000 0x00000000 0x00421660 0x004381C0
0x00438130 0x00438160 0x00421618 0x00421690 0x00000000 0x00000000 0x00000000
0x00438178 0x004381A8 0x004381F0 0x00000000 0x004381D8 0x00000000 0x00438190
0x00000000 0x004216A8 0x00000000 0x00438130 [24] OBJ java/util/Hashtable$Entry
```

Una dirección de bloque de clases y referencias de almacenamiento dinámico, incluidas las referencias nulas:

```
0x4158CB88 0x004219B8 0x004341F0 0x00000000
```

Registros de clases

Los registros de clases son varios registros, uno de cada clase cargada, que proporcionan la dirección de bloque de clases, el tamaño, el tipo y las referencias de la clase.

```
<class block address, in hexadecimal> [<length in bytes of class block, in decimal>]
CLS <class type>
<class block reference, in hexadecimal> <class block reference, in hexadecimal> ...
<heap reference, in hexadecimal> <heap reference, in hexadecimal>...
```

La dirección de bloque de clases y las referencias de bloque de clases están fuera del almacenamiento dinámico, pero el registro de clases también puede contener referencias en el almacenamiento dinámico, normalmente para miembros de datos de clase estática. Todas las referencias encontradas en el bloque de clases aparecen, incluidas las de valores nulos. El tipo de clase es un nombre de clase que incluye paquete o una matriz primitiva o tipo de matriz de clase, mostrada mediante su firma de tipo de JVM estándar; consulte “Firmas de tipo de la máquina virtual Java” en la página 58.

Ejemplos:

Un bloque de clases, longitud de 32 bytes, para clase java/lang/Runnable:

```
0x41532E68 [32] CLS java/lang/Runnable
```

Referencias a otros bloques de clases y referencias de almacenamiento dinámico, incluidas las referencias nulas:

```
0x4152F018 0x41532E68 0x00000000 0x00000000 0x00499790
```

Un bloque de clases, longitud de 168 bytes, para clase java/lang/Math:

```
0x00000000 0x004206A8 0x00420720 0x00420740 0x00420760 0x00420780 0x004207B0
0x00421208 0x00421270 0x00421290 0x004212B0 0x004213C8 0x00421458 0x00421478
0x00000000 0x41589DE0 0x00000000 0x4158B340 0x00000000 0x00000000 0x00000000
0x4158ACE8 0x00000000 0x4152F018 0x00000000 0x00000000 0x00000000
```

Registro de cola 1

El registro de cola 1 es un registro único que contiene recuentos de registros.

```
// Breakdown - Classes: <class record count, in decimal>,
Objects: <object record count, in decimal>,
ObjectArrays: <object array record count, in decimal>,
PrimitiveArrays: <primitive array record count, in decimal>
```

Ejemplo:

```
// Breakdown - Classes: 321, Objects: 3718, ObjectArrays: 169,
PrimitiveArrays: 2141
```

Registro de cola 2

El registro de cola 2 es un registro único que contiene totales.

```
// EOF:  Total 'Objects',Refs(null) :
<total object count, in decimal>,
<total reference count, in decimal>
(,total null reference count, in decimal>)
```

Ejemplo:

```
// EOF:  Total 'Objects',Refs(null) : 6349,23240(7282)
```

Firmas de tipo de la máquina virtual Java

Las firmas de tipo de la máquina virtual Java son abreviaciones de los tipos de Java que aparecen en la siguiente tabla:

Firmas de tipo de la máquina virtual Java	Tipo de Java
Z	booleano
B	byte
C	carácter
S	corto
I	int
J	largo
F	flotante
D	doble
L <*clase completa*> ;	<*clase completa*>
[<*tipo*>	<*tipo*>[ ] (matriz de <*tipo*>)
(<*arg-tipos*>) <*ret-tipo*>	método

Uso de volcados del sistema y el visor de volcados

La JVM puede generar volcados del sistema nativos, también conocidos como volcados del núcleo, en condiciones configurables. Los volcados del sistema suelen ser grandes. La mayoría de las herramientas usadas para analizar los volcados del sistema también son específicas para cada plataforma.Use la herramienta **gdb** para analizar un volcado del sistema en Linux.

La Guía de usuario de IBM SDK para Java V7 contiene instrucciones útiles sobre el uso del visor de volcados y los volcados del sistema, y cubre:

- Visión general de volcados del sistema
- Valores predeterminados de volcado del sistema
- Uso del visor de volcados
  - Uso de **jextract**
  - Problemas con los que se enfrenta el visor de volcados
  - Mandatos disponibles en **jdmpview**
  - Sesión de ejemplo
  - Referencia rápida de mandatos **jdmpview**

Puede encontrar esta información aquí:IBM SDK para Java 7 - Uso del visor de volcados y volcados del sistema.

Información suplementaria para IBM WebSphere Real Time for Linux:

Mandatos disponibles en `jdmpview`

`jdmpview` es una herramienta de línea de mandatos interactiva para explorar la información desde un volcado del sistema JVM y realizar varias funciones de análisis.

`info jitm`
Lista los métodos AOT y JIT compilados y sus direcciones:
- Nombre de método y signatura
- Dirección inicial de método
- Dirección final de método

Para ver el resto de opciones de mandato, consulte la Guía de usuario de IBM SDK para Java V7.

Rastreo de aplicaciones Java y la JVM

El rastreo de la máquina virtual Java (JVM) es un recurso de rastreo que se facilita en IBM WebSphere Real Time for Linux con una impacto mínimo en el rendimiento. En la mayoría de los casos, los datos de rastreo se conservan en un formato binario compacto, que se puede formatear con el formateador Java que se suministra.

El rastreo está habilitado de forma predeterminada, junto con un pequeño conjunto de puntos de rastreo que se dirigen a las memorias intermedias. Puede habilitar los puntos de rastreo en tiempo de ejecución utilizando niveles, componentes, nombres de grupos o identificadores de puntos de rastreo individuales.

La Guía del usuario de IBM SDK para Java V7 contiene información detallada sobre aplicaciones de rastreo, cubriendo:
- Qué se puede rastrear
- Tipos de punto de rastreo
- Rastreo predeterminado
- Registro de datos de rastreo
- Control del rastreo
- Rastreo de aplicaciones Java
- Rastreo de métodos Java

Cuando rastrea IBM WebSphere Real Time for Linux, debe invocar correctamente la JVM en tiempo real cuando incluya las opciones de rastreo. Por ejemplo, cuando especifique opciones de rastreo, escriba:

```
java -Xgcpolicy:metronome -Xtrace:<options>
```

Puede encontrar información de IBM SDK para Java V7 aquí: Rastreo de aplicaciones Java y de la JVM.

Determinación de problemas de JIT y AOT

Puede utilizar opciones de línea de mandatos para ayudar a diagnosticar problemas del compilador JIT y AOT y para ajustar el rendimiento.

Aunque IBM WebSphere Real Time for Linux comparte algunos componentes comunes con el IBM SDK para Java V7, el comportamiento de JIT y AOT es

distinto. En esta sección se trata la resolución de problemas para incidencias de JIT y AOT en IBM WebSphere Real Time for Linux.

Diagnóstico de un problema de JIT o AOT

Ocasionalmente, los códigos de bytes válidos podrían compilarse en código nativo no válido, haciendo que el programa Java falle. Al determinar si el compilador JIT o AOT está defectuoso y, si es así, *dónde* está defectuoso, podrá proporcionar ayuda válida al equipo de servicio de Java.

Acerca de esta tarea

Para determinar qué métodos están compilados al rellenarse la memoria caché de clase compartida, utilice la opción **-Xaot:verbose** de la línea de mandatos de admincache. Por ejemplo:

```
admincache -Xrealtime -Xaot:verbose -populate -aot my.jar -cp <Mi classpath>
```

Este apartado describe cómo puede determinar si el problema está relacionado con el compilador. Este apartado también sugiere algunos métodos alternativos posibles y técnicas de depuración para resolver problemas relacionados con el compilador.

Inhabilitación del compilador JIT o AOT:

Si sospecha que se está produciendo un problema en el compilador JIT o AOT, inhabilite la compilación para ver si el problema continúa. Si el problema sigue produciéndose, sabrá que el compilador no es su causa.

Acerca de esta tarea

El compilador JIT está habilitado de forma predeterminada. El compilador AOT también está habilitado, pero, no está activo a menos que estén habilitadas las clases compartidas. Por motivos de eficacia, no todos los métodos de una aplicación Java están compilados. La JVM mantiene un recuento de llamadas para cada método de la aplicación; cada vez que se llama y se interpreta un método, el recuento de llamadas de dicho método se incrementa. Cuando el recuento alcanza el umbral de compilación, el método se compila y se ejecuta de forma nativa.

El mecanismo de recuento de llamadas extiende la compilación de métodos durante toda la vida de la aplicación, dando mayor prioridad a los métodos que se utilizan con más frecuencia. Algunos métodos utilizados con menos frecuencia podrían no compilarse nunca. Como resultado, cuando falla un programa Java, el problema podría estar en el compilador JIT o AOT o podría estar en otro lugar de la JVM.

El primer paso para diagnosticar la anomalía es determinar *dónde* está el problema. Para ello, debe primero ejecutar el programa Java en modalidad puramente interpretada (es decir, con los compiladores JIT y AOT inhabilitados).

Procedimiento

1. Elimine las opciones **-Xjit**y **-Xaot** (y los parámetros que las acompañan) de la línea de mandatos.
2. Utilice la opción de línea de mandatos **-Xint** para inhabilitar los compiladores JIT y AOT. Por motivos de rendimiento, no utilice la opción **-Xint** en un entorno de producción.

Qué hacer a continuación

Si ejecuta el programa Java con la compilación inhabilitada se producirá una de las siguientes situaciones:

- La anomalía continúa. El problema no está en el compilador JIT ni AOT. En algunos casos, el programa podría empezar a fallar de otra manera; sin embargo, el problema no estará relacionado con el compilador.
- La anomalía desaparece. Lo más probable es que el problema esté en el compilador JIT o AOT.

  Si no utiliza clases compartidas, el compilador JIT estará defectuoso. Si utiliza clases compartidas, deberá determinar qué compilador está defectuoso ejecutando su aplicación sólo con la compilación JIT habilitada. Ejecute la aplicación con la opción **-Xnoaot** en lugar de con la opción **-Xint**. Esto le llevará a una de las situaciones siguientes:
  - La anomalía continúa. El problema estará en el compilador JIT. También puede utilizar **-Xnojit** en lugar de la opción **-Xnoaot** para asegurarse de que sólo el compilador JIT esté defectuoso.
  - La anomalía desaparece. El problema estará en el compilador AOT.

Inhabilitación selectiva del compilador JIT:

Si la anomalía del programa Java apunta a un problema con el compilador JIT, podrá intentar de esta forma reducir más el ámbito del problema.

Acerca de esta tarea

De forma predeterminada, el compilador JIT optimiza métodos en diversos niveles de optimización. Diferentes selecciones de optimizaciones se aplican a distintos métodos, en función de sus recuentos de llamadas. Los métodos a los que se llama con más frecuencia se optimizan en niveles superiores. Al cambiar los parámetros del compilador JIT, puede controlar el nivel de optimización al que se optimizan los métodos. Puede determinar si el optimizador está defectuoso y, si lo está, qué optimización es problemática.

Los parámetros de JIT se especifican como lista separada por comas, añadidos a la opción **-Xjit**. La sintaxis es **-Xjit**:*<param1>*,*<param2>*=*<valor>*. Por ejemplo:

```
java -Xjit:verbose,optLevel=noOpt HelloWorld
```

lanza el programa HelloWorld, habilita verbose output desde JIT y hace que JIT genere el código nativo sin realizar ninguna optimización.

Siga estos pasos para determinar qué parte del compilador está causando la anomalía:

Procedimiento

1. Establezca el parámetro de JIT **count=0** para cambiar el umbral de compilación a cero. Este parámetro hará que cada método Java se compile antes de ejecutarse. Utilice **count=0** únicamente cuando diagnostique problemas porque se compilen muchos métodos, incluidos métodos que se utilizan con poca frecuencia. La compilación adicional utiliza más recursos informáticos para la compilación y ralentiza la aplicación. Con **count=0**, la aplicación falla inmediatamente cuando se alcance el área del problema. En algunos casos, el uso de **count=1** puede reproducir la anomalía de forma más fiable.

2. Añada **disableInlining** a los parámetros del compilador JIT. **disableInlining** inhabilita la generación de código más grande y más complejo. Si el problema ya no se produce, utilice **disableInlining** como método alternativo mientras el equipo de servicio de Java analiza y soluciona el problema del compilador.
3. Disminuya los niveles de optimización añadiendo el parámetro **optLevel** y vuelva a ejecutar el programa hasta que ya no se produzca la anomalía o se alcance el nivel "noOpt". En caso de problema con el compilador JIT, empiece con "scorching" y vaya bajando por la lista. Los niveles de optimización son, en orden decreciente:
   a. scorching
   b. veryHot
   c. hot
   d. warm
   e. cold
   f. noOpt

Qué hacer a continuación

Si uno de estos valores hace que desaparezca la anomalía, tendrá un método alternativo que podrá utilizar. Este método alternativo es temporal mientras el equipo de servicio de Java analiza y soluciona el problema del compilador. Si eliminar **disableInlining** de la lista de parámetros de JIT hace que no reaparezca la anomalía, hágalo así para mejorar el rendimiento. Siga las instrucciones de "Ubicación del método erróneo" para mejorar el rendimiento del método alternativo.

Si sigue produciéndose la anomalía en el nivel de optimización "noOpt", deberá inhabilitar el compilador JIT como método alternativo.

Ubicación del método erróneo:

Cuando haya determinado el nivel de optimización más bajo al que el compilador JIT o AOT debe compilar métodos para desencadenar la anomalía, podrá descubrir qué parte del programa Java, cuando está compilado, causa la anomalía. A continuación, puede indicar al compilador que limite el método alternativo a un método, una clase o un paquete específicos, permitiendo al compilador compilar el resto del programa como lo hace normalmente. En caso de anomalías del compilador JIT, si se produce la anomalía con **-Xjit:optLevel=noOpt**, también puede indicar al compilador que no compile el método o los métodos que causan la anomalía.

Antes de empezar

Si ve una salida de error como la de este ejemplo, puede utilizarla para identificar el método erróneo:

```
Unhandled exception
Type=Segmentation error vmState=0x00000000
Target=2_30_20050520_01866_BHdSMr (Linux 2.4.21-27.0.2.EL)
CPU=s390x (2 logical CPUs) (0x7b6a8000 RAM)
J9Generic_Signal_Number=00000004 Signal_Number=0000000b Error_Value=4148bf20 Signal_Code=00000001
Handler1=00000100002ADB14 Handler2=00000100002F480C InaccessibleAddress=0000000000000000
gpr0=0000000000000006 gpr1=0000000000000006 gpr2=0000000000000000 gpr3=0000000000000006
gpr4=0000000000000001 gpr5=0000000080056808 gpr6=0000010002BCCA20 gpr7=0000000000000000
......
Compiled_method=java/security/AccessController.toArrayOfProtectionDomains([Ljava/lang/Object;
Ljava/security/AccessControlContext;)[Ljava/security/ProtectionDomain;
```

Las líneas importantes son:

vmState=0x00000000
Indica que el código que ha fallado no era código de tiempo de ejecución de la JVM.

Module= o Module_base_address=
No en la salida (podría estar vacía o ser cero) porque el código fue compilado por el JIT, y fuera de cualquier DLL o biblioteca.

Compiled_method=
Indica el método Java para el que se produjo el código compilado.

Acerca de esta tarea

Si la salida no indica el método erróneo, siga estos pasos para identificar el método erróneo:

Procedimiento

1. Ejecute el programa Java con los parámetros de JIT **verbose** y **vlog=**<*nombre_archivo*> añadido a la opción **-Xjit**o **-Xaot**. Con estos parámetros, el compilador lista métodos compilados en un archivo de registro llamado <*nombre_archivo*>.<*fecha*>.<*hora*>.<*PID*>, también denominado *archivo de límite*. Un archivo de límite normal contiene líneas que corresponden a los métodos compilados, como:

   ```
   + (hot) java/lang/Math.max(II)I @ 0x10C11DA4-0x10C11DDD
   ```

   Las líneas que no empiezan por el signo más son ignoradas por el compilador en los siguientes pasos y puede eliminarlas del archivo. Los métodos compilados por el compilador AOT empiezan por `+ (AOT cold)`. Los métodos para los que se carga el código AOT desde la memoria caché de clase compartida empiezan por `+ (AOT load)`.
2. Ejecute de nuevo el programa con el parámetro de JIT o AOT **limitFile=**(<*nombre_archivo*>,<*m*>,<*n*>), donde <*nombre_archivo*> es la vía de acceso al archivo de límite, y <*m*> y <*n*> son los números de línea que indican el primer y el último método del archivo de límite que deben compilarse. El compilador solo compila los métodos mostrados en las líneas de <*m*> a <*n*> en el archivo de límite. Los métodos no mostrados en el archivo de límite y los métodos mostrados en líneas fuera del rango no se compilan; no se cargará ningún código AOT de la memoria caché de datos compartidos para dichos métodos. Si el programa ya no falla, uno o varios de los métodos que ha eliminado en la última iteración habrán sido la causa de la anomalía.
3. Opcional: Si está diagnosticando un problema de AOT, ejecute el programa una segunda vez con las mismas opciones para permitir que los métodos compilados se carguen desde la memoria caché de datos compartidos. También puede añadir la opción **–Xaot:scount=0** para asegurarse de que los métodos compilados por AOT almacenados en la memoria caché de datos compartidos se utilicen al llamar por primera vez al método. Algunas anomalías de compilación AOT se producen solo cuando el código compilado por AOT se carga desde la memoria caché de datos compartidos. Para ayudar a diagnosticar estos problemas, utilice la opción **–Xaot:scount=0** para asegurarse de que los métodos compilados por AOT almacenados en la memoria caché de datos compartidos se utilicen al llamar al método por primera vez, lo que podría facilitar la reproducción del problema. Tenga en cuenta que si establece la opción **scount** en 0 forzará la carga del código AOT y pausará cualquier hebra de la aplicación que espere para ejecutar dicho método. Por tanto, solo

debería utilizarse para fines de diagnóstico. Con la opción **–Xaot:scount=0** se pueden producir tiempos de pausa más significativos.

4. Repita este proceso utilizando valores diferentes para <*m*> y <*n*>, tantas veces como sea necesario, para encontrar el conjunto mínimo de métodos que deben compilarse para desencadenar la anomalía. Dividiendo por dos el número de líneas seleccionadas cada vez, puede realizar una búsqueda binaria del método erróneo. A menudo, puede reducir el archivo a una sola línea.

Qué hacer a continuación

Cuando haya ubicado el método erróneo, podrá inhabilitar el compilador JIT o AOT únicamente para el método erróneo. Por ejemplo, si el método java/lang/Math.max(II)I hace que falle el programa cuando se realiza la compilación JIT con **optLevel=hot**, podrá ejecutar el programa con:

```
-Xjit:{java/lang/Math.max(II)I}(optLevel=warm,count=0)
```

para compilar únicamente el método erróneo a un nivel de optimización de "warm", aunque compilando los otros métodos como de costumbre.

Si un método falla cuando se compila mediante JIT en "noOpt", puede excluirlo totalmente de la compilación, utilizando el parámetro **exclude**={<*método*>}:

```
-Xjit:exclude={java/lang/Math.max(II)I}
```

Si un método hace que falle el programa cuando el código AOT se compila o se carga desde la memoria caché de datos compartidos, excluya el método de la compilación AOT y la carga AOT utilizando el parámetro **exclude**={<*método*>}:

```
-Xaot:exclude={java/lang/Math.max(II)I}
```

Los métodos AOT se compilan únicamente en el nivel de optimización "cold". Evitar la compilación AOT o la carga AOT es el mejor enfoque para estos métodos.

Identificación de anomalías de compilación JIT:

Para anomalías del compilador JIT, analice la salida de errores para determinar si se ha producido una anomalía cuando el compilador JIT intentaba compilar un método.

Si la JVM se bloquea y puede ver que la anomalía se ha producido en la biblioteca JIT (`libj9jit26.so`), puede que el compilador JIT haya fallado durante un intento de compilación de un método.

Si ve una salida de error como la de este ejemplo, puede utilizarla para identificar el método erróneo:

```
Unhandled exception
Type=Segmentation error vmState=0x00050000
Target=2_30_20051215_04381_BHdSMr (Linux 2.4.21-32.0.1.EL)
CPU=ppc64 (4 logical CPUs) (0xebf4e000 RAM)
J9Generic_Signal_Number=00000004 Signal_Number=0000000b Error_Value=00000000 Signal_Code=00000001
Handler1=0000007FE05645B8 Handler2=0000007FE0615C20
R0=E8D4001870C00001 R1=0000007FF49181E0 R2=0000007FE2FBCEE0 R3=0000007FF4E60D70
R4=E8D4001870C00000 R5=0000007FE2E02D30 R6=0000007FF4C0F188 R7=0000007FE2F8C290
......
Module=/home/test/sdk/jre/bin/libj9jit26.so
Module_base_address=0000007FE29A6000
......
Method_being_compiled=com/sun/tools/javac/comp/Attr.visitMethodDef(Lcom/sun/tools/javac/tree/
JCTree$JCMethodDecl;)
```

Las líneas importantes son:

vmState=0x00050000
Indica que el compilador JIT está compilando código. Para ver una lista de los números de código vmState, consulte la tabla de códigos del volcado Java en la Guía de usuario de IBM SDK para Java V7, http://publib.boulder.ibm.com/infocenter/java7sdk/v7r0/topic/com.ibm.java.lnx.70.doc/diag/tools/javadump_tags_info.html.

Module=/home/test/sdk/jre/bin/libj9jit26.so
Indica que el error se ha producido en `libj9jit26.so`, el módulo del compilador JIT.

Method_being_compiled=
Indica que el método de Java se está compilando.

Si la salida no indica el método erróneo, utilice la opción **verbose** con los siguientes valores adicionales:

```
-Xjit:verbose={compileStart|compileEnd}
```

Estos valores de **verbose** informan cuándo el compilador JIT empieza a compilar un método, y cuándo lo finaliza. Si JIT falla en un método concreto (esto es, inicia la compilación pero falla antes de poder finalizar), utilice el parámetro **exclude** para excluirla de la compilación (consulte "Ubicación del método erróneo" en la página 62). Si la exclusión del método impide el bloqueo, tendrá un método alternativo que podrá utilizar mientras el equipo de servicio corrige el problema.

Rendimiento de aplicaciones de ejecución breve

El compilador JIT de IBM está ajustado para las aplicaciones de larga ejecución que normalmente se utilizan en un servidor. Puede usar la opción de línea de mandatos **-Xquickstart** para mejorar el rendimiento de las aplicaciones de ejecución breve, en especial, de las aplicaciones en las que el proceso no está concentrado en pocos métodos.

-Xquickstart hace que el compilador JIT utilice un nivel de optimización inferior de forma predeterminada y que compile menos métodos. Realizar menos compilaciones más rápidamente puede mejorar el tiempo de inicio de las aplicaciones. Cuando el compilador AOT está activo (tanto las clases compartidas como la compilación AOT habilitadas), **-Xquickstart** hará que todos los métodos seleccionados para compilación sean compilados por AOT, lo que mejora el tiempo de inicio de las ejecuciones posteriores. **-Xquickstart** puede degradar el rendimiento si se utiliza con aplicaciones que tardan en ejecutarse y que contengan métodos que usan una gran cantidad de recursos de proceso. La implementación de **-Xquickstart** está sujeta a cambios en futuros releases.

También puede intentar mejorar los tiempos de inicio ajustando el umbral de JIT (utilizando prueba y error). Para obtener más información, consulte "Inhabilitación selectiva del compilador JIT" en la página 61.

Comportamiento de la JVM durante períodos desocupados

Puede reducir los ciclos de la CPU consumidos por una JVM desocupada utilizando la opción **-XsamplingExpirationTime** para desactivar la hebra de muestreo de JIT.

La hebra de muestreo de JIT perfila la aplicación Java en ejecución para descubrir métodos utilizados comúnmente. El uso de la memoria y el procesador de la hebra

de muestreo es insignificante y la frecuencia de definición de perfiles se reduce automáticamente cuando la JVM está desocupada.

En algunas circunstancias, puede que no desee ciclos de la CPU consumidos por una JVM desocupada. Para ello, especifique la opción **-XsamplingExpirationTime**<*tiempo*>. Establezca la <*tiempo*> en el número de segundos durante los cuales desea que se ejecute la hebra de muestreo. Utilice esta opción con precaución; una vez desactivada, no podrá reactivar la hebra de muestreo. Permita que la hebra de muestreo se ejecute durante el suficiente tiempo para identificar optimizaciones importantes.

El recopilador de diagnósticos

El recopilador de diagnósticos recopila los archivos de diagnóstico Java para un suceso de un problema.

La recopilación de los archivos necesarios para el servicio de IBM puede acelerar el el proceso de resolución de problemas. La Guía de usuario de IBM SDK para Java V7 contiene información detallada sobre el uso del Recopilador de diagnósticos.

Aquí puede encontrar esta información: IBM SDK para Java 7 - El Recopilador de diagnósticos.

Datos de diagnóstico del recopilador de basura

Este apartado describe cómo diagnosticar problemas en la recopilación de basura.

La Guía de usuario de IBM SDK para Java V7 contiene instrucciones útiles sobre el diagnóstico de problemas con la recopilación de basura, y cubre:

- registro cronológico detallado de la recopilación de basura
- rastreo de la recopilación de basura usando **-Xtgc**

Puede encontrar la información aquí: IBM SDK para Java 7 - Datos de diagnóstico del recopilador de basura.

En las secciones siguientes se proporciona información adicional sobre la Recogida de basura metronome de IBM WebSphere Real Time for Linux.

Resolución de problemas del recopilador de basura Metronome

Con las opciones de línea de mandatos, puede controlar la frecuencia de la recogida de basura Metronome, las excepciones de falta de memoria y el comportamiento de Metronome en llamadas explícitas al sistema.

Utilización de la información de verbose:gc:

Puede utilizar la opción **-verbose:gc** con la opción **-Xgc:verboseGCCycleTime=N** para escribir información en la consola sobre la actividad del sistema de recogida de basura de Metronome. No todas las propiedades XML de la salida **-verbose:gc** de la JVM estándar se crean o se aplican a la salida de Recopilador de basura Metronome.

Utilice la opción **-verbose:gc** para ver el espacio libre mínimo, máximo y medio del almacenamiento dinámico. De este modo, podrá comprobar el nivel de actividad y uso del almacenamiento dinámico y luego ajustar los valores, si fuese necesario. La opción **-verbose:gc** escribe las estadísticas de Metronome en la consola.

La opción **-Xgc:verboseGCCycleTime=N** controla la frecuencia de recuperación de la información. Determina el tiempo en milisegundos en el que se vuelcan los resúmenes. El valor predeterminado para N es 1000 milisegundos. El tiempo de ciclo no implica que el resumen se vuelque exactamente a dicha hora, sino cuando pasa el último suceso de recogida de basura que satisface este criterio temporal. La recogida y visualización de estas estadísticas pueden aumentar los objetivos de tiempo de pausa del recopilador de basura Metronome y, a medida que N se reduce, los tiempos de las pausas pueden ser grandes.

Una cantidad es un único periodo de actividad de Recopilador de basura Metronome, que provoca una interrupción o una pausa en una aplicación.

Ejemplo de salida de verbose:gc

Especifique:

```
java -Xgcpolicy:metronome -verbose:gc -Xgc:verboseGCCycleTime=N mi_aplicación
```

Cuando se desencadena la recogida de basura, se produce un suceso `trigger start`, seguido de varios sucesos `heartbeat` y, por último, un suceso `trigger end` cuando se satisface el desencadenante. En este ejemplo se muestra el ciclo de recogida de basura desencadenado como salida verbose:gc:

```
<trigger-start id="25" timestamp="2011-07-12T09:32:04.503" />

<cycle-start id="26" type="global" contextid="26" timestamp="2011-07-12T09:32:04.503" intervalms="984.285" />

<gc-op id="27" type="heartbeat" contextid="26" timestamp="2011-07-12T09:32:05.209">
  <quanta quantumCount="321" quantumType="mark" minTimeMs="0.367" meanTimeMs="0.524" maxTimeMs="1.878"
    maxTimestampMs="598704.070" />
  <exclusiveaccess-info minTimeMs="0.006" meanTimeMs="0.062" maxTimeMs="0.147" />
  <free-mem type="heap" minBytes="99143592" meanBytes="114374153" maxBytes="134182032" />
  <thread-priority maxPriority="11" minPriority="11" />
</gc-op>

<gc-op id="28" type="heartbeat" contextid="26" timestamp="2011-07-12T09:32:05.458">
  <quanta quantumCount="115" quantumType="sweep" minTimeMs="0.430" meanTimeMs="0.471" maxTimeMs="0.511"
    maxTimestampMs="599475.654" />
  <exclusiveaccess-info minTimeMs="0.007" meanTimeMs="0.067" maxTimeMs="0.173" />
  <classunload-info classloadersunloaded=9 classesunloaded=156  />
  <references type="weak" cleared="660" />
  <free-mem type="heap" minBytes="24281568" meanBytes="55456028" maxBytes="87231320" />
  <thread-priority maxPriority="11" minPriority="11" />
</gc-op>

<gc-op id="29" type="syncgc" timems="136.945" contextid="26" timestamp="2011-07-12T09:32:06.046">
  <syncgc-info reason="out of memory" exclusiveaccessTimeMs="0.006" threadPriority="11" />
  <free-mem-delta type="heap" bytesBefore="21290752" bytesAfter="171963656" />
</gc-op>

<cycle-end id="30" type="global" contextid="26" timestamp="2011-07-12T09:32:06.046" />

<trigger-end id="31" timestamp="2011-07-12T09:32:06.046" />
```

Se pueden producir los siguientes tipos de evento:

<trigger-start ...>
El inicio de un ciclo de recogida de basura, cuando la cantidad de memoria usada llega a ser mayor que el umbral del desencadenante. El umbral predeterminado es el 50% del almacenamiento dinámico. El atributo `intervalms` es el intervalo entre el suceso anterior `trigger end` (con id-1) y este suceso `trigger start`.

<trigger-end ...>
Un ciclo de recogida de basura ha reducido correctamente la cantidad de memoria utilizada por debajo del umbral desencadenante. Si un ciclo de recogida de basura ha terminado, pero la memoria utilizada no ha caído por debajo del umbral desencadenante, se iniciará un nuevo ciclo de recogida de basura con el mismo ID de contexto. Para cada suceso de tipo `trigger start`, existe un suceso `trigger end` coincidente con el mismo ID de contexto. El atributo `intervalms` es el intervalo entre el suceso `trigger start` anterior y el suceso `trigger end` actual. Durante este tiempo, se habrán completado una o varias recogidas de basura hasta que la memoria utilizada esté por debajo del umbral desencadenante.

<gc-op id="28" type="heartbeat"...>
Evento periódico que recopila información (sobre la memoria y el tiempo) acerca de todas las cantidades de la recogida de basura del tiempo que cubre. Un suceso de pulsación solo se puede producir entre un par coincidente de sucesos `trigger start` y `trigger end`; es decir, mientras un ciclo activo de recogida de basura está en curso. El atributo `intervalms` es el intervalo entre el suceso de pulsación anterior (con `id` -1) y este suceso de pulsación.

<gc-op id="29" type="syncgc"...>
Evento de recogida de basura síncrono (no determinante). Consulte el apartado "Recogidas de basura síncronas" en la página 69

Las etiquetas XML de este ejemplo tienen los significados siguientes:

<quanta ...>
Resumen de las pausas de cantidad durante el intervalo de pulsaciones, incluyendo la longitud de las pausas, en milisegundos.

<free-mem type="heap" ...>
Resumen de la cantidad de espacio libre de almacenamiento dinámico durante el intervalo de pulsaciones, con la muestra tomada al final de cada cantidad de recogida de basura.

<classunload-info classloadersunloaded=9 classesunloaded=156 />
Número de cargadores de clase y clases descargadas durante el intervalo de pulsaciones.

<references type="weak" cleared="660 />
Número y tipo de objetos de referencia de Java borrados durante el intervalo de pulsaciones.

Nota:

- Si solo se ha producido una cantidad de recogida de basura en el intervalo entre dos pulsaciones, se toma una muestra de la memoria libre solo al final de esta cantidad. Por lo que las cantidades mínima, máxima y media que se ofrecen en el resumen de la pulsación son todas iguales.
- El intervalo entre dos sucesos de pulsación podría ser significativamente mayor que el tiempo de ciclo especificado sei el almacenamiento dinámico no está suficientemente completo como para precisar de la actividad de recogida de basura. Por ejemplo, si su programa necesita una actividad de recogida de basura solo una vez cada pocos segundos, es probable que vea una pulsación solo una vez cada pocos segundos.
- Es posible que el intervalo sea significativamente mayor que el tiempo del ciclo especificado porque la recogida de basura no tiene trabajo en un almacenamiento dinámico que no está lo suficientemente lleno como para

garantizar la actividad de la recogida de basura. Por ejemplo, si su programa necesita una actividad de recogida de basura solo una vez cada pocos segundos, es probable que vea una pulsación solo una vez cada pocos segundos.

Si se produce un suceso, como la recogida de basura síncrona o un cambio en la prioridad, los detalles del suceso y todos los sucesos pendientes, como pulsaciones, se generan como salida de inmediato.

- Si la cantidad máxima de la recogida de basura para un periodo determinado es demasiado grande, es posible que desee reducir la utilización objetivo utilizando la opción **-Xgc:targetUtilization**. Esta acción proporciona a la recogida de basura más tiemp para trabajar. Como alternativa, es posible que quiera aumentar el tamaño de almacenamiento dinámico con la opción **-Xmx**. Del mismo modo, si su aplicación puede tolerar retardos más largos de los que se están notificando actualmente, puede aumentar la utilización objetivo o reducir el tamaño de almacenamiento dinámico.
- La salida se puede redirigir a un archivo de registro en lugar de a la consola con la opción **-Xverbosegclog:<archivo>**; por ejemplo, **-Xverbosegclog:out** escribe la salida **-verbose:gc** en el archivo *out*.
- La prioridad de la lista de `thread-priority` es la prioridad de la hebra del sistema operativo subyacente, no una prioridad de hebra Java.

Recogidas de basura síncronas

También se graba una entrada en el registro **-verbose:gc** cuando se produce una recogida de basura síncrona (no determinante). Este suceso tiene tres causas posibles:

- Una llamada `System.gc()` explícita en el código.
- La JVM se queda sin memoria y ejecuta una recogida de basura síncrona para evitar una condición de tipo OutOfMemoryError.
- La JVM se apaga durante una recogida de basura continua. La JVM no puede cancelar la recogida, por lo que completa la recogida de forma síncrona y luego sale.

Este es un ejemplo de una entrada `System.gc()`:

```
<gc-op id="9" type="syncgc" timems="12.92" contextid="8" timestamp=
"2011-07-12T09:41:40.808">
  <syncgc-info reason="system GC" totalBytesRequested="260" exclusiveaccessTimeMs="0.009"
  threadPriority="11" />
  <free-mem-delta type="heap" bytesBefore="22085440" bytesAfter="136023450" />
  <classunload-info classloadersunloaded="54" classesunloaded="234" />
  <references type="soft" cleared="21" dynamicThreshold="29" maxThreshold="32" />
  <references type="weak" cleared="523" />
  <finalization enqueued="124" />
</gc-op>
```

Este es un ejemplo de una entrada de recogida de basura síncrona como resultado del cierre de la JVM:

```
<gc-op id="24" type="syncgc" timems="6.439" contextid="19" timestamp="2011-07-12T09:43:14.524">
  <syncgc-info reason="VM shut down" exclusiveaccessTimeMs="0.009" threadPriority="11" />
  <free-mem-delta type="heap" bytesBefore="56182430" bytesAfter="151356238" />
  <classunload-info classloadersunloaded="14" classesunloaded="276" />
  <references type="soft" cleared="154" dynamicThreshold="29" maxThreshold="32" />
  <references type="weak" cleared="53" />
  <finalization enqueued="34" />
</gc-op>
```

Los atributos y etiquetas XML de este ejemplo tienen los significados siguientes:

<gc-op id="9" type="syncgc" timems="6.439" ...
La línea indica que el tipo de evento es una recogida de basura síncrona. El atributo `timems` es la duración de la recogida de basura síncrona, en milisegundos.

<syncgc-info reason="..."/>
El motivo de la recogida de basura síncrona.

<free-mem-delta.../>
Memoria de almacenamiento dinámico de Java libre antes y después de la recogida de basura síncrona en bytes.

<finalization .../>
Número de objetos pendientes de finalización.

<classunload-info .../>
Número de cargadores de clase y clases descargadas durante el intervalo de pulsaciones.

<references type="weak" cleared="53" .../>
Número y tipo de objetos de referencia de Java borrados durante el intervalo de pulsaciones.

La recogida de basura síncrona debida a condiciones de falta de memoria o al cierre de la máquina virtual solo se puede producir si el recopilador de basura está activo. Tiene que ir precedida por un suceso `trigger start`, aunque no es necesario que sea de inmediato. Algunos sucesos de pulsación pueden ocurrir entre un suceso `trigger start` y el suceso `synchgc`. La recogida de basura síncrona causada por `System.gc()` se puede producir en cualquier momento.

Rastreo de todas las cantidades del recopilador de basura

Las cantidades individuales del recopilador de basura se pueden rastrear si se habilitan los puntos de rastreo `GlobalGCStart` y `GlobalGCEnd`. Estos puntos de rastreo se producen al principio y al final de toda la actividad del recopilador de basura Metronome, incluidas las recogidas de basura síncronas. La salida de estos puntos de rastreo tendrá un aspecto similar al siguiente:

```
03:44:35.281 0x833cd00 j9mm.52 - GlobalGC start: globalcount=3

03:44:35.284 0x833cd00 j9mm.91 - GlobalGC end: workstackoverflow=0 overflowcount=0
```

Entradas de falta de memoria

Cuando falta espacio libre en el almacenamiento dinámico, se graba una entrada en el registro **`-verbose:gc`** antes de que se genere la excepción OutOfMemoryError. Este es un ejemplo de esta salida:

```
<out-of-memory id="71" timestamp="2011-07-12T10:21:50.135" memorySpaceName="Metronome"
  memorySpaceAddress="0806DFDC"/>
```

De forma predeterminada, el volcado Java se produce como resultado de una excepción OutOfMemoryError. Este volcado contiene información sobre las de memoria que utiliza su programa.

```
NULL
1STSEGTOTAL    Total memory:           4066080 (0x003E0B20)
1STSEGINUSE    Total memory in use:    3919440 (0x003BCE50)
1STSEGFREE     Total memory free:       146640 (0x00023CD0)
```

Comportamiento del recopilador de basura Metronome en condiciones de falta de memoria:

De forma predeterminada, el recopilador de basura Metronome desencadena una recogida de basura ilimitada y no determinante cuando la JVM se queda sin memoria. Para evitar un comportamiento no determinante, utilice la opción **-Xgc:noSynchronousGCOnOOM** para generar una excepción OutOfMemoryError cuando la JVM se quede sin memoria.

La recogida ilimitada predeterminada se ejecuta hasta que se recoge toda la basura posible en una única operación. El tiempo de pausa necesario suele ser mucho mayor que en una cantidad incremental normal de Metronome.

Información relacionada:

Utilización de -Xverbose:gc para analizar las recogidas de basura síncronas

Comportamiento del recopilador de basura Metronome en llamadas System.gc() explícitas:

Si hay un ciclo de recogida de basura en curso, el recopilador de basura Metronome completa el ciclo de manera síncrona cuando se llama a `System.gc()`. Si no hay ningún ciclo de recogida de basura en curso, se realiza un ciclo síncrono completo cuando se llama a `System.gc()`. Utilice `System.gc()` para limpiar el almacenamiento dinámico de forma controlada. Se trata de una operación no determinante porque realiza una recogida de basura completa antes del retorno.

Algunas aplicaciones llaman al software del proveedor que tiene llamadas System.gc() cuando no es aceptable crear estos retardos no determinantes. Para inhabilitar todas las llamadas System.gc(), utilice la opción **-Xdisableexplicitgc**.

La salida detallada de la recogida de basura para una llamada System.gc() tiene una razón de "recogida de basura del sistema" y es probable que su duración sea larga:

```
<gc-op id="9" type="syncgc" timems="6.439" contextid="8" timestamp="2011-07-12T09:41:40.808">
  <syncgc-info reason="VM shut down" exclusiveaccessTimeMs="0.009" threadPriority="11"/>
  <free-mem-delta type="heap" bytesBefore="126082300" bytesAfter="156085440"/>
  <classunload-info classloadersunloaded="14" classesunloaded="276"/>
  <references type="soft" cleared="154" dynamicThreshold="29" maxThreshold="32"/>
  <references type="weak" cleared="53"/>
  <finalization enqueued="34"/>
</gc-op>
```

Datos de diagnóstico de clases compartidas

Entender cómo diagnosticar problemas que puedan producirse le ayudará a utilizar la modalidad de clases compartidas.

La Guía de usuario de IBM SDK para Java V7 contiene instrucciones útiles sobre el diagnóstico de problema con clases compartidas, y cubre:

- Despliegue de clases compartidas
- Gestión de la modificación del código de bytes en tiempo de ejecución
- Información sobre actualizaciones dinámicas
- Utilización de la API del ayudante Java
- Descripción de la salida de diagnóstico de clases compartidas
- Depuración de problemas con clases compartidas

Puede encontrar la información aquí: IBM SDK para Java 7 - Datos de diagnóstico de clases compartidas .

Uso de JVMTI

JVMTI es una interfaz bidireccional que permite la comunicación entre la JVM y un agente nativo. Sustituye a las interfaces JVMDI y JVMPI.

JVMTI permite a terceros desarrollar herramientas de depuración, definición de perfiles y supervisión para la JVM. La interfaz contiene mecanismos para que el agente notifique a la JVM acerca del tipo de información que requiere. La interfaz también proporciona un medio para recibir las notificaciones importantes. Se pueden conectar varios agentes a una JVM en cualquier momento.

La Guía de usuario de IBM SDK para Java V7 contiene información detallada sobre el uso de JVMTI, incluyendo una sección de referencia de API en ampliaciones de IBM para JVMTI.

Puede encontrar dicha información aquí: IBM SDK for Java 7 - Using JVMTI.

Uso de la Diagnostic Tool Framework for Java

La Diagnostic Tool Framework for Java (DTFJ) es una interfaz de programación de aplicación (API) Java de IBM que se utiliza para dar soporte a la creación de herramientas de diagnósticos de Java. DTFJ funciona con datos de un volcado del sistema o un volcado Java.

La Guía del usuario de IBM SDK para Java V7 contiene información detallada sobre DTFJ. Siga este enlace:Uso de Diagnostic Tool Framework para Java

Capítulo 10. Referencia

Este conjunto de temas muestra las opciones y las bibliotecas de clases que se pueden utilizar con WebSphere Real Time for Linux

Opciones de línea de mandatos

Puede especificar opciones en la línea de mandatos mientras inicia Java. Las opciones por omisión se han seleccionado para el mejor uso general.

Especificación de opciones Java y propiedades del sistema

Existen tres maneras de especificar las propiedades Java y las propiedades del sistema.

Acerca de esta tarea

Puede especificar las opciones Java y las propiedades del sistema de las siguientes formas. En orden de prioridad, éstos modos son:

1. Especificando la opción o propiedad en la línea de mandatos. Por ejemplo:

   ```
   java -Dmysysprop1=tcpip -Dmysysprop2=wait -Xdisablejavadump MyJavaClass
   ```

2. Al crear un archivo que contenga las opciones y especificar dicho archivo en la línea de mandatos con la opción **-Xoptionsfile=**<*nombre_archivo*>.

   En el archivo de opciones, especifique cada opción en una nueva línea; puede utilizar el carácter '\' como carácter de continuación si desea que una única opción abarque varias líneas. Utilice el carácter "#" para definir líneas de comentarios. No puede especificar **-classpath** en un archivo de opciones. He aquí un ejemplo de archivo de opciones:

   ```
   #My options file
   -X<option1>
   -X<option2>=\
   <value1>,\
   <value2>
   -D<sysprop1>=<value1>
   ```

3. Creando una variable de entorno denominada **IBM_JAVA_OPTIONS** que contenga las opciones. Por ejemplo:

   ```
   export IBM_JAVA_OPTIONS="-Dmysysprop1=tcpip -Dmysysprop2=wait -Xdisablejavadump"
   ```

La última opción que especifique en la línea de mandatos tiene preferencia sobre la primera. Por ejemplo, si especifica las opciones **-Xint -Xjit myClass**, la opción **-Xjit** prevalece sobre **-Xint**.

Propiedades del sistema

Las propiedades del sistema están a disposición de las aplicaciones y ayudan a proporcionar información sobre el entorno de ejecución.

com.ibm.jvm.realtime
: Esta propiedad habilita a las aplicaciones de Java para determinar si se están ejecutando en un entorno de WebSphere Real Time for Linux.

  Si su aplicación se está ejecutando en el entorno de ejecución de IBM WebSphere Real Time for RT Linux y se ha iniciado con la opción **-Xrealtime**, la propiedad **com.ibm.jvm.realtime** tendrá el valor “hard”.

Si su aplicación se ejecuta en el entorno de ejecución de IBM WebSphere Real Time for RT Linux, pero no se ha iniciado con la opción **-Xrealtime**, no se define la propiedad **com.ibm.jvm.realtime**.

Si su aplicación se está ejecutando en el entorno de ejecución de IBM WebSphere Real Time, la propiedad **com.ibm.jvm.realtime** tendrá el valor "soft".

Opciones estándar

Las definiciones de las opciones estándar.

-agentlib:<*nombre biblioteca*>**[=**<*opciones*>**]**
Carga la biblioteca nativa del agente <*nombre biblioteca*>; por ejemplo **-agentlib:hprof**. Para obtener más información, especifique **-agentlib:jdwp=help** y **-agentlib:hprof=help** en la línea de mandatos.

-agentpath:*nombre de biblioteca***[=**<*opciones*>**]**
Carga la biblioteca nativa del agente mediante el nombre de vía de acceso completo.

-assert Imprime ayuda sobre opciones relacionadas con aserciones.

-cp or -classpath <*directorios y archivos .zip or .jar separados por* **:**>
Establece la vía de acceso de búsqueda de clases y recursos de aplicaciones. Si no se utilizan **-classpath** y **-cp** y no está establecida **CLASSPATH**, la classpath del usuario es, de forma predeterminada, el directorio activo (.).

-D<nombre_propiedad>=<*valor*>
Establece una propiedad de sistema.

-help o -?
Imprime un mensaje de uso.

-javaagent:<*vía de acceso de jar*>**[=**<*opciones*>**]**
Carga un agente de lenguaje de programación Java. Para obtener más información, consulte la documentación sobre la API java.lang.instrument.

-jre-restrict-search
Incluye los JRE privados en la búsqueda de versión.

-no-jre-restrict-search
Excluye los JRE privados en la búsqueda de versión.

-showversion
Imprime la versión del producto y prosigue.

-verbose:[*clase***,***gc***,***dynload***,***tamaños***,***pila***,***jni***]**
Habilita la salida verbosa.

-verbose:clase
Graba una entrada en stderr para cada clase cargada.

-verbose:gc
Consulte "Utilización de la información de verbose:gc" en la página 66.

-verbose:dynload
Proporciona información detallada a medida que la JVM carga cada clase, entre ella:
- El nombre de clase y paquete
- Para archivos de clase que estén en un archivo `.jar`, el nombre y la vía de acceso del directorio del `.jar`

- Detalles del tamaño de la clase y el tiempo que tarda en cargarse la clase

Los datos se escriben fuera de stderr. Un ejemplo de la salida es el siguiente:

```
<Loaded java/lang/String from /myjdk/sdk/jre/lib/i386/
softrealtime/jclSC160/vm.jar>
<Class size 17258; ROM size 21080; debug size 0>
<Read time 27368 usec; Load time 782 usec; Translate time 927 usec>
```

Nota: Las clases cargadas desde la memoria caché de clase compartida no aparecen en la salida de **-verbose:dynload**. Utilice **-verbose:class** para obtener información sobre estas clases.

-verbose:tamaños
Escribe información en stderr que describe la cantidad de memoria utilizada para pilas y volcados en la JVM

-verbose:pila
Escribe información en stderr que describe el uso de la pila C y Java.

-verbose:jni
Graba información en stderr que describe los servicios JNI que invocan la aplicación y la JVM.

-version
Imprime la información de la versión para la modalidad que no es de tiempo real.

-version:<*valor*>
Requiere que se ejecute la versión especificada.

-X Imprime ayuda sobre las opciones no estándar.

Opciones no estándar

Las opciones con el prefijo **-X** no son estándar, y están sujetas a cambios sin aviso.

La Guía del usuario de IBM SDK para Java V7 contiene información detallada sobre opciones que no son estándar. Puede encontrar la información aquí: IBM SDK para Java 7 - Opciones de la línea de mandatos.

En las secciones siguientes se proporciona información adicional para IBM WebSphere Real Time for Linux.

Opciones del recopilador de basura Metronome

Definiciones de las opciones del recopilador de basura Metronome.

-Xgc:synchronousGCOnOOM | -Xgc:nosynchronousGCOnOOM
Una de las ocasiones en las que se produce la recogida de basura es cuando el almacenamiento dinámico se queda sin memoria. Si no hay más espacio libre en el almacenamiento dinámico, utilizar **-Xgc:synchronousGCOnOOM** detiene su aplicación mientras la recogida de basura elimina los objetos no utilizados. Si el espacio libre se agota de nuevo, puede reducir la utilización objetivo para así dejar más tiempo a la recogida de basura para finalizar. Definir **-Xgc:nosynchronousGCOnOOM** implica que cuando la memoria de almacenamiento dinámico está llena, su aplicación se detiene y emite un mensaje de falta de memoria. El valor predeterminado es **-Xgc:synchronousGCOnOOM**.

-Xnoclassgc
Inhabilita la recogida de basura de clases. Esta opción desactiva la recogida de basura del almacenamiento asociado con las clases Java que ya no utiliza la JVM. El comportamiento predeterminado es **-Xnoclassgc**.

-Xgc:targetPauseTime=*N*
Establece el tiempo de pausa de la recogida de basura, donde *N* es el tiempo, en milisegundos. Cuando se especifica esta opción, GC funciona con pausas que no sobrepasan el valor especificado. Si no se especifica esta opción, el tiempo de pausa predeterminado se establece en 3 milisegundos. Por ejemplo, la ejecución con **-Xgc:targetPauseTime=20** hace que GC no se detenga más de 20 milisegundos durante operaciones de GC.

-Xgc:targetUtilization=*N*
Define la utilización de la aplicación en N%; el recopilador de basura intenta utilizar como máximo (100-N)% de cada intervalo de tiempo. Los valores razonables se encuentran en el rango 50-80%. Las aplicaciones con una tasa de asignación baja se pueden ejecutar al 90%. El valor predeterminado es 70%.

En este ejemplo se muestra que el tamaño máximo de la memoria de almacenamiento dinámico es 30 MB. El recopilador de basura trata de utilizar el 25% de cada intervalo de tiempo porque la utilización objetivo de la aplicación es del 75%.

```
java -Xgcpolicy:metronome -Xmx30m -Xgc:targetUtilization=75 Test
```

-Xgc:threads=*N*
Especifica el número de hebras del recopilador de basura que se van a ejecutar. El valor predeterminado es el número de núcleos de procesador disponibles para el proceso. El valor máximo que puede especificar es el número de procesadores a disposición del sistema operativo.

-Xgc:verboseGCCycleTime=N
N es el tiempo en milisegundos durante el que se debe volcar la información de resumen.

Nota: El tiempo de ciclo no implica que la información de resumen se vuelque exactamente a dicha hora, sino cuando pasa el último suceso de recogida de basura que satisface este criterio temporal.

-Xmx<*tamaño*>
Especifica el tamaño del almacenamiento dinámico de Java. Al contrario que otras estrategias de recogida de basura, la recogida de basura Metronome en tiempo real no admite la expansión de almacenamiento dinámico. No hay una opción de tamaño de almacenamiento dinámico máximo o inicial. Solo puede especificar el tamaño de almacenamiento dinámico máximo.

Valores predeterminados para la JVM

Se aplican los valores predeterminados a JVM de tiempo real si no se realizan cambios en el entorno en el que se ejecuta la JVM. Se muestran los valores comunes para utilizar cómo referencia.

Los valores predeterminados pueden cambiarse mediante variables de entorno o parámetros de línea de mandatos en el inicio de la JVM. La tabla muestra algunos de los valores comunes de la JVM. La última columna indica cómo cambiar el comportamiento cuando son aplicables las siguientes claves:

- **e**: valor controlado solo por variable de entorno
- **c**: valor controlado solo por parámetro de línea de mandatos
- **ec** - valor controlado por variable de entorno y por parámetro de línea de mandatos, pero tiene prioridad este último.

La información se proporciona como una referencia rápida y no está detallada.

Valor de JVM	Valor predeterminado	Valor afectado por
Volcados Java	Habilitado	ec
Volcados Java en falta de memoria	Habilitado	ec
Volcados de almacenamiento dinámico	Inhabilitado	ec
Volcados de almacenamiento dinámico en falta de memoria	Habilitado	ec
Sysdumps	Habilitado	ec
Donde se producen los archivos de volcado	Directorio actual	ec
Salida detallada	Inhabilitado	c
Búsqueda de classpath de arranque	Inhabilitado	c
Comprobaciones JNI	Inhabilitado	c
Depuración remota	Inhabilitado	c
Comprobaciones de cumplimiento estricta	Inhabilitado	c
Inicio rápido	Inhabilitado	c
Servidor de información de depuración remoto	Inhabilitado	c
Señalización reducida	Inhabilitado	c
Encadenamiento de manejador de señal	Habilitado	c
Classpath	No establecida	ec
Compartimiento de datos de clases	Inhabilitado	c
Soporte de accesibilidad	Habilitado	e
compilador JIT	Habilitado	ec
Compilador AOT (la JVM no utiliza AOT a menos que estén también habilitadas las clases compartidas)	Habilitado	c
Opciones de depuración de JIT	Inhabilitado	c
Tamaño máximo Java2D de fonts con negrita algorítmica	14 puntos	e
Mapas de bits representados de uso de Java2D en fonts escalables	Habilitado	e
Trazado de font de tipo libre Java2D	Habilitado	e
Fonts AWT de uso de Java2D	Inhabilitado	e
Entorno local predeterminado	Ninguno	e
Tiempo de espera antes de iniciar plugin	cero	e
Directorio temporal	/tmp	e
Redirección de plugin	Ninguno	e
Conmutación IM	Inhabilitado	e
Modificadores IM	Inhabilitado	e
Modelo de hebra	N/D	e

Valor de JVM	Valor predeterminado	Valor afectado por
Tamaño de pila inicial para hebras Java de 32 bits. Utilice: **-Xiss<tamaño>**	2 KB	c
Tamaño de pila máximo para hebras Java de 32 bits. Utilice: **-Xss<tamaño>**	256 KB	c
Tamaño de pila para hebras de sistema operativo de 32 bits. Utilice **-Xmso<tamaño>**	256 KB	c
Tamaño de almacenamiento dinámico inicial. Utilice **-Xms<tamaño>**	64 MB	c
Tamaño de almacenamiento dinámico Java máximo. Utilice **-Xmx<tamaño>**	Mitad de la memoria disponible con un mínimo de 16 MB y un máximo de 512 MB	c
Uso del intervalo de tiempo de destino para una aplicación. El colector de basura intenta utilizar el resto. Utilice **-Xgc:targetUtilization=<porcentaje>**	70%	c
El número de hebras de colector de basura para ejecutar. Utilice **-Xgc:threads=<valor>**	El número de núcleos de procesadores disponibles para el proceso	c
Cantidad máxima de memoria que puede asignarse a memorias de ámbito en la modalidad **-Xrealtime**. Utilice **-Xgc:scopedMemoryMaximumSize=<tamaño>**.	8 MB	c
Establece el tamaño de un área de memoria inmortal en la modalidad **-Xrealtime**. Utilice **-Xgc:immortalMemorySize=<tamaño>**	16 MB	c

Nota: "tamaño disponible" es la cantidad real (física) de memoria o el valor de **RLIMIT_AS**, el que sea más pequeño.

Avisos

Esta información se ha desarrollado para productos y servicios que se ofrecen en los EE.UU. IBM puede que no ofrezca los productos, servicios o características que se discuten en este documento en otros países. Consulte a su representante local de IBM para obtener información acerca de los productos y servicios actualmente disponibles en su zona. Cualquier referencia a un producto, programa o servicio de IBM no pretende afirmar ni implica que solo se pueda utilizar dicho producto, programa o servicio de IBM. En su lugar, puede utilizarse cualquier producto, programa o servicio funcionalmente equivalente que no infrinja ninguno de los derechos de propiedad intelectual de IBM. Sin embargo, la evaluación y la verificación del funcionamiento conjuntamente con otros productos, programas o servicios que no son de IBM son responsabilidad del usuario.

IBM puede tener aplicaciones patentadas o pendientes de patente que cubran el tema tratado en este documento. La posesión de este documento no le otorga ninguna licencia sobre estas patentes. Puede enviar consultas de licencias por escrito a la dirección siguiente:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
EE.UU.

Para consultas de licencias relacionadas con la información de doble byte (DBCS), póngase en contacto con el Departamento de la propiedad intelectual de IBM de su país o envíe las consultas por escrito a la dirección siguiente:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan Ltd.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, Japón

El párrafo siguiente no se aplica al Reino Unido ni a ningún otro país en el que tales disposiciones entren en contradicción con la legislación nacional:

INTERNATIONAL BUSINESS MACHINES CORPORATION PROPORCIONA ESTA PUBLICACIÓN "TAL CUAL" SIN GARANTÍA DE NINGUNA CLASE, YA SEA EXPLICITA O IMPLÍCITA, INCLUIDAS, PERO SIN LIMITARSE A ELLAS, LAS GARANTÍAS IMPLÍCITAS DE NO VULNERACIÓN, COMERCIALIZACIÓN O IDONEIDAD PARA UN PROPÓSITO DETERMINADO. Algunos estados no contemplan la limitación de responsabilidades, ni implícitas ni explícitas, en determinadas transacciones, por lo que cabe la posibilidad de que esta declaración no sea aplicable en su caso.

Esta información puede contener imprecisiones técnicas o errores tipográficos. Periódicamente se realizarán modificaciones en la información aquí contenida; dichos cambios se incorporarán en nuevas ediciones de la publicación. IBM puede efectuar mejoras y/o cambios en los productos y/o programas descritos en esta información en cualquier momento y sin previo aviso.

Cualquier referencia hecha en esta información a sitios web que no son de IBM se proporciona únicamente para su comodidad y no debe considerarse en modo alguno como promoción de dichos sitios web. Los materiales de estos sitios Web no forman parte de los materiales de IBM para este producto y el uso que se haga de estos sitios Web es de la entera responsabilidad del usuario.

IBM puede utilizar o distribuir la información que facilite de la manera que considere apropiada sin incurrir en obligaciones con el remitente.

Los poseedores de licencias de este programa que deseen obtener información acerca del mismo con el propósito de permitir (i) el intercambio de información entre programas creados independientemente y otros programas (incluido éste) y (ii) la utilización mutua de la información intercambiada, deben ponerse en contacto con la dirección siguiente:

- JIMMAIL@uk.ibm.com [Contacto de Hursley Java Technology Center (JTC)]

Esta información estará disponible, sujeta a los términos que correspondan, lo que en algunos casos supondrá el pago de una cuota.

IBM proporciona el programa con licencia descrito en este documento y todo el material con licencia disponible para el mismo bajo los términos del Acuerdo de cliente de IBM, el Acuerdo internacional de licencia de programas IBM o cualquier acuerdo equivalente entre las dos partes.

Cualquier información acerca del rendimiento que contenga el presente documento se ha determinado en un entorno controlado. Por lo tanto, los resultados obtenidos en otros entornos operativos pueden variar significativamente. Es posible que algunas medidas se hayan tomado en sistemas de nivel de desarrollo y no hay garantías de que estas medidas vayan a ser iguales en los sistemas habitualmente disponibles. Asimismo, algunas mediciones pueden haber sido estimadas mediante extrapolación. Los resultados reales pueden variar. Los usuarios de este documento deben verificar los datos aplicables para su entorno específico.

La información relativa a productos que no son de IBM se ha obtenido de los proveedores de estos productos, de los anuncios públicos o de otras fuentes disponibles públicamente. IBM no ha probado esos productos y no puede confirmar la precisión del rendimiento, su compatibilidad o cualquier otra afirmación relacionada con productos que no son de IBM. Las preguntas sobre las posibilidades de los productos que no son de IBM deben dirigirse a los proveedores de dichos productos.

Consideraciones acerca de la política de privacidad

Los productos de software de IBM, que incluyen software como soluciones de servicio (“Ofertas de software”), pueden utilizar cookies u otras tecnologías para recopilar información de uso de los productos, para ayudar a mejorar la experiencia final del usuario, para personalizar las interacciones con el usuario final o para otros fines. En muchos casos, las Ofertas de software no recopilan información de identificación personal. Algunas de nuestras Ofertas de software pueden ayudarle a recopilar información de identificación personal. Si la oferta de software utiliza cookies para recopilar información de identificación personal, se establece a continuación información específica sobre el uso de cookies de esta oferta.

Esta oferta de software no utiliza cookies u otras tecnologías para recopilar información de identificación personal.

Si las configuraciones desplegadas para esta oferta de software le ofrecen como cliente la posibilidad de recopilar información de identificación personal de los usuarios finales mediante cookies y otras tecnologías, debe buscar asesoramiento jurídico sobre la legislación aplicable a esa recopilación de datos, que incluye cualquier requisito de aviso y consentimiento.

Para obtener más información sobre el uso de las diferentes tecnologías, incluidas las cookies, para estos fines, consulte: (i) la política de privacidad de IBM, en http://www.ibm.com/privacy; (ii) La Declaración de privacidad en línea de IBM, enhttp://www.ibm.com/privacy/details (en concreto, la sección titulada “Cookies, balizas web y otras tecnologías”); (iii) la “Declaración de privacidad de productos de software y software como servicio de IBM”, en http://www.ibm.com/software/info/product-privacy.

Marcas registradas

IBM, el logotipo de IBM e ibm.com son marcas registradas o marcas comerciales registradas de International Business Machines Corporation en los Estados Unidos y/o en otros países. Si estos u otros términos que son marcas registradas de IBM se marcan como tales la primera vez que aparecen en esta información con un símbolo de marca registrada (® o ™), estos símbolos indican que son marcas registradas de los Estados Unidos o marcas registradas de uso comercial propiedad de IBM en el momento en que se publicó esta información. Estas marcas registradas también pueden ser marcas registradas o marcas registradas de uso comercial en otros países. Existe una lista actual de marcas registradas de IBM en la web, bajo el epígrafe "Información de copyright y marcas registradas" en la página web: http://www.ibm.com/legal/copytrade.shtml.

Adobe, el logotipo de Adobe, PostScript y el logotipo de PostScript son marcas registradas o marcas comerciales de Adobe Systems Incorporated en Estados Unidos o en otros países.

Intel e Itanium son marcas registradas de Intel Corporation o sus filiales en los Estados Unidos y/o en otros países.

Linux es una marca registrada de Linus Torvalds en los Estados Unidos y/o en otros países.

Java y todas las marcas comerciales y logotipos de Java son marcas registradas o marcas comerciales registradas de Oracle y/o sus filiales.

Otros nombres de empresas, productos o servicios pueden ser marcas registradas o de servicio de otros.

Índice

Caracteres Especiales

A

B

C

D

E

F

G

H

I

J

L

M

N

O

P

R

S

U

V

Impreso en España