IBM WebSphere Real Time for Linux
Versió 3

Guia de l'usuari

IBM WebSphere Real Time for Linux
Versió 3

Guia de l'usuari

Nota

Abans d'utilitzar aquesta informació i el producte que suporta, llegiu la informació de "Avisos" a la pàgina 79.

Cinquena edició (febrer de 2014)

Aquesta edició de la guia de l'usuari fa referència al producte IBM WebSphere Real Time for Linux, Versió 3, i a totes les versions i modificacions posteriors fins que s'indiqui d'una altra manera en futures edicions.

Contingut

Figures

Taules

Prefaci

Aquesta guia de l'usuari proporciona informació general sobre l'IBM® WebSphere Real Time for Linux.

Capítol 1. Introducció

Aquesta informació descriu l'IBM WebSphere Real Time for Linux.

Qualsevol modificació nova que es faci a aquesta guia de l'usuari s'indicarà amb barres verticals a l'esquerra dels canvis.

Trobareu la informació més recent sobre el IBM WebSphere Real Time for Linux que no estigui disponible a la guia de l'usuari a: http://www.ibm.com/support/docview.wss?uid=swg21501145

Visió general del WebSphere Real Time for Linux

El WebSphere Real Time for Linux inclou capacitats en temps real amb la màquina virtual IBM J9 (JVM).

El WebSphere Real Time for Linux és un JRE (Java™ Runtime Environment) amb un kit de desenvolupament de programari (SDK) que amplia l'IBM SDK for Java capacitats en temps real. Les aplicacions que depenen d'uns temps de resposta precisos poden aprofitar les característiques en temps real que ofereix el WebSphere Real Time for Linux en tecnologia Java estàndard.

Característiques

Les aplicacions en temps real necessiten un temps d'execució coherent en lloc d'una velocitat absoluta.

Tot seguit s'indiquen les preocupacions principals en desplegar aplicacions en temps real amb JVM tradicionals:
- Retards imprevisibles (possiblement llargs) de l'activitat de recollida de deixalles (GC).
- Retards a l'execució del mètode quan es produeix una compilació a temps (JIT) i una recompilació, amb la variabilitat en el temps d'execució.
- Planificació arbitrària del sistema operatiu.

El WebSphere Real Time for Linux proporciona el següent per eliminar aquests obstacles:
- El Recollidor de deixalles Metronome, un recollidor de deixalles determinista incremental amb temps de pausa molt curts.

Novetats

Aquest tema presenta els canvis de l'IBM WebSphere Real Time for Linux.

WebSphere Real Time for Linux V3

El WebSphere Real Time for Linux V3 és una extensió de l'IBM SDK for Java V7, que es basa en les característiques i funcions disponibles en aquesta versió i hi

afegeix capacitats en temps real. Les versions anteriors del WebSphere Real Time for Linux estaven basades en versions anteriors de l'IBM SDK for Java.

Per obtenir més informació sobre les novetats de l'IBM SDK for Java V7, consulteu: Novetats a l'Information Center de l'IBM SDK for Java 7.

Qualsevol modificació nova que es faci a aquesta guia de l'usuari s'indicarà amb barres verticals a l'esquerra dels canvis.

Planificació de fils de Java amb polítiques de Linux

A partir de l'actualització 1 del servei, podeu planificar fils regulars de Java amb la política de planificació **SCHED_RR** per ajustar aplicacions en temps real. Per obtenir més informació, consulteu "Planificació de fils" a la pàgina 6.

Control dels temps de pausa del recollidor de deixalles Metronome

Per defecte, el recollidor de deixalles Metronome fa una pausa de 3 mil•lisegons entre cicles de recollida de deixalles. Podeu canviar aquest valor per controlar el temps de pausa mitjançant una opció nova de la línia d'ordres. Per obtenir més informació sobre aquesta opció, consulteu "Control del temps de pausa" a la pàgina 23.

Referències comprimides

Ara el recollidor de deixalles Metronome admet tant referències no comprimides com referències comprimides en plataformes de 64 bits. Per veure les implicacions en el rendiment, consulteu Capítol 7, "Rendiment", a la pàgina 31.

Avantatges

Els avantatges de l'entorn en temps real són que les aplicacions Java s'executen amb un grau superior de predicció que amb la JVM estàndard i proporcionen un comportament de temporització constant per a l'aplicació Java. Les activitats de segon pla, com ara la compilació i la recollida de deixalles, es produeixen en moments determinats i, per tant, eliminen els pics inesperats d'activitat de segon pla quan s'executa l'aplicació.

Aquests avantatges s'obtenen ampliant la JVM amb la tecnologia de recollida de deixalles en temps real Metronome.

Accessibilitat

Les característiques d'accessibilitat permeten als usuaris amb una discapacitat, com ara una mobilitat restringida o una visió limitada, utilitzar productes de tecnologia de la informació de manera satisfactòria.

IBM s'esforça per oferir productes amb accés a tothom, independentment de l'edat o la capacitat.

Per exemple, podeu utilitzar el WebSphere Real Time for Linux sense ratolí, només amb el teclat.

Per llegir sobre problemes que afectin l'accessibilitat del IBM SDK for Java V7 subjacent, consulteu l'IBM Information Center. No hi ha cap problema d'accessibilitat que afecti les característiques in les funciones úniques del WebSphere Real Time for Linux.

Navegació mitjançant el teclat

Aquest producte utilitza les tecles de navegació estàndards del Microsoft Windows.

Per als usuaris que requereixin la navegació per teclat, trobareu una descripció de les pulsacions per les aplicacions Swing: Vinculacions de tecles Swing.

IBM i accessibilitat

Consulteu l'IBM Human Ability and Accessibility Center per obtenir més informació sobre el compromís d'IBM amb l'accessibilitat.

Capítol 2. Descripció de l'IBM WebSphere Real Time for Linux

En aquest apartat es presenten els components clau de l'IBM WebSphere Real Time for Linux.

- "Introducció al recollidor de deixalles Metronome"

Introducció al recollidor de deixalles Metronome

El recollidor de deixalles Metronome substitueix el recollidor de deixalles estàndard al WebSphere Real Time for Linux.

La diferència clau entre la recollida de deixalles Metronome i la recollida de deixalles estàndard és que la primera es produeix en petits passos que es poden interrompre, mentre que la segona atura l'aplicació mentre marca i recull deixalles.

Per exemple:

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

L'exemple especifica que la vostra aplicació s'executa al 80% cada 60 ms. El 20% del temps que queda pot utilitzar-se per a la recollida de deixalles, si n'hi ha. El recollidor de deixalles Metronome garanteix nivells d'ús sempre que tingui prou recursos. La recollida de deixalles s'inicia quan la quantitat d'espai lliure a l'emmagatzematge dinàmic és inferior al llindar determinat de manera dinàmica.

Recollida de deixalles Metronome i baixada de classes

Metronome admet baixar classes de la mateixa manera que un kit de desenvolupador Java estàndard. Això no obstant, a causa de la feina implicada, mentre es baixen classes pot ser que es produeixin valors atípics de temps de pausa durant les activitats de recollida de deixalles.

Fils de recollidor de deixalles Metronome

El recollidor de deixalles Metronome té dos tipus de fil: un únic fil d'alarma i diversos fils de recollida (GC). Per defecte, el recollidor de deixalles utilitza un fil per a cada processador actiu lògic disponible per al sistema operatiu. Això permet assolir els processament paral•lel més eficient durant els cicles de recollida de deixalles. Un cicle de recollida de deixalles vol dir el temps entre que s'activa la recollida de deixalles i la finalització d'alliberament de deixalles. En funció de la mida de l'emmagatzematge dinàmic Java, el temps transcorregut per a un cicle de recollida de deixalles complet podria suposar diversos segons. Un cicle de recollida de deixalles sol tenir centenars de quàntums de recollida de deixalles. Aquests quàntums són pauses molt curtes per al codi d'aplicació, que normalment duren 3 mil•lisegon. Utilitzeu **-verbose:gc** per obtenir informes de resum dels cicles i els quàntums. Per obtenir més informació, vegeu "Utilització de la informació verbose:gc" a la pàgina 66. Podeu definir el nombre de fils de recollida de deixalles per a la JVM mitjançant l'opció **-Xgcthreads**.

No suposa cap avantatge incrementar el valor de **-Xgcthreads** per sobre del valor per defecte. Reduir el valor de **-Xgcthreads** pot reduir la càrrega de CPU general durant els cicles de recollida de deixalles, tot i que aquests cicles seran més llargs.

Nota: La durada dels quàntums de recollida de deixalles és constant a 3 mil•lisegons.

No podeu canviar el nombre de fils d'alarma per a la JVM.

El recollidor de deixalles Metronome verifica periòdicament la JVM per veure si la memòria de l'emmagatzematge dinàmic té prou espai lliure. Quan la quantitat d'espai lliure és inferior al límit, el recollidor de deixalles Metronome activa la JVM per iniciar la recollida de deixalles.

Fil d'alarma
El fil d'alarma únic garanteix l'ús d'un nombre mínim de recursos. S'"activa" de manera periòdica i fa aquestes comprovacions:
- La quantitat d'espai lliure a la memòria d'emmagatzematge dinàmic
- Si s'està executant la recollida de deixalles

Si no hi ha prou espai lliure disponible i no s'està executant la recollida de deixalles, el fil d'alarma activa els fils de recollida per iniciar la recollida de deixalles. El fil d'alarma no fa res fins a la propera hora planificada per verificar la JVM.

Fils de recollida
Els fils de recollida executen la recollida de deixalles.

Quan el cicle de recollida de deixalles ha finalitzat, el recollidor de deixalles Metronome verifica la quantitat d'espai d'emmagatzematge dinàmic lliure. Si encara no hi ha prou espai d'emmagatzematge dinàmic lliure, s'inicia un altre cicle de recollida de deixalles amb el mateix ID d'activador. Si ni ha prou espai d'emmagatzematge dinàmic lliure, l'activador finalitza i els fils de recollida de deixalles s'aturen. El fil d'alarma segueix supervisant l'espai d'emmagatzematge dinàmic lliure i activarà un altre cicle de recollida de deixalles quan sigui necessari.

Per obtenir més informació sobre la utilització del recollidor de deixalles Metronome, consulteu "Utilització del recollidor de deixalles Metronome" a la pàgina 23.

Planificació de fils

Les polítiques de planificació del Linux es poden fer servir amb fils de Java normals per ajustar aplicacions en temps real.

Amb el WebSphere Real Time for Linux, podeu executar fils Java normals amb la política de planificació SCHED_RR. L'ús de la política SCHED_RR us atorga més control sobre la vostra aplicació, que pot millorar el rendiment en temps real dels fils Java. La JVM detecta la prioritat i la política del fil principal quan Java s'inicia amb la política SCHED_RR. La JVM altera les assignacions de prioritat i política conseqüentment. Per obtenir més informació sobre l'alteració de prioritats i polítiques de fils de Java normals, consulteu "Planificació i distribució de fils" a la pàgina 19

Les polítiques de planificació del Linux són:

SCHED_OTHER
Política de planificació en temps compartit universal per defecte que s'utilitza a la majoria de fils. Aquests fils han de tenir assignada una prioritat zero.

SCHED_OTHER utilitza la porció de temps, la qual cosa vol dir que cada fil s'executa durant un període de temps determinat després del qual el fil es pot executar.

SCHED_FIFO
Només se pot utilitzar amb prioritats superiors a zero. Quan un fil SCHED_FIFO passa a estar disponible, el fil té una prioritat més alta que qualsevol altre fil SCHED_OTHER normal.

Si un fil SCHED_FIFO que té una prioritat més alta passa a estar disponible, aquest fil té prioritat sobre els fils SCHED_FIFO existents amb prioritat més baixa. En aquest cas, aquest fil es manté a la part superior de la cua per la seva prioritat.

No hi ha porció de temps.

Nota: SCHED_FIFO no s'utilitza al WebSphere Real Time for Linux.

SCHED_RR
És una millora de SCHED_FIFO. La diferència és que cada fil només es pot executar durant un temps limitat. Si el fil sobrepassa aquest temps, es retorna a la llista per la seva prioritat. El WebSphere Real Time for Linux V3pot utilitzar SCHED_RR.

Per obtenir més informació sobre aquestes polítiques de planificació del Linux, vegeu la pàgina **man** per a **sched_setscheduler**.

Per obtenir més informació sobre com es fan servir les polítiques de planificació del Linux camb elWebSphere Real Time for Linux, consulteu “Planificació i distribució de fils” a la pàgina 19.

Capítol 3. Planificació

Llegiu aquest apartat abans d'instal•lar l'WebSphere Real Time for Linux.

•
• “Entorns admesos”
•
• “Consideracions” a la pàgina 10

Migració

L' Podeu executar les aplicacions Java estàndard al WebSphere Real Time for Linux sense necessitat de modificar-les.

Entorns admesos

L'IBM WebSphere Real Time for Linux s'admet en determinades plataformes de maquinari i sistemes operatius.

IBM WebSphere Real Time for Linux

Les arquitectures de plataforma admeses són:

- Arquitectura Intel, 32 bits (IA-32)
  - Pentium 4
  - Pentium Xeon
  - Pentium M
  - Pentium D i equivalents
- AMD64/EM64T
- IBM POWER 32
- IBM POWER 64

Nota: El maquinari Pentium 3 ja no s'admet.

Els sistemes operatius admesos són:

Taula 1. Entorns Linux provats

Maquinari	**IA-32 de 32 bits**	**AMD64/EM64T de 64 bits**	
Espai d'adreces de l'SDK	**32 bits**	**32 bits**	**64 bits**
RHEL 5 Actualitz. 7	Sí	Sí	Sí
RHEL 6 Actualitz. 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: SLES 9, SLES 10 i RHEL 4 no s'admeten.

Consideracions

Heu de tenir present diversos factors quan utilitzeu el WebSphere Real Time for Linux.

- Quan sigui possible, no executeu més d'una JVM en temps real al mateix sistema. Això s'explica perquè aleshores tindríeu diversos recollidors de deixalles. Cada JVM no coneix les àrees de memòria de l'altra. Un efecte és que els cicles de recollida de deixalles i els temps de pausa no es poden coordinar entre les JVM, cosa que significa que és possible que una JVM afecti negativament el rendiment de la recollida de deixalles d'una altra JVM. Si heu d'utilitzar diverses JVM, assegureu-vos que cada JVM estigui vinculada a un subconjunt específic de processadors utilitzant l'ordre **taskset**.
- Les memòries cau compartides utilitzades en versions anteriors del WebSphere Real Time for Linux per emmagatzemar codi i classes precompilats no són compatibles amb les memòries cau utilitzades en aquesta versió del WebSphere Real Time for Linux. Heu de regenerar el contingut de les memòries cau anteriors.
- Quan s'utilitzen memòries cau de classes compartides, el nom de la memòria cau no pot tenir més de 53 caràcters.

Capítol 4. Instal•lació del WebSphere Real Time for Linux

Seguiu aquests passos per instal•lar el producte.

Fitxers d'instal•lació

Tot seguit s'indiquen els fitxers d'instal•lació necessaris.

L'IBM WebSphere Real Time for Linux es proporciona en dos tipus de paquets InstallAnywhere.

Paquets instal•lables
Els paquets instal•lables configuren el sistema. Per exemple, els programes poden definir variables d'entorn.

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

Paquets d'arxiu
Aquests paquets extreuen els fitxers al sistema, però no realitzen cap configuració.

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

Nota: <arch> és l'arquitectura de plataforma; x86_32 o x86_64.

Instal•lació des d'un fitxer InstallAnywhere

Aquests paquets proporcionen un programa interactiu que us orientarà a través de les opcions d'instal•lació. Podeu executar el programa com a interfície gràfica de l'usuari o des de la consola del sistema.

Abans de començar

El vostre sistema ha de tenir les biblioteques compartides següents:

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

Si no teniu la biblioteca compartida `libstdc++.so.5`, podeu veure un abocament de memòria del nucli de Java quan instal•leu, amb els errors següents:

```
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 esteu instal•lant un paquet instal•lable, heu de tenir l'eina rpm-build instal•lada al sistema, en cas contrari el programa d'instal•lació no pot registrar el paquet nou a la base de dades de RPM. Per saber si l'eina rpm-build està instal•lada, introduïu l'ordre següent:

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

Quant a aquesta tasca

Els paquets InstallAnywhere tenen una extensió de fitxer `.bin`.

Hi ha dos tipus de paquet:

Instal•lable
La instal•lació d'aquests paquets permet també configurar el sistema, per exemple definint variables d'entorn.

Arxiu La instal•lació d'aquests paquets extrau els fitxers al sistema, però no realitza cap configuració.

Procediment

- Per instal•lar el paquet de forma interactiva, completeu una instal•lació assistida.
- Per instal•lar el paquet sense cap interacció addicional de l'usuari, completeu una instal•lació desassistida. Podeu triar aquesta opció si voleu instal•lar diversos sistemes.
- Quan el procés d'instal•lació hagi finalitzat, seguiu els passos de configuració d'aquesta secció, com ara la definició de les variables d'entorn path i classpath.

Resultats

El producte està instal•lat.

Nota: No interrompeu el procés d'instal•lació, per exemple prement Ctrl+C. Si interrompeu el procés, possiblement haureu de tornar a instal•lar el producte. Per obtenir-ne més informació, consulteu “Instal•lació interrompuda” a la pàgina 14.

Si esteu utilitzant un paquet instal•lable, possiblement veureu missatges que us advertiran que s'ha trobat un problema. La instal•lació de paquets d'arxiu no produeix cap missatge. Alguns dels missatges que podríeu veure quan s'utilitza un paquet instal•lable es mostren a la llista següent:

The installer cannot run on your configuration. It will now quit.
Aquest missatge d'error es produeix quan l'ID d'usuari no té autorització per executar el procés d'instal•lació. Com que no pot continuar, finalitza el programa d'instal•lació. Per arreglar el problema, inicieu la instal•lació novament amb un ID d'usuari que tingui autoritat root.

An RPM package is already installed. Uninstall the package before proceeding.
Aquest missatge indica que ja hi ha un paquet RPM instal•lat. Com que no pot continuar, finalitza el programa d'instal•lació. Per arreglar el problema, desinstal•leu el paquet RPM abans de continuar.

Completar una instal•lació assistida

Instal•lació del producte des d'un paquet InstallAnywhere, de forma interactiva.

Abans de començar

Comproveu les condicions següents abans de començar amb el procés d'instal•lació:

- Si heu instal•lat prèviament el WebSphere Real Time for Linux des d'un paquet RPM, heu de desinstal•lar aquest paquet abans de continuar.
- Heu de tenir un ID d'usuari amb autoritat root.

Procediment

1. Descarregueu el fitxer del paquet d'instal•lació en un directori temporal.
2. Canvieu al directori temporal.
3. Inicieu el procés d'instal•lació escrivint `./paquet` en un indicador del shell, on *paquet* és el nom del paquet que esteu instal•lant.
4. Seleccioneu un idioma de la llista que es mostra a la finestra de l'instal•lador i posteriorment feu clic a **Next**. La llista d'idiomes disponibles es basa en la configuració local del sistema.
5. Llegiu l'acord de llicència, utilitzant la barra de desplaçament per arribar al final del text de la llicència. Per continuar amb la instal•lació heu d'acceptar els termes de l'acord de llicència. Per acceptar els termes, seleccioneu el botó d'opció i posteriorment feu clic a **OK**.

   Nota: No podeu seleccionar el botó d'opció per acceptar l'acord de llicència fins que hàgiu llegit fins el final el text de la llicència.
6. Se us sol•licitarà que escolliu el directori de destinació per a la instal•lació. Si no voleu fer la instal•lació en el directori de destinació, feu clic a **Choose** per seleccionar un directori alternatiu, utilitzant la finestra del navegador. Quan hàgiu escollit el directori d'instal•lació, feu clic a **Next** per continuar.
7. Se us sol•licitarà que reviseu les opcions que heu triat. Per canviar la selecció, feu clic a **Previous**. Si les opcions triades són correctes, feu clic a **Install** per continuar amb la instal•lació.
8. Quan el procés d'instal•lació hagi finalitzat, feu clic a **Done** per finalitzar.

Completar una instal•lació desassistida

Si heu d'instal•lar més d'un sistema i ja sabeu les opcions d'instal•lació que voleu utilitzar, és possible que vulgueu utilitzar el procés d'instal•lació desassistida. La instal•lació es fa una vegada utilitzant el procés d'instal•lació assistida, i posteriorment s'utilitza el fitxer de resposta per completar més instal•lacions sense cap interacció addicional de l'usuari.

Procediment

1. Creeu un fitxer de resposta completant una instal•lació desassistida. Utilitzeu una de les opcions següents:
   - Utilitzeu la GUI i especifiqueu que el programa d'instal•lació creï un fitxer de resposta. El fitxer de resposta s'anomena `installer.properties` i es crea al directori d'instal•lació.
   - Utilitzeu la línia d'ordres i afegiu l'opció -r a l'ordre d'instal•lació assistida, especificant el camí d'accés complet al fitxer de resposta. Per exemple:

     `./paquet -r /camí_accés/installer.properties`

Contingut del fitxer de resposta d'exemple:

```
INSTALLER_UI=silent
USER_INSTALL_DIR=/directori_personal
```

En aquest exemple, */directori_personal* és el directori d'instal•lació per defecte que escolliu per a la instal•lació.

2. Opcional: Si cal, editeu el fitxer de resposta per canviar les opcions.

   Nota: Els paquets d'arxiu tenen el següent problema conegut: les instal•lacions que utilitzen el fitxer de resposta utilitzen el directori per defecte fins i tot si canvieu el directori al fitxer de resposta. Si hi ha una instal•lació prèvia al directori per defecte, se sobreescriu.

   Si esteu creant més d'un fitxer de resposta, cadascun amb opcions d'instal•lació diferents, especifiqueu un nom únic per a cada fitxer de resposta, en el format *fitxer_personal*.properties.

3. Opcional: Genereu un fitxer de registre. Com que esteu fent una instal•lació silenciosament, no es visualitza cap missatge d'estat al final del procés d'instal•lació. Per generar un fitxer de registre que contingui l'estat de la instal•lació, completeu els passos següents:

   a. Definiu les propietat del sistema necessàries utilitzant l'ordre següent.

      ```
      export _JAVA_OPTIONS="-Dlax.debug.level=3 -Dlax.debug.all=true"
      ```

   b. Definiu la variable d'entorn següent per enviar la sortida del registre a la consola.

      ```
      export LAX_DEBUG=1
      ```

4. Inicieu una instal•lació desassistida executant l'instal•lador del paquet amb l'opció silenciosa **-i**, i l'opció **-f** per especificar el fitxer de resposta. Per exemple:

   ```
   ./paquet -i silent -f /camí_accés/installer.properties 1>console.txt 2>&1
   ./paquet -i silent -f /camí_accés/fitxer_personal.properties 1>console.txt 2>&1
   ```

   Podeu utilitzar un camí d'accés complet o un camí d'accés relatiu al fitxer de propietats. En aquests exemples, la cadena `1>console.txt 2>&1` redirigeix la informació del procés d'instal•lació de les cadenes stderr i stdout al fitxer de registre `console.txt` del directori actual. Reviseu aquest fitxer de registre si creieu que hi ha hagut algun problema amb la instal•lació.

   Nota: Si el directori d'instal•lació conté diversos fitxers de resposta, s'utilitza el fitxer de resposta per defecte, `installer.properties`.

Instal•lació interrompuda

Si l'instal•lador del paquet s'atura inesperadament durant la instal•lació, per exemple si premeu Ctrl+C, la instal•lació es malmetrà i no podreu desinstal•lar o tornar a instal•lar el producte. Si intenteu desinstal•lar o tornar a instal•lar, veureu un missatge de tipus `Error fatal de l'aplicació`.

Quant a aquesta tasca

Per solucionar aquest problema, suprimiu fitxers i torneu a instal•lar, com es descriu als passos següents:

Procediment

1. Suprimiu el fitxer de registre `/var/.com.zerog.registry.xml`.

2. Suprimiu el directori que conté la instal•lació, si s'ha creat. Per exemple, /opt/IBM/javawrt3[_64]/.
3. Executeu el procés d'instal•lació una altra vegada.

Limitacions i problemes coneguts

Els paquets InstallAnywhere tenen algunes limitacions i problemes coneguts.

- Si no teniu la biblioteca compartida `libstdc++.so.5` al vostre sistema, la instal•lació falla, provocant un abocament de memòria del nucli de Java. Per obtenir més informació, consulteu "Instal•lació des d'un fitxer InstallAnywhere" a la pàgina 11.
- La GUI del paquet d'instal•lació no suporta el programa de lectura de pantalla Orca. Podeu utilitzar el mode d'instal•lació desassistida com a alternativa a la GUI.
- Si, després de la instal•lació, introduïu `./paquet` per iniciar el programa novament, el programa visualitza el missatge següent:

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

  Si premeu l'Intro per acceptar el valor per defecte, el programa no respon. Escriviu un número i després premeu l'Intro.
- Si instal•leu el paquet i posteriorment intenteu instal•lar de nou en un mode diferent, per exemple en mode de consola o en mode silenciós, possiblement veureu el missatge d'error següent:

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

  No hauríeu de veure aquest missatge si heu instal•lat utilitzant el mode de GUI o esteu executant el programa d'instal•lació novament en mode de consola. Si veieu aquest error i esteu executant el programa per seleccionar l'opció de desinstal•lació (només els paquets instal•lables), utilitzeu com a alternativa l'ordre **`./_uninstall/uninstall`**, com es descriu a "Desinstal•lació del WebSphere Real Time for Linux" a la pàgina 18.

Només els paquets instal•lables

- No podeu actualitzar una instal•lació existent utilitzant els paquets InstallAnywhere. Per actualitzar el WebSphere Real Time for Linux, primer heu de desinstal•lar qualsevol versió anterior.
- No podeu instal•lar dues instàncies diferents de la mateixa versió del WebSphere Real Time for Linux al mateix sistema a la vegada, fins i tot si utilitzeu directoris d'instal•lació diferents. Per exemple, no podeu tenir simultàniament el WebSphere Real Time for Linux V3 al directori `/previous` i una instal•lació de renovació de servei del WebSphere Real Time for Linux al directori `/current`. El programa d'instal•lació comprova el número de versió. Si el programa troba un paquet existent amb el mateix número de versió, se us sol•licitarà que desinstal•leu el paquet existent.
- Si s'instal•la el paquet i executeu l'instal•lador del paquet utilitzant la GUI, podeu seleccionar la desinstal•lació del paquet. Aquesta opció de desinstal•lació no està disponible en mode desassistit. Si executeu novament l'instal•lador del paquet en mode desassistit, el programa s'executa però no realitza cap acció.

Només els paquets d'arxiu

- Si canvieu el directori d'instal•lació en un fitxer de resposta i posteriorment executeu una instal•lació desassistida utilitzant aquest fitxer de resposta, el

programa d'instal•lació ignora el directori d'instal•lació nou i utilitza com a alternativa el directori per defecte. Si hi ha una instal•lació prèvia al directori per defecte, se sobreescriu.

Definició del camí d'accés

Quan definiu la variable d'entorn **PATH**, podeu executar una aplicació o un programa escrivint-ne el nom en un indicador de l'intèrpret d'ordres.

Quant a aquesta tasca

Nota: Si modifiqueu la variable d'entorn **PATH** com es descriu en aquesta secció, sobreescriureu els executables Java que tingueu al camí d'accés.

Podeu especificar el camí d'accés a una eina escrivint cada vegada el camí d'accés abans del nom de l'eina. Per exemple, si l'SDK està instal•lat a `/opt/IBM/javawrt3[_64]/`, podeu compilar un fitxer anomenat *myfile.java* escrivint l'ordre següent a l'indicador de l'intèrpret d'ordres:

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

Per evitar haver d'escriure el camí d'accés complet cada vegada:

1. Editeu el fitxer d'inici de l'intèrpret d'ordres al directori inicial (normalment **.bashrc**, en funció de l'intèrpret d'ordres) i afegiu els camins d'accés absoluts a la variable d'entorn **PATH**; per exemple:

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

2. Torneu a iniciar la sessió o executeu l'script de l'intèrpret d'ordres actualitzat per activar la nova definició del **PATH**.
3. Compileu el fitxer amb l'eina **javac**. Per exemple, per compilar el fitxer *myfile.java*, introduïu el següent a l'indicador de l'intèrpret d'ordres:

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

   la variable d'entorn **PATH** permet al sistema operatiu Linux trobar fitxers executables, com ara **javac**, **java** i l'eina **javadoc**, des de qualsevol directori actual. Per visualitzar el valor actual del camí d'accés, escriviu aquesta ordre en un indicador d'ordres:

   ```
   echo $PATH
   ```

Què cal fer posteriorment

Vegeu "Definició de la variable classpath" per determinar si heu de definir la variable d'entorn CLASSPATH.

Definició de la variable classpath

La variable d'entorn **CLASSPATH** indica a les eines de l'SDK, com ara **java**, **javac**i **javadoc** on trobar les biblioteques de classes Java.

Quant a aquesta tasca

Definiu la variable d'entorn **CLASSPATH** de manera explícita només en una d'aquestes condicions:

- Necessiteu una biblioteca o un fitxer de classes diferent, com ara un que desenvolupeu, i no es troba al directori actual.

- Canvieu la ubicació dels directoris `bin` i `lib` i aquests ja no tenen el mateix directori pare.
- Teniu la intenció de desenvolupar o executar aplicacions mitjançant la utilització de diferents entorns de temps d'execució en el mateix sistema.

Per mostrar el valor actual del vostre **CLASSPATH**, introduïu l'ordre següent a indicador de l'intèrpret d'ordres:

```
echo $CLASSPATH
```

Si desenvolupeu i executeu aplicacions que utilitzen diferents entorns de temps d'execució, incloses altres versions que heu instal•lat separadament, heu d'establir **CLASSPATH** i **PATH** explícitament per a cada aplicació. Si executeu diverses aplicacions de manera simultània i utilitzeu entorns d'execució diferents, cada aplicació s'ha d'executar al seu propi intèrpret d'ordres.

Si només executeu una versió del Java alhora, podeu utilitzar un script de l'intèrpret d'ordres pe commutar entre els diferents entorns d'execució.

Què cal fer posteriorment

Vegeu "Prova de la instal•lació" per verificar que la instal•lació ha estat satisfactòria.

Prova de la instal•lació

Utilitzeu l'opció **-version** per verificar si la instal•lació ha estat satisfactòria.

Quant a aquesta tasca

La instal•lació Java consta d'una JVM en temps real.

Procediment

Seguiu aquests passos per provar la instal•lació:

1. Per veure informació sobre la versió per a la JVM en temps real, escriviu l'ordre següent a l'indicador de l'intèrpret d'ordres:

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

   Aquesta ordre retorna els missatges següents si és satisfactòria:

   ```
   java version "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 informació sobre la versió és correcta, però pot ser que l'arquitectura de plataforma i les dates siguin diferents que les de l'exemple. El format de la sèrie de data és: `yyyymmdd` seguit probablement per informació addicional específica del component.

Desinstal•lació del WebSphere Real Time for Linux

El procés que utilitzeu per eliminar el WebSphere Real Time for Linux depèn del tipus d'instal•lació que utilitzeu.

Abans de començar

Per als paquets instal•lables InstallAnywhere, heu de tenir un ID d'usuari amb autoritat root.

Quant a aquesta tasca

No hi ha cap procés de desinstal•lació per als paquets d'arxiu InstallAnywhere. Per eliminar un paquet d'arxiu del sistema, suprimiu el directori de destinació que escolliu quan instal•leu el paquet. Per als paquets instal•lables InstallAnywhere, desinstal•leu el producte utilitzant una ordre o executant de nou el programa d'instal•lació, com es descriu als passos següents.

Procediment

- Opcional: Desinstal•leu manualment utilitzant l'ordre **uninstall**.
  1. Canvieu al directori que conté la instal•lació. que crida l'atenció. Per exemple:
     ```
     cd /opt/IBM/javawrt3
     ```
  2. Inicieu el procés de desinstal•lació introduint l'ordre següent:
     ```
     ./_uninstall/uninstall
     ```
- Opcional: Si no podeu localitzar fàcilment el programa de desinstal•lació, com a alternativa podeu executar un altra instal•lació assistida. El programa d'instal•lació detecta que el producte ja està instal•lat, i posteriorment us dóna l'oportunitat de desinstal•lar la instal•lació anterior.

Capítol 5. Execució d'aplicacions de l'IBM WebSphere Real Time for Linux

Informació important d'ajuda quan executeu aplicacions en temps real.

Planificació i distribució de fils

El sistema operatiu Linux admet diverses polítiques de planificació. La política de planificació en temps compartit universal per defecte és SCHED_OTHER, que s'utilitza en la majoria de fils. Els fils de les aplicacions en temps real poden utilitzar SCHED_RR i SCHED_FIFO. El WebSphere Real Time for Linux només utilitza SCHED_OTHER i SCHED_RR.

El kernel decideix quin és el proper fil executable que el processador ha d'executar. El kernel manté una llista dels fils executables. Busca el fil que té la prioritat més alta i selecciona el fil com a proper fil que cal executar.

Amb l'ordre següent es pot obtenir una llista de les prioritats i les polítiques de fils:

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

on la política:

- `TS` és SCHED_OTHER
- `RR` és SCHED_RR
- `FF` és SCHED_FIFO
- `-` no té cap política notificada

La sortida és com la d'aquest exemple:

```
  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
```

Aquesta sortida mostra el procés Java i diversos fils amb la política SCHED_RR i la prioritat 6.

Per consultar la política de planificació actual, utilitzeu **`sched_getscheduler`** o l'ordre **`ps`** que es mostra a l'exemple.

Per obtenir informació sobre els processos, vegeu “Tècniques de depuració generals” a la pàgina 36.

Prioritats i polítiques de fils Java normals

Els fils Java normals, és a dir, els fils assignats com a objectes `java.lang.Thread`, utilitzen la política de planificació per defecte SCHED_OTHER. A partir de l'actualització 1 del servei del WebSphere Real Time for Linux V3, podeu executar fils Java normals amb la política de planificació SCHED_RR.

Per defecte, els fils Java s'executen utilitzant la política SCHED_OTHER per defecte. Aquesta política assigna fils Java a la prioritat 0 del sistema operatiu.

L'ús de la política SCHED_RR us atorga més control sobre la vostra aplicació, que pot millorar el rendiment en temps real dels fils Java. La JVM detecta la prioritat i la política del fil principal quan Java s'inicia amb la política SCHED_RR. La JVM altera les assignacions de prioritat i política conseqüentment. Tots els fils Java s'executen a la mateixa prioritat de sistema operatiu que el fil principal. Tot i que es poden assignar les prioritats 1 - 99 a SCHED_RR, les prioritats utilitzables de SCHED_RR per al WebSphere Real Time for Linux V3 són 1 - 10. Si la prioritat té un valor superior a 10, la prioritat del fil principal es disminueix a 10 i l'assignació s'aplica en base al valor 10.

Una manera de modificar la propietat de planificació en temps real d'un procés de la línia d'ordres és utilitzar l'ordre **`chrt`**. A l'exemple següent, la prioritat del fil Java principal es defineix de manera que utilitzi la planificació SCHED_RR, amb una prioritat del sistema operatiu 6.

```
chrt -r 6 java
```

Pot ser que hagueu de configurar el sistema perquè permeti modificar les prioritats. Per obtenir més informació, consulteu l'apartat “Configuració del sistema per permetre canvis de prioritat” a la pàgina 21.

Taula 2. Prioritats de Java i del sistema operatiu

Prioritat de Java	Valor de la prioritat de Java per al fil	Prioritat del sistema operatiu
1	MIN_PRIORITY	6
2		6
3		6
4		6
5	NORM_PRIORITY (valor per defecte)	6
6		6
7		6
8		6
9		6
10	MAX_PRIORITY	6

Tots els fils associats amb el fil Java principal s'executen a la mateixa prioritat del sistema operatiu.

Si executeu l'ordre `chrt -r 11 java`, el resultat és el mateix que si executeu l'ordre `chrt -r 10 java`. Això és així perquè no podeu aplicar una prioritat superior a 10

a l'assignació de prioritat utilitzada pels fils de la JVM, tot i que el fil que inicia la JVM i que espera la finalització de la JVM es manté a la prioritat 11.

La JVM produeix un missatge d'error si proveu d'utilitzar l'ordre `chrt -f 6 java`, perquè SCHED_FIFO no s'admet al WebSphere Real Time for Linux V3.

Per obtenir més informació sobre l'ordre **chrt**, consulteu http://publib.boulder.ibm.com/infocenter/lnxinfo/v3r0m0/index.jsp?topic=/liaai/realtime/liaairtchrt.htm.

Configuració del sistema per permetre canvis de prioritat

Per defecte, els usuaris no root del Linux no poden incrementar la prioritat d'un fil o procés. Podeu canviar la configuració del sistema per permetre canvis en la prioritat utilitzant el mòdul `pam_limits` dels mòduls PAM (Pluggable Authentication Module) per al Linux.

Si no podeu canviar la prioritat d'un fil o un procés mitjançant la utilitat **chrt**, normalment veureu aquest missatge:

```
sched_setscheduler: Operation not permitted
```

En kernels Linux recents, podeu canviar la configuració del sistema per permetre canvis de prioritat utilitzant el mòdul `pam_limits`. Aquest mòdul us permet configurar els límits dels recursos del sistema del fitxer de configuració de límits. El fitxer per defecte és `/etc/security/limits.conf`.

Una entrada al fitxer `/etc/security/limits.conf` té aquest format:

```
<domini> <tipus> <element> <valor>
```

on:

<domini> pot ser:

- un nom d'usuari del sistema que pot modificar els límits d'un recurs.
- un nom de grup, amb la sintaxis `@group`, els membres del qual poden modificar els límits d'un recurs.
- un comodí "`*`", que indica que qualsevol usuarui o grup pot modificar els límits d'un recurs.

<tipus> pot ser:

- `hard`, on el kernel aplica límits fixos.
- `soft`, on s'apliquen límits variables, que es poden alterar dintre de l'interval especificat pels límits fixos.
- un guionet "`-`", que indica límits fixos i variables.

<element> és:

- un recurs. Utilitzeu `rtprio` per a les prioritats en temps real.

<valor> és:

- un límit. Utilitzeu un valor de l'interval 1 - 100 per indicar el límit màxim per al valor de prioritat en temps real.

Per exemple:

```
* - rtprio 100
```

Permet que tots els usuaris canviïn la prioritat dels processos en temps real, utilitzant **chrt** o altres mecanismes.

Per defecte, l'usuari root pot incrementar les prioritats en temps real sense límits. Per aplicar un límit a root, l'usuari root s'ha d'especificar de manera explícita. Els límits de grup i comodí del fitxer de configuració no s'apliquen a l'usuari root.

Si especifiqueu límits d'usuari individual al fitxer, aquests límits tenen prioritat sobre els límits de grup.

Els canvis fets a `limits.conf` no són efectius de manera immediata. Heu de reiniciar els serveis afectats o reiniciar el sistema perquè un canvi de configuració sigui efectiu.

La capacitat d'incrementar aquesta prioritat en temps real d'una JVM (Java Virtual Machine) no està disponible als kernels del Linux 2.6.12 i anteriors. A la taula s'indica si hi ha suport disponible per a aquesta característica en algunes distribucions Linux comunes.

Taula 3. Suport per a canvis de prioritat en temps real

Distribució del Linux	**Versió del kernel Linux**	**Suport per a canvis de prioritat en temps real (sí/no)**
Red Hat Enterprise Linux (RHEL) 4	2.6.9	no
RHEL 5 i posteriors	2.6.18 i posteriors	sí
SUSE Linux Enterprise Server (SLES) 9	2.6.5-7	no
SLES 10 i posteriors	2.6.16 i posteriors	sí
Red Hat Enterprise MRG - Totes les versions	2.6.24 i posteriors	sí
SUSE Linux Enterprise Real Time (SLERT) - Totes les versions	2.6.16 i posteriors	sí
Ubuntu 5.10	2.6.12	no
Ubuntu 6.06 i posteriors	2.6.15 i posteriors	sí

Per habilitar els canvis de prioritat en un sistema Linux en temps real, podeu afegir un usuari al grup `realtime`, que es mostra al fitxer `limits.conf`.

Inici de processos secundaris

Els mètodes java.lang.Runtime.exec de l'API de màquina virtual Java (JVM) ofereixen a l'aplicació Java la capacitat d'executar una ordre en un procés independent.

Des d'aquesta crida de mètode, es crea un objecte java.lang.Process nou. L'objecte es pot utilitzar per controlar el procés nou o per obtenir-ne informació.

Amb aquest objectiu, els mètodes exec creen diversos fils. A l'IBM WebSphere Real Time for Linux, les modificacions del procediment fan possible un comportament més determinista en un entorn en temps real.

La crida Runtime.exec crea un fil "segador" per a cada subprocés bifurcat. El fil segador és l'únic fil que espera que el subprocés finalitzi. Quan el subprocés

finalitza, el fil segador en registra l'estat de sortida. El fil segador genera el nou procés i li assigna la mateixa prioritat que el fil que inicialment ha fet la crida Runtime.exec.

Si el procés generat és un altre JVM del WebSphere Real Time for Linux, i el mètode Runtime.exec ha estat cridat per un altre mètode que s'executa amb una política en temps real i una prioritat Linux, el fil principal de la nova màquina virtual assigna la seva política i prioritat a la mateixa política i prioritat en temps real de Linux. La prioritat d'aquest fil de Java és entre 1 i 10.

El fil segador també crea dos fils nous que escolten les seqüències `stdout` i `stderr` del nou procés. Les dades de `stdout` i `stderr` es desen en memòries intermèdies utilitzades per aquests fils. Les memòries intermèdies persisteixen més enllà del temps de vida del procés generat. Aquesta persistència permet que els recursos mantinguts pel procés generat s'esborrin immediatament quan finalitza el procés.

Utilització del recollidor de deixalles Metronome

El recollidor de deixalles Metronome substitueix el recollidor de deixalles estàndard al WebSphere Real Time for Linux.

Control del temps de pausa

El temps de pausa del recollidor de deixalles (GC) Metronome en pot ajustar per a cada procés Java.

Per defecte, el Recollidor de deixalles Metronome es posa en pausa durant 3 mil•lisegons en cada pausa individual, la qual cosa es coneix com a quantum. Un cicle complet de recollida de deixalles requereix moltes d'aquestes pauses, que es distribueixen per tal de donar suficient temps a l'aplicació per executar-se. Podeu canviar aquest valor temporal de pausa individual màxim amb l'opció **`-Xgc:targetPauseTime`**. Per exemple, l'execució amb **`-Xgc:targetPauseTime=20`** fa que el recollidor de deixalles funcioni amb pauses individuals no superiors als 20 mil•lisegons.

Les IBM Monitoring and Diagnostics Tools per a Java - Garbage Collection and Memory Visualizer (GCMV) es poden utilitzar per supervisar els temps de pausa del recollidor de deixalles per a la vostra aplicació, i per ajudar a diagnosticar i a ajustar els problemes de rendiment a l'aplicació Java. L'eina analitza i mostra dades de diversos tipus de registre, incloent:

- Registres de la recollida de deixalles detallada.
- Registres de recollida de deixalles amb traces, generats utilitzant el paràmetre **`-Xtgc`**.
- Registres de memòria nativa, generats utilitzant les ordres del sistema **`ps`**, **`svmon`** o **`perfmon`**.

Els gràfics d'aquesta secció els genera el GCMV, i mostren l'efecte derivat del canvi del temps de pausa de destinació en els cicles de recollida de deixalles. Cada gràfic mostra els temps de pausa reals entre els cicles de la recollida de deixalles Metronome (eix Y) i el temps d'execució d'una aplicació (eix X).

Nota: El GCMV admet un format més antic de recollida de deixalles. Si voleu analitzar la sortida de la recollida de deixalles detallada amb el GCMV, genereu la sortida amb l'opció **`-Xgc:verboseFormat=deprecated`**. Per obtenir-ne més informació, vegeu Opcions de línia d'ordres del recollidor de deixalles.

Amb el temps de pausa de destinació per defecte definit, el gràfic del temps de pausa de la recollida de deixalles detallada mostra que els temps de pausa es mantenen al voltant o per sota de la marca dels 3 mil•lisegons:

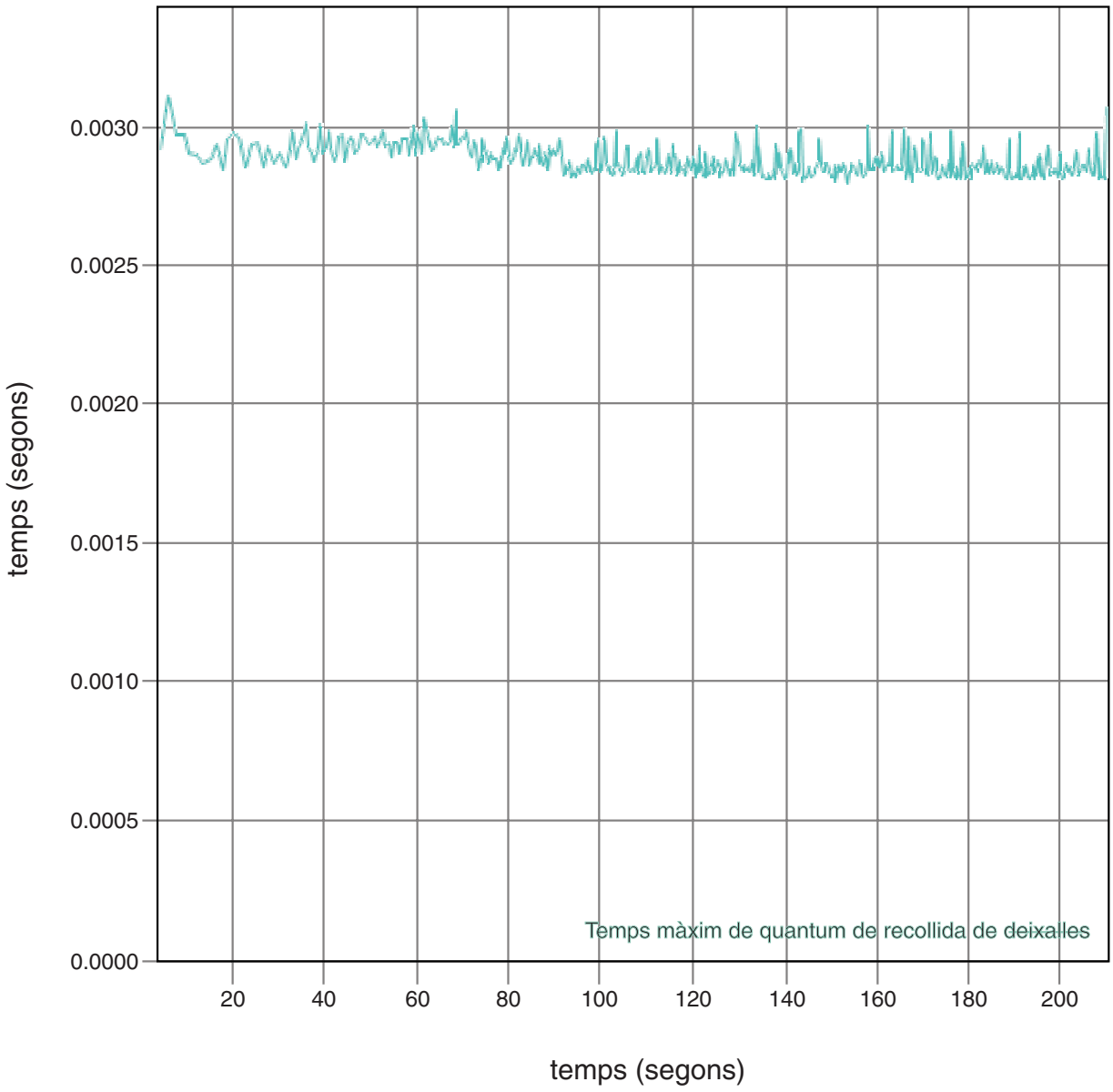

Figura 1. Temps de pausa de la recollida de deixalles reals quan el temps de pausa de destinació es defineix en el valor per defecte (3 mil•lisegons)

Amb un temps de pausa de destinació definit en 6 mil•lisegons, el gràfic del temps de pausa de la recollida de deixalles detallada mostra que els temps de pausa es mantenen al voltant o per sota de la marca dels 6 mil•lisegons:

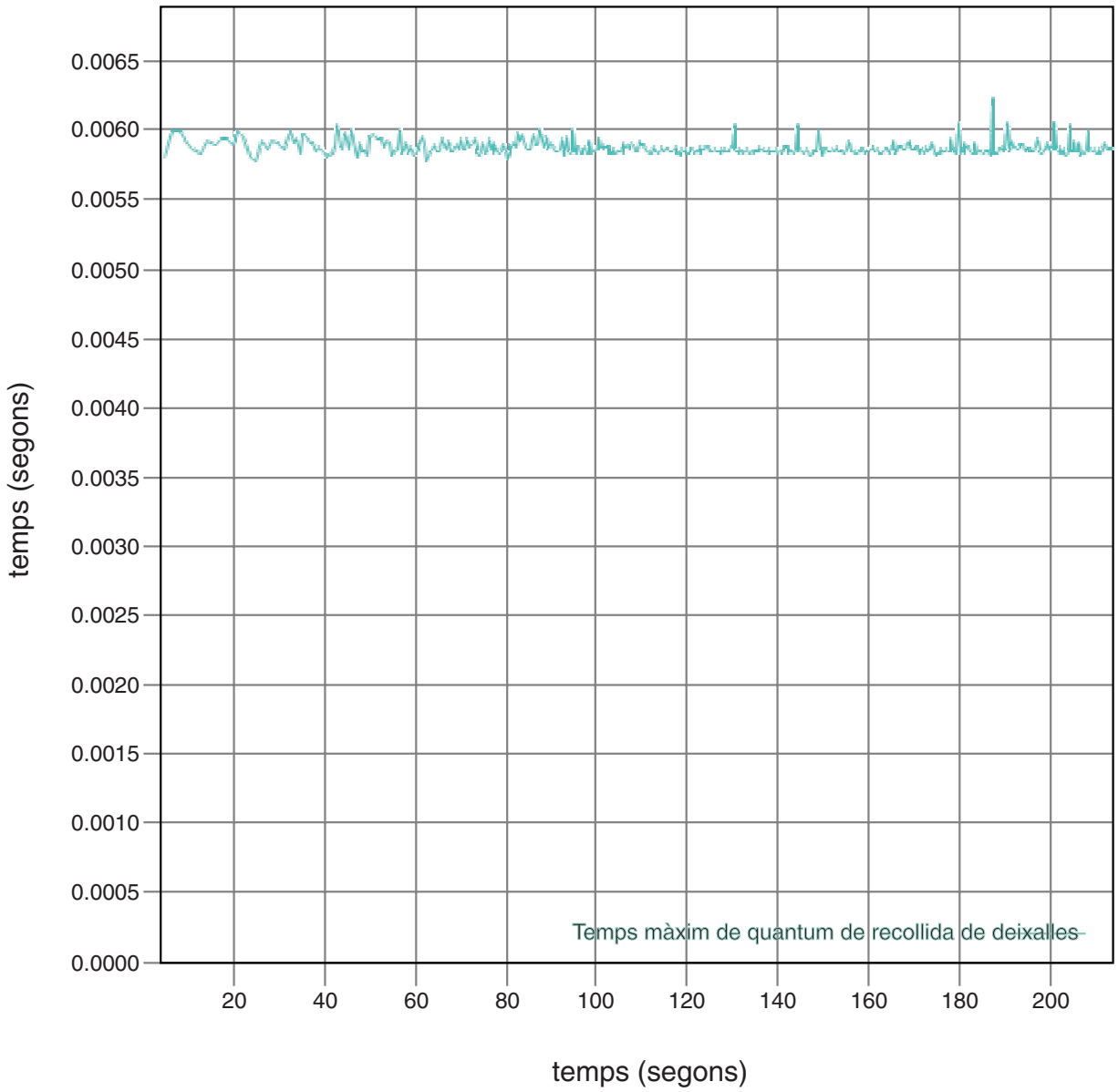

Figura 2. Temps de pausa reals quan el temps de pausa de destinació es defineix en 6 mil•lisegons

Amb un temps de pausa de destinació definit en 10 mil•lisegons, el gràfic del temps de pausa de la recollida de deixalles detallada mostra que els temps de pausa es mantenen al voltant o per sota de la marca dels 10 mil•lisegons:

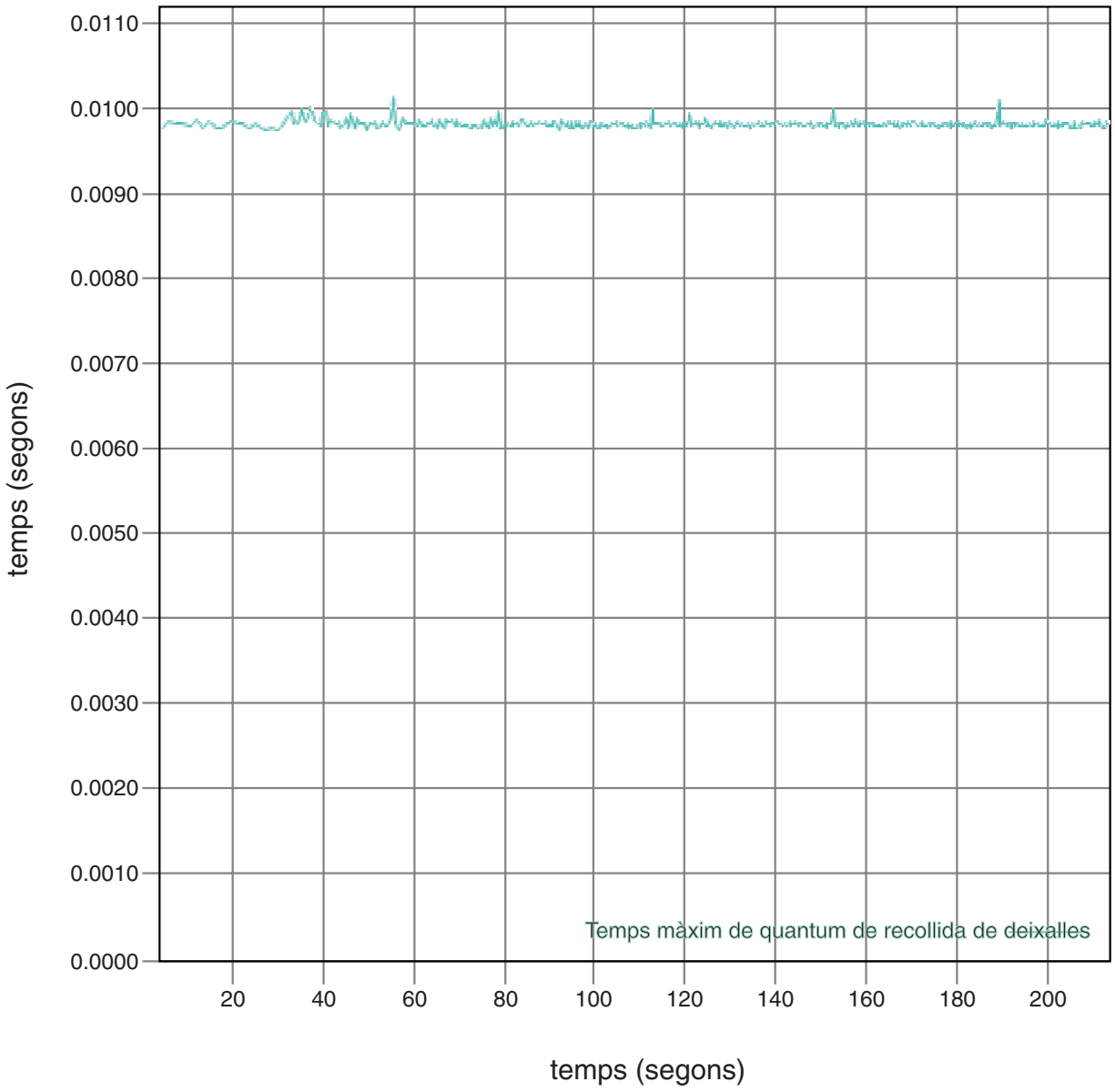

Figura 3. Temps de pausa reals quan el temps de pausa de destinació es defineix en 10 mil•lisegons

Amb un temps de pausa de destinació definit en 15 mil•lisegons, el gràfic del temps de pausa de la recollida de deixalles detallada mostra que els temps de pausa es mantenen al voltant o per sota de la marca dels 15 mil•lisegons:

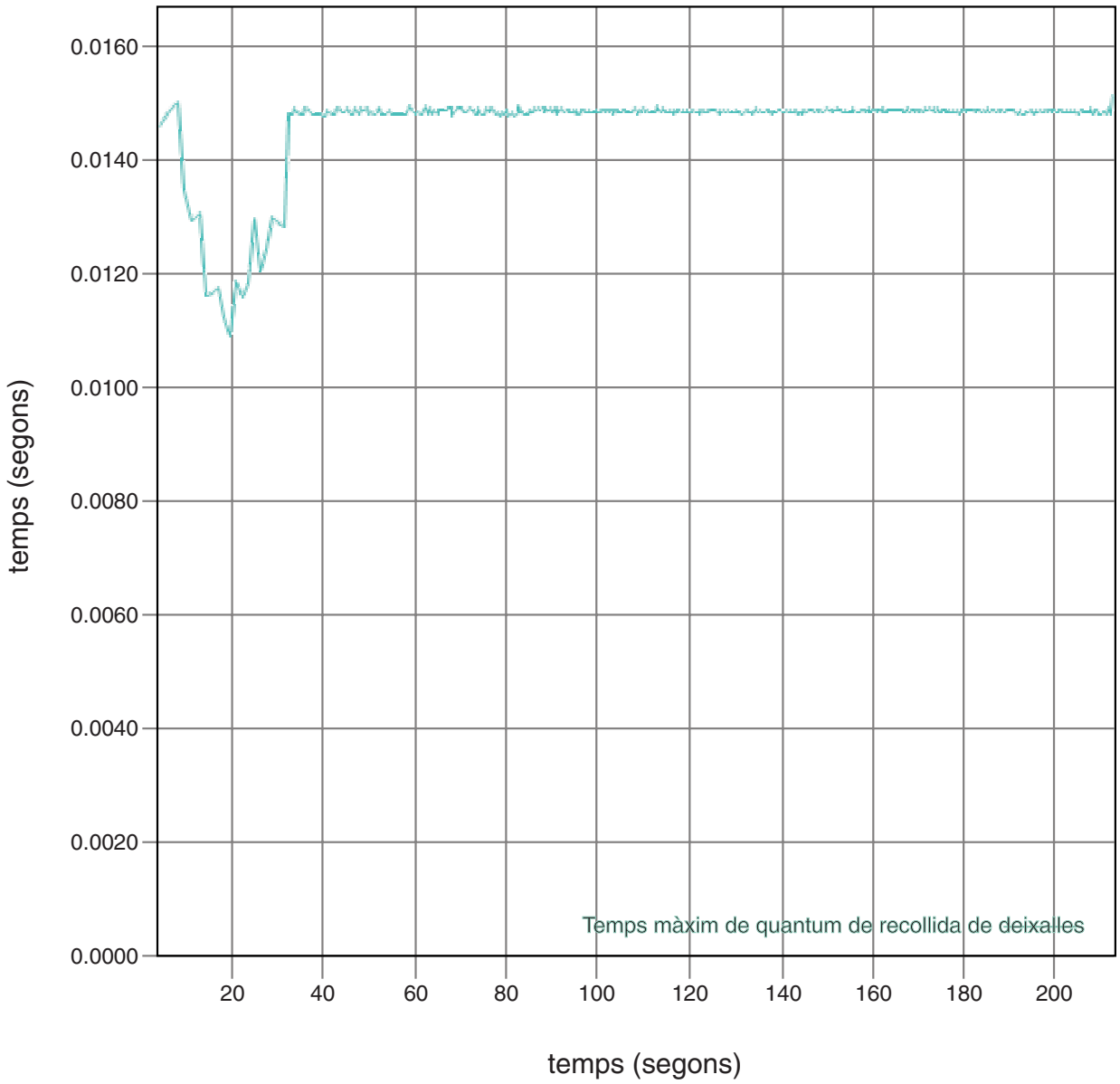

Figura 4. Temps de pausa reals quan el temps de pausa de destinació es defineix en 15 mil•lisegons

Control de la utilització del processador

Podeu limitar la quantitat de potència de processament disponible al Recollidor de deixalles Metronome.

Podeu controlar la recollida de deixalles amb el Recollidor de deixalles Metronome mitjançant l'opció **-Xgc:targetUtilization=N** per limitar la quantitat de CPU utilitzada pel recollidor de deixalles.

Per exemple:

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

L'exemple especifica que l'aplicació s'executa en un 80% cada 60 mil•lisegons. El 20% del temps que queda s'utilitza per a la recollida de deixalles. El Recollidor de deixalles Metronome garanteix nivells d'ús sempre que tingui prou recursos. La recollida de deixalles s'inicia quan la quantitat d'espai lliure a l'emmagatzematge dinàmic és inferior al llindar determinat de manera dinàmica.

Limitacions del recollidor de deixalles Metronome

Aquest tema descriu els problemes coneguts o les limitacions que afecten la política GC de Metronome.

Suport AESNI en plataformes x86

Actualment no s'admet l'explotació del software d'instruccions d'AESNI en arquitectures x86 amb la política del GC de Metronome.

Pauses de llarga durada durant la recopilació de dades corrompudes

En determinades circumstàncies, rarament, és possible que experimenteu unes pauses més llargues del que espereu durant la recollida de deixalles. Durant la recollida de deixalles, s'utilitza un procés d'exploració arrel. El recollidor de deixalles guia l'emmagatzematge dinàmic, començant per referències actives conegudes. Aquestes referències inclouen:

- Variables de referència actives a les piles de crides de fils actius.
- Referències estàtiques.

Per trobar totes les referències actives a objectes en una pila de fil d'aplicació, el recollidor de deixalles explora tots els marcs de pila de la pila de crides del fil. Cada pila de fils activa s'explora en un pas que no es pot interrompre. Això vol dir que l'exploració s'ha de fer dins d'un sol pause de recollida de deixalles.

L'efecte és que el rendiment del sistema pot ser pitjor del que s'esperava si teniu alguns fils amb piles molt profundes, a causa de les pauses llargues de recollida de deixalles al principi del cicle de recollida.

Capítol 6. Desenvolupament d'aplicacions

Informació important sobre com escriure aplicacions en temps real, inclosos exemples de codi.

- "Mapa hash en temps real d'exemple"

Mapa hash en temps real d'exemple

El WebSphere Real Time for Linux inclou les implementacions HashMap i HashSet que proporcionen un rendiment més coherent per al mètode put que el HashMap estàndard de l'IBM SDK per a Java 7.

El mapa java.util.HashMap estàndard que IBM proporciona funciona bé per a les aplicacions d'alt rendiment. També ajuda en les aplicacions que la mida màxima a la qual ha de créixer el seu mapa hash. Per a les aplicacions que necessiten un mapa hash que podria créixer a mides variables, en funció de l'ús, hi ha un possible problema de rendiment amb el mapa hash estàndard. El mapa hash estàndard proporciona bons temps de resposta per afegir noves entrades al mapa hash mitjançant el mètode put. Això no obstant, quan el mapa hash s'emplena, cal assignar un magatzem de suport més gran. Això vol dir que cal migrar les entrades del magatzem de suport actual. Si el mapa hash és gran, el temps per realitzar una operació put també podria ser llarg. Per exemple, l'operació podria trigar diversos mil•lisegons.

El WebSphere Real Time for Linux inclou un mapa hash en temps real. Proporciona la mateixa interfície funcional que el mapa java.util.HashMap estàndard, però permet un rendiment molt més coherent per al mètode put. En comptes de crear un magatzem de suport i migrar-hi totes les entrades quan s'emplena el mapa hash, el mapa hash d'exemple crea un magatzem de suport addicional. El nou magatzem de suport s'encadena amb la resta de magatzems de suport del mapa hash. Inicialment, la cadena causa una lleugera reducció del rendiment mentre el magatzem de suport buit s'assigna i encadena a la resta de magatzems de suport. Una vegada que s'ha actualitzat el mapa hash de suport, és més ràpid que haver d emigrar totes les entrades. Un dels inconvenients del mapa hash en temps real és que les operacions get, put i remove són lleugerament més lentes. Les operacions són més lentes perquè cada cerca ha de passar per un conjunt de mapes hash de suport en lloc de passar només per un.

Per provar el mapa hash en temps real, afegiu el fitxer `RTHashMap.jar` a l'inici del camí d'accés de classes d'arrencada. Si heu instal•lat el WebSphere Real Time for Linux al directori `$WRT_ROOT`, afegiu l'opció següent per utilitzar el mapa hash en temps real amb la vostra aplicació, en comptes d'utilitzar el mapa hash estàndard:

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

Els fitxers font i de classes per a la implementació del mapa hash en temps real s'inclouen al fitxer `demo/realtime/RTHashMap.jar`. A més, també es proporciona una implementació de java.util.LinkedHashMap i java.util.HashSet.

Capítol 7. Rendiment

El WebSphere Real Time for Linux està optimitzat per a pauses curtes i consistents de la recollida de deixalles, més que no pas per a la mitjana de rendiment més alta o l'impacte de memòria més insignificant.

Rendiment en configuracions de maquinari certificades

Els sistemes certificats tenen una granularitat de rellotge i una velocitat de processador suficients per fer possibles els objectius de rendiment del WebSphere Real Time for Linux. Per exemple, una aplicació ben escrita que s'executi en un sistema que no estigui sobrecarregat, i amb una mida d'emmagatzematge dinàmic adequada, generalment experimentaria temps de pausa de recollida de deixalles d'uns 3 mil•lisegons, i no més de 3,2 mil•lisegons. Durant els cicles de recollida de deixalles, una aplicació amb els valors d'entorn per defecte no s'atura durant més del 30% del temps transcorregut durant cap marge de temps variable de 60 mil•lisegons. El temps col•lectiu dedicat a les pauses de recollida de deixalles per sobre d'un període de 60 mil•lisegons normalment suma uns 18 mil•lisegons.

Reducció de la variabilitat de temps

Les fonts principals de variabilitat en una JVM estàndard són les pauses en la recollida de deixalles. Al WebSphere Real Time for Linux, les pauses possiblement llargues dels modes del recollidor de deixalles estàndard s'eviten mitjançant el Recollidor de deixalles Metronome. Consulteu “Utilització del recollidor de deixalles Metronome” a la pàgina 23.

Capacitat de compartir de dades de classes entre les JVM

La capacitat de compartir dades proporciona un mètode transparent de reduir l'impacte de memòria i millorar el temps d'inici de la JVM. Per obtenir més informació sobre la capacitat de compartir dades de classes, vegeu “Compartició de dades de classe entre les JVM” a la pàgina 32

Referències comprimides

El recollidor de deixalles Metronome suporta referències comprimides i no comprimides en plataformes de 64 bits. Quan s'utilitzen les referències comprimides, la JVM emmagatzema totes les referències a objectes, classes, fils i supervisors com a valors de 32 bits. L'ús de referències comprimides millora el rendiment de moltes aplicacions ja que els objectes són més petits i, per tant, la recollida de deixalles es realitza amb menys freqüència i millora la utilització de la memòria cau.

Nota: La mida de l'emmagatzematge dinàmic disponible per a referències comprimides està limitada a uns 28 GB.
Per obtenir més informació sobre la referències comprimides, vegeu http://publib.boulder.ibm.com/infocenter/java7sdk/v7r0/topic/com.ibm.java.lnx.70.doc/diag/understanding/mm_compressed_references.html.

Compartició de dades de classe entre les JVM

El suport per a classes compartides és igual quan s'executa amb, o sense, l'opció **-Xrealtime**.

Podeu compartir dades de classe entre màquines JVM (Java Virtual Machine) emmagatzemant-les en una memòria cau de la memòria compartida. La compartició redueix el consum d'emmagatzematge virtual global quan més d'una JVM comparteix una memòria cau. La compartició també redueix el temps d'inici d'una JVM un cop s'ha creat la memòria cau. La memòria cau de classes compartida és independent de qualsevol JVM en execució i persisteix fins que es suprimeix.

Una memòria cau compartida pot contenir:

- Classes bootstrap
- Classes d'aplicació
- Metadades que descriuen les classes
- Codi compilat de manera anticipada (AOT)

Nota: Una JVM que no sigui de temps reals no pot eliminar una memòria cau de classe compartida en temps real.

Capítol 8. Seguretat

Aquest apartat conté informació important sobre la seguretat.

Consideracions sobre seguretat per a la memòria cau de classes compartida

La memòria cau de classes compartida està dissenyada per facilitar la gestió i l'ús de la memòria cau, però pot ser que la política de seguretat per defecte no sigui adequada.

Quan utilitzeu la memòria cau de classes compartida, heu de tenir presents els permisos per defecte per als fitxers nous de manera que pugueu millorar la seguretat mitjançant la restricció d'accés.

Fitxer	**Permisos per defecte**
Memòries cau compartides noves	Permisos de lectura per a grup i altres
Directori `javasharedresources`	Permís de lectura, escriptura i execució

Cal tenir permís d'escriptura al fitxer de la memòria cau i al directori de la memòria cau per poder destruir o incrementar una memòria cau.

Canvi dels permisos de fitxer al fitxer de la memòria cau

Per limitar l'accés a una memòria cau de classes compartida, podeu utilitzar l'ordre **`chmod`**.

Canvi necessari	**Ordre**
Limitar l'accés a l'usuari i grup	`chmod 770 /tmp/javasharedresources`
Limitar l'accés a l'usuari	`chmod 700 /tmp/javasharedresources`
Limitar l'usuari a accés de lectura i escriptura per a una memòria cau determinada	`chmod 600 /tmp/javasharedresources/` `<`*`fitxer per a memòria cau compartida`*`>`
Limitar l'usuari i grup a accés de lectura i escriptura per a una memòria cau determinada	`chmod 660 /tmp/javasharedresources/` `<`*`fitxer per a memòria cau compartida`*`>`

Connexió a una memòria cau per a la qual no teniu permís d'accés

Si proveu de connectar-vos a una memòria cau per a la quan o teniu els permisos d'accés adequats, obtindreu un missatge d'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.
```

Capítol 9. Resolució de problemes i suport

Resolució de problemes i suport per al WebSphere Real Time for Linux

Mètodes de determinació de problemes generals

La determinació de problemes us ajuda a entendre el tipus d'error que teniu i l'acció correctora apropiada que cal seguir.

Quan sabeu el tipus de problema que teniu, podeu fer una o més de les tasques següents:

- Corregir el problema.
- Trobar una bona solució temporal.
- Recollir les dades necessàries amb les quals cal generar un informe d'errors per a IBM.

Determinació de problemes del Linux

En aquest apartat es descriu la determinació de problemes al Linux.

La guia de l'usuari de l'IBM SDK for Java V7 conté informació útil per diagnosticar problemes del Linux, com per exemple:

- Configuració i comprovació de l'entorn Linux
- Tècniques de depuració generals
- Diagnòstic de bloqueigs
- Depuració de bloqueigs
- Depuració de les fuites de memòria
- Depuració de problemes de rendiment

Podeu trobar aquesta informació aquí: IBM SDK for Java 7 - Determinació de problemes del Linux.

La informació següent és complementària per a l'IBM WebSphere Real Time for Linux

Configuració i comprovació de l'entorn Linux

A l'IBM WebSphere Real Time for Linux, comproveu que la JVM està configurada correctament per generar un abocament de memòria del sistema.

Abocaments de memòria del sistema del Linux (fitxers core)

Quan es produeix una fallada, les dades de diagnòstic més importants que cal obtenir és l'abocament de memòria del sistema del Linux (fitxer core). Per garantir que aquest fitxer es generi, cal que comproveu els paràmetres del sistema operatiu i l'espai de disc disponible, tal com es descriu a la guia de l'usuari de l'IBM SDK for Java V7.

Paràmetres de la màquina virtual Java

La JVM ha d'estar configurada per generar fitxers core quan es produeix una fallada. Executeu `java -Xdump:what` a la línia d'ordres. La sortida de l'opció és:

```
-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
```

Els valors que es mostren són els paràmetres per defecte. Com a mínim, cal definir `events=gpf` per generar un fitxer core quan es produeix una fallada. Podeu canviar i definir opcions amb l'opció de línia d'ordres **`-Xdump:system[:name1=value1,name2=value2 ...]`**

Tècniques de depuració generals

Ja que els noms de fils Java són visibles al sistema operatiu, podeu utilitzar l'ordre ps per facilitar la depuració. Quan utilitzeu eines de traça, heu d'utilitzar les ordres correctes per a l'IBM WebSphere Real Time for Linux.

Examen de la informació del procés

La sortida que veureu en executar l'ordre ps a l'IBM WebSphere Real Time for Linux és:

```
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 tots els processos.

L Mostra fils.

o Proporciona un format predefinit de columnes per visualitzar. Les columnes especificades són l'identificador de procés, l'identificador de fil, la política de planificació, la prioritat del fil en temps real i l'ordre associada amb el procés. Aquesta informació és útil per entendre quins fils de la vostra aplicació, així com la màquina virtual, s'executen en un moment determinat.

Eines de traça

Tres eines de traça al Linux són **`strace`**, **`ltrace`** i **`mtrace`**. L'ordre `man strace` visualitza un conjunt complet d'opcions disponibles.

`strace`

L'eina strace traça les crides del sistema. Podeu utilitzar-la en un procés que ja

estigui disponible, o iniciar-la amb un procés nou. L'eina strace enregistra les crides del sistema fetes per un programa, i els senyals rebuts per un procés. Per a cada crida del sistema, es fan servir el nom, els arguments i el valor de retorn. L'eina strace us permet traçar un programa sense que calgui el codi font (no es requereix cap recompilació). Si utilitzeu l'eina strace amb l'opció **-f**, traçarà els processos secundaris creats com a resultat d'una crida del sistema bifurcada. Podeu utilitzar l'eina strace per investigar problemes de connector o per intentar entendre els motius pels quals un programa no s'inicia adequadament.

Per utilitzar l'eina strace amb una aplicació Java, escriviu `strace java -Xgcpolicy:metronome <nom-classe>`.

Podeu dirigir el resultat de la traça des de l'eina strace a un fitxer utilitzant l'opció `-o`.

`ltrace`
L'eina ltrace depèn de la distribució. És molt semblant a l'eina strace. Aquesta eina intercepta i enregistra les crides de biblioteca dinàmica a mesura que les crida el procés d'execució. L'eina strace fa el mateix per als senyals rebuts per procés d'execució.

Per utilitzar l'eina ltrace amb una aplicació Java, escriviu `ltrace java -Xgcpolicy:metronome <nom-classe>`.

`mtrace`
L'eina mtrace s'inclou al conjunt d'eines GNU. Instal•la gestors especials per a malloc, realloc i free, i permet que tots els usos d'aquestes funcions es tracin i s'enregistrin en un fitxer. Aquesta traça disminueix l'eficàcia d'un programa i no s'hauria d'habilitar durant l'ús normal. Per utilitzar l'eina mtrace, definiu **IBM_MALLOCTRACE** a 1 i definiu **MALLOC_TRACE** per apuntar a un fitxer vàlid en el qual s'emmagatzemi la informació de traça. Heu de tenir permís d'escriptura per accedir a aquest fitxer.

Per utilitzar l'eina mtrace amb una aplicació Java, escriviu:

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

Diagnòstic de bloqueigs

Quan recolliu informació sobre els processos en execució i sobre l'entorn Java abans d'una fallada, seguiu aquestes indicacions.

Recollida d'informació del procés

Quan feu la recerca sobre què estava passant abans de produir-se la fallada, utilitzeu l'ordre **`gdb`** i **`bt`** per visualitzar la traça de pila del fil que ha fallat, en lloc d'analitzar el fitxer core.

Coneixement de l'entorn del Java

Utilitzeu el Javadump per determinar el que fa cada fil i quins mètodes del Java s'executen. Compareu les adreces de les funcions amb les adreces de les biblioteques per determinar l'origen del codi que s'executa a diversos punts.

Utilitzeu l'opció **`-verbose:gc`** per veure l'estat de l'emmagatzematge dinàmic del Java. Feu-vos aquestes preguntes:

- Hi ha una manca de memòria en alguna de les àrees de memòria que hagi pogut provocar la fallada?
- La fallada s'ha produït durant la recollida de deixalles, cosa que indicaria una possible fallada de la recollida de deixalles?
- La fallada s'ha produït després de la recollida de deixalles, cosa que indicaria possibles danys a la memòria?

Depuració de problemes de rendiment

En depurar problemes de rendiment, tingueu en compte aquests aspectes específics per a l'IBM WebSphere Real Time for Linux a més dels temes de la guia de l'usuari de l'IBM SDK for Java V7.

Dimensionament de les àrees de memòria

La mida de l'emmagatzematge dinàmic del Java és un dels paràmetres d'ajustament més importants de la JVM. Trieu la mida correcta per optimitzar el rendiment. L'ús de la mida correcta pot facilitar que el recollidor de deixalles proporcioni l'ús necessari.

Per obtenir més informació sobre el canvi de la mida de les àrees de memòria, consulteu "Resolució de problemes del recollidor de deixalles Metronome" a la pàgina 66.

Compilació i rendiment de JIT

Quan utilitzeu el JIT, heu de tenir en compte les implicacions en el comportament en temps real.

Limitacions conegudes del Linux

El Linux ha sofert un desenvolupament molt ràpid, i hi ha diversos problemes amb la interacció de la JVM i del sistema operatiu, especialment en l'àrea dels fils.

Observeu les limitacions següents que podrien afectar el vostres sistema Linux.

Fils com a processos

Si el nombre de fils del Java supera el nombre màxim de processos permesos, possiblement el vostre programa:

- Obtindrà un missatge d'error
- Obtindrà un error **SIGSEGV**
- S'aturarà

Per obtenir més informació, vegeu *l'informe Volano* a l'adreça http://www.volano.com/report/index.html.

Limitacions de piles flotants

Si esteu executant sense piles flotants, independentment de la definició per a **-Xss**, es proporciona una mida de pila nativa de 256 KB per a cada fil.

En un sistema Linux de pila flotant, s'utilitzen els valors de **-Xss**. Si migreu des d'un sistema Linux que no és de pila flotant, assegureu-vos que tots els valors **-Xss** són suficientment grans i no confien en un mínim de 256 KB.

limitacions de glibc

Si rebeu un missatge que indica que la biblioteca `libjava.so` no es pot carregar a causa d'un símbol que no es troba (com ara __bzero), és possible que tingueu instal•lada una versió anterior de la glibc (GNU C Runtime Library). La implementació de fils de l'SDK per al Linux requereix la glibc versió 2.3.2 o superior.

Limitacions de tipus de lletra

Quan feu la instal•lació en un sistema Red Hat, per permetre que el servidor de tipus de lletra trobi els tipus de lletra TrueType del Java TrueType, executeu (al Linux IA32, per exemple):

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

Podeu fer-ho en temps d'execució i heu d'haver iniciat la sessió com a usuari "root" per executar l'ordre. Per veure problemes de tipus de lletra més detallats, consulteu la publicació *Linux SDK and Runtime Environment User Guide*.

El planificador CFS (Completely Fair Scheduler) de Linux afecta el rendiment de Java

Les aplicacions Java que utilitzen àmpliament la sincronització podrien funcionar malament en les distribucions de Linux que inclouen el planificador CFS (Completely Fair Scheduler). CFS (Completely Fair Scheduler) és un planificador que s'ha adoptat en el kernel de Linux de línia principal a partir de la versió 2.6.23. L'algorisme del CFS és diferent dels algorismes de planificació de les versions anteriors del Linux. Podria canviar les propietats de rendiment d'algunes aplicacions. Concretament, el CFS implementa sched_yield() de manera diferent, i fa que sigui més probable que a un fil de rendiment se li doni temps de CPU.

Si detecteu aquest problema, és possible que observeu un ús alt de la CPU de l'aplicació Java i una progressió lenta a través dels blocs sincronitzats. Podria semblar que l'aplicació s'aturés a causa de la progressió lenta.

Hi ha dues possibles solucions temporals:

- Inicieu la JVM amb l'argument addicional **-Xthr:minimizeUserCPU**.
- Configureu el kernel del Linux per tal que utilitzi una implementació d'sched_yield() que sigui més compatible amb les versions anteriors. Feu-ho establint la propietat de kernel ajustable sched_compat_yield en **1**. Per exemple:

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

No utilitzeu aquestes solucions temporals si no experimenteu un funcionament deficient.

Aquest problema podria afectar el kit de desenvolupament i l'entorn de temps d'execució d'IBM per al Linux 5.0 (totes les versions) i 6.0 (totes les versions fins a l'SR 4, inclosa) que s'executen en els kernels del Linux que inclouen el planificador CFS (Completely Fair Scheduler). En el cas del kit de desenvolupament i entorn de temps d'execució d'IBM per al Linux versió 6.0 després de l'SR 4, es detecta l'ús del CFS al kernel i l'opció **-Xthr:minimizeUserCPU** s'habilita automàticament. Algunes distribucions del Linux que inclouen el planificador CFS (Completely Fair Scheduler) són Ubuntu 8.04 i SUSE Linux Enterprise Server 11.

Podeu trobar més informació sobre el CFS a l'apartat Multiprocessing with the Completely Fair Scheduler.

Problemes de rendiment als kernels del Linux Red Hat MRG

Un problema de configuració amb els kernels del Red Hat MRG pot causar pauses inesperades en els fils d'una aplicació quan s'inicia el WebSphere Real Time amb la recollida de deixalles detallada habilitada. No s'informa d'aquestes pauses al resultat detallat de GC, però poden durar diversos mil•lisegons, en funció de la configuració de la xarxa. Les JVM iniciades des d'usuaris definits de forma remota són les més afectades, perquè el daemon de la memòria cau de servei (`nscd`) no s'inicia, provocant retards a la xarxa. Solucioneu aquest problema iniciant el `nscd`. Seguiu aquests passos per comprovar l'estat del servei `nscd` i corregir el problema:

1. Comproveu que el daemon d'`nscd` s'executa, escrivint l'ordre:
   ```
   /sbin/service nscd status
   ```
   Si el daemon no s'executa, veureu el missatge següent:
   ```
   nscd is stopped
   ```
2. Com a usuari root, inicieu el servei `nscd` amb l'ordre següent:
   ```
   /sbin/service nscd start
   ```
3. Com a usuari root, canvieu la informació d'inici per al servei `nscd` amb l'ordre següent:
   ```
   /sbin/chkconfig nscd on
   ```

Ara el procés `nscd` s'està executant, i s'inicia automàticament després del reinici.

Determinació de problemes de suport multingüístic

La JVM conté un suport integrat per a diferents entorns locals.

La guia de l'usuari de l'IBM SDK for Java V7 conté informació útil per diagnosticar problemes del suport multingüístic, com per exemple:

- Visió general dels tipus de lletra
- Utilitats de tipus de lletra
- Problemes comuns de suport multingüístic i possibles causes

Podeu trobar aquesta informació aquí: IBM SDK for Java 7 - Determinació de problemes de suport multingüístic.

Determinació de problemes amb l'ORB

Una de les primeres tasques quan es depura un problema d'ORB és determinar si el problema és al client o al servidor de l'aplicació distribuïda. Considereu una sessió d'RMI-IIOP habitual com una comunicació simple i síncrona entre un client que sol•licita accés a un objecte i un servidor que el proporciona.

La guia de l'usuari de l'IBM SDK for Java V7 conté informació útil per diagnosticar problemes de l'ORB, com per exemple:

- Identificació d'un problema d'ORB
- Interpretació de la traça de pila
- Interpretació de les traces d'ORB
- Problemes comuns
- Servei ORB d'IBM: recollida de dades

Podeu trobar aquesta informació aquí: IBM SDK for Java 7 - Determinació de problemes amb l'ORB.

La informació següent és complementària per a l'IBM WebSphere Real Time for Linux.

Servei ORB d'IBM: recollida de dades

En recollir la sortida de la versió de Java per al servei, executeu l'ordre següent:

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

Proves preliminars

Si es produeix un problema, l'ORB pot generar una excepció `org.omg.CORBA.*` que inclou:

- El text que n'indica la causa
- Un codi menor minor
- Un estat de finalització

Abans que assumiu que l'ORB és la causa del problema, comproveu el següent:

- L'escenari es pot reproduir en una configuració semblant.
- El JIT està inhabilitat.
- No s'utilitza cap codi compilat AOT.

Altres accions:

- Desactiveu els processadors addicionals.
- Desactiveu els fils simultanis (SMT) on sigui possible.
- Elimineu les dependències de memòria amb el client o el servidor. La manca de memòria física pot ser la causa del baix rendiment, de bloqueigs aparents o de fallades. Per eliminar aquests problemes, assegureu-vos que teniu un marge raonable de memòria.
- Comproveu si hi problemes físics a la xarxa, com ara tallafocs, enllaços de comunicació, encaminadors i servidors de noms DNS. Són la causa principal de les excepcions COMM_FAILURE de CORBA. Com a prova, feu ping al nom de la vostra estació de treball.
- Si l'aplicació utilitza una base de dades com ara DB2, canvieu al programa de control més fiable. Per exemple, per aïllar DB2 AppDriver, canvieu a Net Driver, que és més lent i utilitza sòcols, però és més fiable.

Resolució d'errors OutOfMemory

Gestió de les excepcions OutOfMemoryError.

Per obtenir informació de resolució de problemes del recollidor de deixalles Metronome, consulteu “Resolució de problemes del recollidor de deixalles Metronome” a la pàgina 66.

Diagnòstic d'OutOfMemoryErrors

Diagnosticar excepcions OutOfMemoryError al Recollidor de deixalles Metronome pot ser més complex que en una JVM estàndard a causa de la naturalesa periòdica del recollidor de deixalles.

En general, una aplicació en temps real necessita aproximadament un 20% més d'espai d'emmagatzematge dinàmic que una aplicació Java estàndard.

Per defecte, la JVM produeix la següent sortida de diagnòstic quan es produeix un error OutOfMemoryError no detectat:

- Un abocament de memòria d'instantània; vegeu "Utilització d'agents d'abocament de memòria" a la pàgina 47.
- Un abocament heapdump; vegeu "Utilització del Heapdump" a la pàgina 55.
- Un abocament de memòria Java; vegeu "Utilització de Javadump" a la pàgina 50
- Un abocament de memòria del sistema; vegeu "Utilització dels abocaments de memòria del sistema i el visualitzador d'abocaments de memòria" a la pàgina 58.

Ells noms dels fitxers d'abocament de memòria s'indiquen a la sortida 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".
```

La traça en sentit invers de Java que es mostra a la sortida de la consola, i que també està disponible al Javadump, indica on s'ha produït l'error OutOfMemoryError de l'aplicació Java. El component de gestió de memòria de la JVM emet un punt de traça que indica la mida, l'adreça del bloc de classes i el nom de l'espai de memòria de l'assignació que falla. Aquest punt de traça és a l'abocament d'instantània:

```
<< 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
```

L'ID del punt de traça i els camps de dades poden ser diferents dels que es mostren aquí, en funció del tipus d'objecte assignat. En aquest exemple, el punt de traça mostra que l'error d'assignació s'ha produït quan l'aplicació provava d'assignar un objecte de 33,6 MB de tipus `class 0x81312d8` a `Metronome heap`, segment de memòria `id=0x809c5f0`.

Podeu determinar quina àrea de memòria queda afectada consultant la informació de gestió de memòria del Javadump:

```
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)

<< línies eliminades per facilitar la lectura >>
```

Podeu determinar el tipus d'objecte que s'assigna consultant la secció de classes del Javadump:

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

La informació del Javadump confirma que l'assignació que s'ha provat de fer era per a una matriu de caràcters, a l'emmagatzematge dinàmic nromal (`ID=0xF288B584`) i que la mida total assignada de l'emmagatzematge dinàmic, indicada per la línia `1STHEAP` corresponent, és 67108864 bytes decimals o 0x04000000 bytes hexadecimals, o 64 MB.

En aquest exemple, l'assignació que falla és gran en relació amb la mida total de l'emmagatzematge dinàmic. Si la vostra aplicació esperava crear objectes de 33 MB, el pas següent consisteix a incrementar la mida de l'emmagatzematge dinàmic, utilitzant l'opció **-Xmx**.

És molt comú que l'assignació que falla sigui petita amb relació a la mida total del 'emmagatzematge dinàmic. Això és així a causa de les assignacions anteriors que emplenen l'emmagatzematge dinàmic. En aquests casos, el proper pas consisteix a utilitzar Heapdump per analitzar la quantitat de memòria assignada als objectes existents.

El Heapdump és un fitxer binari comprimit que té una llista de tots els objectes amb les seves classes d'objecte, mida i referències. Analitzeu el Heapdump mitjançant l'eina IBM Monitoring and Diagnostic Tool for Java - Memory Analyzer, que podeu baixar des de l'IBM Support Assistant (ISA).

Mitjançant l'eina MDD4J, podeu carregar un Heapdump i localitzar tres estructures d'objectes que se sospita que consumeixen grans quantitats d'espai d'emmagatzematge dinàmic. L'eina proporciona diverses visualitzacions per a objectes a l'emmagatzematge dinàmic. Per exemple, MDD4J pot mostrar una visualització on s'indiquen les sospites de pèrdua de memòria i els cinc objectes i paquets principals que col•laboren a la mida de l'emmagatzematge dinàmic. Si seleccionem l'arbre, obtenim més informació quant a la naturalesa de l'objecte contenidor que té fuites.

Com gestiona la memòria la JVM d'IBM

L'IBM JVM necessita memòria per a diversos components, incloent-hi regions de memòria per a classes, codi compilat, objectes Java, piles Java i piles JNI. Algunes d'aquestes regions de memòria han de ser en memòria contigua. Altres regions de memòria es poden segmentar en regions de memòria més petites que es poden enllaçar entre elles.

El codi compilat i les classes carregades de manera dinàmica s'emmagatzemen en regions de memòria segmentada per a classes carregades de manera dinàmica. Les classes es subdivideixen encara més en regions de memòria gravables (classes RAM) i en regions de memòria de només lectura (classes ROM). En temps d'execució, les classes ROM i el codi AOT de la memòria cau de classes són memòries que s'assignen, però es carreguen, en una regió de memòria contigua en arrencar l'aplicació. A mesura que l'aplicació fa referència a les classes, les classes i el codi compilat de la memòria cau de classes s'assignen a l'emmagatzematge. El component ROM de la classe es comparteix entre diversos processos que fan referència a aquesta classe. El component de RAM de la classe es crea a les regions

de memòria segmentada per a classes carregades dinàmicament la primera vegada que la JVM fa referència a la classe. El codi compilat AOT per als mètodes d'una classe de la memòria cau de classes es copia en una regió de memòria de codi dinàmic executable, perquè els processos no comparteixen aquest codi. Les classes que no es carreguen de la memòria cau de classes són similars a les classes emmagatzemades a la memòria cau, tret que la informació de les classes ROM es crea en regions de memòria segmentada per a classes carregades de manera dinàmica. El codi generat de manera dinàmica s'emmagatzema a les mateixes regions de memòria de codi dinàmic que contenen el codi AOT per a les classes emmagatzemades a la memòria cau.

Aquesta pila per a cada fil Java pot abraçar una regió de memòria segmentada. La pila JNI per a cada fil ocupa una regió de memòria contigua.

Per determinar la manera com està configurada la JVM, utilitzeu l'opció **-verbose:sizes**. Aquesta opció mostra informació sobre les regions de memòria en les quals podeu gestionar la mida. Per ales regions de memòria que no són continues, es mostra un increment que descriu quanta memòria s'adquireix cada vegada que la regió necessita créixer.

Tot seguit es mostra un exemple de sortida amb les opcions **-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
```

Aquest exemple indica que el segment de classes RAM és inicialment 0, però es creix en blocs de 32 KB segons sigui necessari. El segment de classes ROM és inicialment 0, i creix en blocs de 128 KB segons sigui necessari. Podeu utilitzar les opcions **-Xmca** i **-Xmco** per controlar aquestes mides. Els segments de classes RAM i classes ROM creixen segons és necessari, per la qual cosa normalment no caldrà canviar aquestes opcions.

Utilitzeu l'opció **-Xshareclasses** per determinar la mida de la regió assignada de memòria si utilitzeu la memòria cau de classes. Tot seguit es mostra un exemple de la sortida de l'ordre `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
```

En temps d'execució, es copien uns 3 MB de bytes AOT i bytes de metadades a la regió segmentada de codi dinàmic, quan es fa referència a les classes. Els bytes de dades es copien al la regió segmentada de classes RAM, quan es fa referència a les classes.

Utilització d'eines de diagnòstic

Hi ha diverses eines de diagnòstic disponibles que ajuden a diagnosticar els problemes de la JVM de l'IBM WebSphere Real Time for Linux.

L'IBM SDK for Java 7 proporciona diverses eines de diagnòstic que podeu utilitzar per ajudar a diagnosticar els problemes de la JVM de l'IBM WebSphere Real Time for Linux. En aquest apartat es presenten les eines que hi ha disponibles i es proporcionen enllaços a informació sobre el seu ús.

Hi ha un aspecte important que cal recordar quan s'utilitzen les eines de diagnòstic de l'SDK. Quan invoqueu la JVM en temps real, utilitzeu l'opció següent:

```
java  -Xgcpolicy:metronome
```

Cal utilitzar aquesta opció quan s'executen les eines de diagnòstic per a la JVM en temps real. Per exemple, per mostrar els agents d'abocament de memòria registrats de la JVM de l'IBM WebSphere Real Time for Linux, escriviu:

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

A continuació, i com a informació complementària, es descriuen les diferències d'ús d'aquestes eines amb l'IBM WebSphere Real Time for Linux i s'inclou exemples de sortida per facilitar-ne el diagnòstic.

Per veure un resum de la informació de diagnòstic generada per l'IBM SDK for Java 7, vegeu Resum de la informació de diagnòstic.

Utilització de IBM Monitoring and Diagnostic Tools for Java

IBM proporciona eines i documentació per ajudar l'usuari a entendre, controla, i diagnosticar problemes en aplicacions mitjançant l'IBM JRE.

Hi ha disponibles les eines següents:

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

Garbage Collection and Memory Visualizer

El Garbage Collection and Memory Visualizer (GCMV) permet saber quin és l'ús de la memòria, el comportament de la recopilació de dades corrompudes i el rendiment de les aplicacions Java.

El GCMV analitza i mostra dades de diversos tipus de registre, per exemple:

- Registres de la recollida de deixalles detallada.

- Registres de recopilació de dades corrompudes amb traces, generats utilitzant el paràmetre -Xtgc.
- Registres de memòria nadiua, generats utilitzant les ordres del sistema ps, **svmon** o **perfmon**.

L'eina permet diagnosticar problemes com ara pèrdues de memòria, analitzar dades en diversos formats visuals i proporciona recomanacions d'ajust.

GCMV es proporciona com a complement de l'IBM Support Assistant (ISA). Per obtenir informació sobre com instal•lar i començar a utilitzar el complement, consulteu: http://www.ibm.com/developerworks/java/jdk/tools/gcmv/.

Trobareu més informació sobre el GCMV a l'IBM Information Center.

Health Center

El Health Center és una eina de diagnòstic per supervisar l'estat d'una màquina virtual Java (JVM) en execució.

L'eina es proporciona en dues parts:

- L'agent del Health Center que recopila les dades d'una aplicació en execució.
- Un client basat en Eclipse que es connecta a l'agent. El client interpreta les dades i proporciona recomanacions per millorar el rendiment de l'aplicació supervisada.

El Health Center es proporciona com a complement de l'IBM Support Assistant (ISA). Per obtenir informació sobre com instal•lar i començar a utilitzar el complement, consulteu: http://www.ibm.com/developerworks/java/jdk/tools/healthcenter/.

Trobareu més informació sobre el Health Center a l'IBM Information Center.

Interactive Diagnostic Data Explorer

L'Interactive Diagnostic Data Explorer (IDDE) és una alternativa al visor de l'abocament de memòria i que es basa en l'interfície d'usuari gràfica (ordre **jdmpview**). L'IDDE proporciona la mateixa funcionalitat que el visor de l'abocament de memòria, però amb un suport extra com ara la possibilitat de desar la sortida d'ordres.

Utilitzeu l'IDDE per explorar i examinar més fàcilment els fitxers de l'abocament de memòria que produeix la JVM. Dins de l'IDDE, cal introduir ordres en un registre d'investigació per explorar el fitxer de l'abocament de memòria. El suport que proporciona el registre d'investigació inclou els elements següents:

- Assistència per a ordres
- Compleció automàtica del text i alguns paràmetres com ara noms de classe
- La possibilitat de desar ordres i sortides que després podeu enviar a altres usuaris
- Text ressaltat i indicació de problemes
- La possibilitat d'afegir els vostres comentaris
- Suport per utilitzar el Memory Analyzer des de l'IDDE

L'IDDE es proporciona com a complement de l'IBM Support Assistant (ISA). Per obtenir informació sobre com instal•lar i començar a utilitzar el complement, consulteu Visió general de l'IDDE a developerWorks.

Trobareu més informació sobre l'IDDE a l'IBM Information Center.

Memory Analyzer

El Memory Analyzer us permet analitzar l'emmagatzematge dinàmic de Java mitjançant abocaments de memòria de nivell del sistema operatiu i Portable Heap Dumps (PHD).

Aquesta eina pot analitzar abocaments de memòria que continguin milions d'objectes si es proporciona la informació següent:

- La mida retinguda dels objectes.
- Els processos que impedeixen que el Recollidor de deixalles recopili objectes.
- Un informe per extreure sospistes de pèrdua de memòria automàticament.

Aquesta eina es basa en el projecte Eclipse Memory Analyzer (MAT) i utilitza la característica IBM Diagnostic Tool Framework for Java (DTFJ) per activar el processament d'abocaments de memòria de les JVM d'IBM.

El Memory Analyzer es proporciona com a complement de l'IBM Support Assistant (ISA). Per obtenir informació sobre com instal•lar i començar a utilitzar el complement, consulteu: http://www.ibm.com/developerworks/java/jdk/tools/memoryanalyzer/.

Trobareu més informació sobre el Memory Analyzer a l'IBM Information Center.

Utilització d'agents d'abocament de memòria

Els agents d'abocament de memòria es configuren durant la inicialització de la JVM. Permeten utilitzar incidències que tenen lloc a la JVM, com ara la recollida de deixalles, l'inici dels fils o la terminació de la JVM, a fi d'iniciar abocaments de memòria o iniciar una eina externa.

La guia de l'usuari de l'IBM SDK for Java V7 conté informació útil sobre els agents d'abocament de memòria, com per exemple:

- Utilització de l'opció **-Xdump**
- Agents d'abocament de memòria
- Incidències d'abocament de memòria
- Control avançat dels agents d'abocament de memòria
- Testimonis d'agent d'abocament de memòria
- Agents d'abocament de memòria per defecte
- Eliminació dels agents d'abocament de memòria
- Variables d'entorn d'agent d'abocament de memòria
- Mapatges de senyals
- Ubicacions per defecte dels agents d'abocament de memòria

Podeu trobar aquesta informació aquí: IBM SDK for Java 7 - Utilització dels agents d'abocament de memòria.

A continuació es proporciona informació complementària per a l'IBM WebSphere Real Time for Linux:

Incidències d'abocament de memòria

Els agents d'abocament de memòria s'activen quan es produeixen incidències durant el funcionament de la JVM. Per a l'IBM WebSphere Real Time for Linux, el valor per defecte per a la incidència slow és de 5 mil•lisegons.

Algunes incidències es poden filtrar per millorar la rellevància de la sortida. Consulteu “Opció de filtre” a la pàgina 49 per obtenir-ne més informació.

Nota: Actualment no es produeixen incidències de càrrega i expansió al WebSphere Real Time. Les classes són a memòria permanent i no es poden descarregar.

Nota: Les incidències gpf i abort no poden activar un heapdump, preparar l'emmagatzematge dinàmic (sol•licitud=prepwalk) ni compactar l'emmagatzematge dinàmic (sol•licitud=compact).

La taula següent mostra les incidències disponibles com a activadors d'agents d'abocament de memòria:

Incidència	Produïda quan...	Operació de filtre
gpf	Quan es produeix un error de protecció general (GPF).	
usuari	Quan la JVM rep el senyal SIGQUIT del sistema operatiu.	
abort	Quan la JVM rep el senyal SIGABRT del sistema operatiu.	
vmstart	Quan s'inicia la màquina virtual.	
vmstop	Quan s'atura la màquina virtual.	Filtra per codi de sortida; per exemple, **`filter=#129..#192#-42#255`**
load	Quan es carrega una classe.	Filtra per nom de classe; per exemple, **`filter=java/lang/String`**
unload	Quan es carrega una classe.	
throw	Quan es genera una excepció.	Filtra per nom de classe d'excepció; per exemple, **`filter=java/lang/OutOfMem*`**
catch	Quan es detecta una excepció.	Filtra per nom de classe d'excepció; per exemple, **`filter=*Memory*`**
uncaught	Quan l'aplicació no detecta una excepció Java.	Filtra per nom de classe d'excepció; per exemple, **`filter=*MemoryError`**
systhrow	Quan la JVM està a punt de generar una excepció Java. És diferent de la incidència 'throw' perquè només s'activa per a condicions d'error que es detecten internament a la JVM.	Filtra per nom de classe d'excepció; per exemple, **`filter=java/lang/OutOfMem*`**
thrstart	Quan s'inicia un fil nou.	
blocked	Quan es bloqueja un fil.	
thrstop	Quan s'atura un fil.	
fullgc	Quan s'inicia un cicle de recollida de deixalles.	
slow	Quan un fil triga més de 5 ms a respondre a una sol•licitud interna de la JVM.	Canvia el temps que es triga a considerar que una incidència és lenta; per exemple **`filter=#300ms`** s'activarà quan un fil trigui més de 300 ms a respondre a una sol•licitud interna de la JVM.
allocation	Quan un objecte Java s'assigna amb una mida que coincideix amb l'especificació de filtre indicada.	Filtra per mida d'objecte; cal proporcionar un filtre. Per exemple, **`filter=#5m`** s'activarà amb objectes de més de 5 Mb. També s'admeten intervals; per exemple, **`filter=#256k..512k`** s'activarà amb objectes d'una mida entre 256 Kb i 512 Kb.

Incidència	Produïda quan...	Operació de filtre
traceassert	Es produeix un error intern a la JVM	No aplicable.
corruptcache	La JVM descobreix que la memòria cau de classes compartida s'ha malmès.	No aplicable.

Opció de filtre

Algunes incidències de la JVM es produeixen milers de vegades durant la vida útil d'una aplicació. Els agents d'abocament de memòria podeu utilitzar filtres i intervals per evitar que es generin massa abocaments de memòria.

Comodins

Podeu utilitzar un comodí al filtre d'incidències d'excepció col•locant un asterisc només al començament o al final del filtre. L'ordre següent no funciona perquè el segon asterisc no és al final:

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

Perquè aquest filtre funcioni, cal canviar-lo a:

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

Càrrega de classes i incidències d'excepció

Podeu filtrar la càrrega de classes (load) i les incidències d'excepció (throw, catch, uncaught, systhrow) pel nom de classe Java:

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

Podeu filtrar les incidències d'excepció throw, uncaught i systhrow pel nom de mètode Java:

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

Les parts opcionals es mostren entre parèntesis.

Podeu filtrar les incidències d'excepció pel nom de mètode Java:

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

Les parts opcionals es mostren entre claudàtors.

Incidència vmstop

Podeu filtrar la incidència de conclusió de la JVM utilitzant un o més codis de sortida:

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

Incidència slow

Podeu filtrar la incidència slow per canviar el llindar de temps del valor per defecte 5 ms:

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

No podeu definir el filtre en menys temps que el temps per defecte.

Incidència allocation

Cal que filtreu la incidència allocation per especificar la mida dels objectes que ocasionen un activador. Podeu definir la mida del filtre de zero fins al valor màxim de punter de 32 bits a les plataformes de 32 bits o el valor màxim d'un punter de 64 bits a les plataformes de 64 bits. La definició del valor de filtre més baix en zero activa un abocament de memòria a totes les assignacions.

Per exemple, per activar abocaments de memòria a les assignacions amb més de 5 Mb, utilitzeu:

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

Per activar els abocaments de memòria a les assignacions amb una mida entre 256 Kb i 512 Kb, utilitzeu:

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

Altres incidències

Si apliqueu un filtre a una incidència que no admet el filtratge, el filtre s'ignora.

Utilització de Javadump

Un Javadump genera fitxers que contenen informació de diagnòstic relacionada amb la JVM i una aplicació Java capturada en un punt durant l'execució. Per exemple, la informació pot ser sobre el sistema operatiu, l'entorn d'aplicació, els fils, les piles, els bloqueigs i la memòria.

La guia de l'usuari de l'IBM SDK for Java V7 conté informació útil sobre Javadumps, com per exemple:

- Habilitació d'un Javadump
- Activació d'un Javadump
- Interpretació d'un Javadump
- Variables d'entorn i Javadump

Podeu trobar aquesta informació aquí: IBM SDK for Java 7 - Utilització de Javadump.

Als temes següents es proporcionen informació complementària i sortida d'exemple per a l'IBM WebSphere Real Time for Linux.

Gestió de l'emmagatzematge (MEMINFO)

La secció `MEMINFO` proporciona informació sobre el gestor de memòria, que inclou les àrees d'emmagatzematge dinàmic, memòria immortal i amb àmbit.

La secció `MEMINFO` d'un Javadump mostra informació sobre el gestor de memòria. Consulteu Utilització del Recollidor de deixalles Metronome per obtenir informació detallada sobre com funciona el component de gestor de memòria.

Aquesta part del Javadump proporciona diversos valors de gestió de l'emmagatzematge, com ara:

- Quantitat de memòria lliure
- Quantitat de memòria utilitzada
- Mida actual de l'emmagatzematge dinàmic
- Mida actual de les àrees de memòria immortal

- Mida actual de les àrees de memòria amb àmbit

Aquesta secció també conté dades històriques sobre la recollida de deixades. Les dades es mostren com a seqüència de punts de traça, cadascun amb una marca horària, ordenats a partir del punt de traça més recent.

Els Javadumps generats per la JVM estàndard contenen una secció "GC History". Aquesta informació no està continguda als Javadumps generats quan s'utilitza la JVM en temps real. Utilitzeu l'opció **-verbose:gc** o la traça breu de la JVM per obtenir informació sobre el comportament de la recollida de deixalles. Consulteu "Utilització de la informació verbose:gc" a la pàgina 66 i l'apartat sobre agents d'abocament de memòria de la guia de l'usuari de l'IBM SDK for Java V7 per obtenir-ne més detalls.

En un Javadump, els segments són blocs de memòria assignats pel temps d'execució Java per a tasques que utilitzen grans quantitats de memòria. Aquests són exemples de tasques:

- Mantenir les memòries cau JIT
- Emmagatzemar classes Java

L'entorn de temps d'execució Java també assigna una altra memòria nativa, que no s'indica a la secció `MEMINFO`. La memòria total utilitzada pels segments de temps d'execució Java no representa necessàriament l'impacte de memòria total del temps d'execució Java. Un segment de temps d'execució Java està format per l'estructura de dades del segment, i un bloc associat de memòria nativa.

L'exemple següent mostra una sortida habitual. Tots els valors es proporcionen com a valors hexadecimals. Les capçaleres de columna de la secció `MEMINFO` signifiquen el següent:

- Secció de memòria d'objecte (`HEAPTYPE`):

  id L'ID de l'espai o de la regió.

  start Adreça inicial d'aquesta regió de l'emmagatzematge dinàmic.

  end Adreça final d'aquesta regió de l'emmagatzematge dinàmic.

  size La mida d'aquesta regió de l'emmagatzematge dinàmic.

  space/region
  En una línia que només conté un `id` i un nom, aquesta columna mostra el nom de l'espai de memòria. Altrament, la columna mostra el nom de l'espai de memòria seguit del nom d'una regió determinada inclosa dins d'aquest espai de memòria.

- Secció de memòria interna (`SEGTYPE`), que inclou memòria de classe, memòria cau de codi JIT i memòria cau de dades JIT:

  segment
  Adreça de l'estructura de dades de segment de control.

  start L'adreça d'inici del segment de memòria nadiu.

  alloc L'adreça d'assignació actual del segment de memòria nadiu.

  end L'adreça final del segment de memòria nadiu.

  type Un camp de bits intern que descriu les característiques d'un segment de memòria nadiu.

  size La mida del segment de memòria nadiu.

```
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
       (línies eliminades per 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
        (línies eliminades per 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
            (línies eliminades per facilitar la lectura)
NULL
NULL           ------------------------------------------------------------------------
```

Fils i traça de pila (THREADS)

Per al programador d'aplicacions, una de les parts més útils d'un abocament de memòria Java és la secció THREADS. Aquesta secció mostra una llista de fils Java, fils natius, i traces de pila.

Els fils Java els implementen els fils natius del sistema operatiu. Cada fil es representa mitjançant un conjunt de línies com ara:

```
"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)
```

El noms de fils Java són visibles al sistema operatiu quan s'utilitza l'ordre **ps**. Per obtenir més informació sobre la utilització de l'ordre **ps**, consulteu “Tècniques de depuració generals” a la pàgina 36.

Les propietats de la primera línia són el nom del fil, les adreces de les estructures de fils de la JVM i de l'objecte de fil Java, l'estat del fil i la prioritat del fil Java. Les propietats de la segona línia són l'ID de fil del sistema operatiu natiu, la prioritat de fil del sistema operatiu natiu i la política de planificació del sistema operatiu natiu.

Els noms de fils són visibles de tres maneres:

- S'enumeren als fitxers javacore. No tots els fils s'enumeren als fitxers javacore.
- Quan es llisten fils del sistema operatiu mitjançant l'ordre **ps**.
- Quan s'utilitza el mètode java.lang.Thread.getName().

La taula següent proporciona informació sobre els noms de fils de l'IBM WebSphere Real Time for Linux.

Taula 4. Noms de fils a l'IBM WebSphere Real Time for Linux

Detall del fil	Nom del fil
Un fil intern de la JVM que utilitza el mòdul de recollida de deixalles per distribuir la finalització dels objectes per fils secundaris.	`Mestre finalitzador`

Taula 4. Noms de fils a l'IBM WebSphere Real Time for Linux (continuació)

Detall del fil	Nom del fil
El fil d'alarma que utilitza el recollidor de deixalles.	`Alarma de GC`
Els fils utilitzats per a la recollida de deixalles.	`Esclau del GC`
Un fil intern de la JVM que utilitza el mòdul del compilador JIT (just-in-time) per obtenir mostres de l'ús de mètodes de l'aplicació.	`IProfiler`
Un fil que utilitza la VM per gestionar els senyals que rep l'aplicació, tant si s'han generat externament com interna.	`Informador de senyals`
Un fil JVM intern que s'utilitza per compilar codi Java.	`JIT Compilation Thread`
Un fil JVM intern que s'utilitza per permetre que agenst JVMTI s'adjuntin a una JVM en execució.	`Attach API wait loop`

La prioritat de fil Java es mapa amb un valor de prioritat de sistema operatiu en funció de la plataforma. Un valor elevat per a la prioritat de fil Java significa que el fil té una prioritat alta. És a dir, el fil s'ha d'executar més sovint que els fils de prioritat més baixa.

El valors d'estat poden ser:

- R: executable; el fil es pot executar quan hi ha oportunitat.
- CW: espera de condició; el fil està a l'espera. Per exemple, perquè:
  - S'ha fet una crida a sleep().
  - S'ha bloquejat el fil per a E/S.
  - S'ha cridat un mètode wait() per esperar que es notifiqui un supervisor.
  - El fil se sincronitza amb un altre fil amb una crida a join().
- S: suspès; el fil l'ha suspès un altre fil.
- Z: zombi; el fil s'ha eliminat amb kill.
- P: aparcat; l'API nova de simultaneïtat ha aparcat el fil (java.util.concurrent).
- B: bloquejat; el fil està a l'espera d'obtenir un bloqueig propietat d'un altre element.

Si un fil està aparcat o blocat, la sortida inclou una línia per al fil, que comença per `3XMTHREADBLOCK`, i indica el recurs que el fil està esperant i, si pot ser, el fil que actualment és propietari del recurs. Per obtenir més informació, vegeu el tema sobre fils blocats a la guia de l'usuari de l'IBM SDK for Java V7.

Quan inicieu un Javadump per obtenir informació de diagnòstic, la JVM consulta els fils Java abans de produir el javacore. Es mostra l'estat de preparació `exclusive_vm_access` a la línia `1TIPREPSTATE` de la secció `TITLE` section.

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

Els fils que executaven codi Java en activar javacore tenen l'estat CW (condició d'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)
```

La secció javacore LOCKS mostra que aquests fils esperen en un bloqueig intern de la JVM.

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

Utilització del Heapdump

El terme heapdump descriu el mecanisme de la màquina virtual d'IBM per a Java que genera un abocament de memòria de tots els objectes actius que hi ha a l'emmagatzematge dinàmic de Java, és a dir, els que fa servir l'aplicació Java.

La guia de l'usuari de l'IBM SDK for Java V7 conté informació útil sobre Heapdumps, com per exemple:

- Obtenció de heapdumps
- Eines per processar heapdumps
- Utilització de **-Xverbose:gc** per obtenir l'emmagatzematge dinàmic
- Variables d'entorn i heapdump
- Format de fitxer de heapdump de text (clàssic)
- Format de fitxer PHD (Headump portàtil)

Podeu trobar aquesta informació aquí: IBM SDK for Java 7 - Utilització del Heapdump.

Informació complementària per a l'IBM WebSphere Real Time for Linux:

Format de fitxer de heapdump de text (clàssic)

El heapdump de text o clàssic és una llista de totes les instàncies d'objecte de l'emmagatzematge dinàmic, incloent-hi el tipus d'objecte, la mida i les referències entre objectes.

Registre de capçalera

El registre de capçalera és un registre individual que conté una cadena d'informació de versió.

```
// Version: <version string containing SDK level, platform and JVM build level>
```

Exemple:

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

Registres d'objecte

Els registres d'objecte són diversos registres, un per a cada instància de l'emmagatzematge dinàmic, que proporcionen l'adreça d'objecte, la mida, el tipus i les referències de l'objecte.

```
<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> ...
```

Les adreces d'objecte i les referències d'emmagatzematge dinàmic són a l'emmagatzematge dinàmic, però l'adreça de bloc de classes és fora de

l'emmagatzematge dinàmic. Totes les referències trobades a la instància d'objecte es llisten, incloses les referències que són valors nuls. El tipus d'objecte és un nom de classe, que inclou un paquet o un tipus de matriu de primitius o de classes, com mostra la seva signatura de tipus de JVM estàndard; consulteu "Signatures de tipus de màquina virtual Java" a la pàgina 57. Els registres d'objecte també poden contenir referències de bloc de classes addicionals, normalment en el cas de les instàncies de classes de reflex.

Exemples:

Una instància d'objecte de 28 bytes de longitud del tipus java/lang/String:

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

Una adreça de bloc de classes de java/lang/String, seguida per una referència a una instància de matriu char:

```
0x415319D8 0x00436EB0
```

Una instància d'objecte de 44 bytes de longitud de matriu char de tipus:

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

Una adreça de bloc de classes de matriu char:

```
0x41530F20
```

Un objecte de matriu de tipus classe interna java/util/Hashtable Entry:

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

Un objecte de tipus classe interna java/util/Hashtable:

```
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 adreça de bloc de classes i referències d'emmagatzematge dinàmic, incloent-hi les referències nul•les.

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

Registres de classe

Els registres de classe són diversos registres, un per a cada classe carregada, que proporcionen l'adreça de bloc de classes, la mida, el tipus i les referències de la classe.

```
<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>...
```

L'adreça de bloc de classes i les referències de bloc de classes són fora de l'emmagatzematge dinàmic, però el registre de classe pot contenir referències a l'emmagatzematge dinàmic, normalment per als membres de dades de classe estàtiques. Totes les referències trobades al bloc de classe es llisten, incloent-hi les que són valors nuls. El tipus de classe és un nom de classe, que inclou un paquet o un tipus de matriu de primitius o de classes, com mostra la seva signatura de tipus de JVM estàndard; consulteu "Signatures de tipus de màquina virtual Java" a la pàgina 57.

Exemples:

Un bloc de classe, d'una longitud de 32 bytes, per a la classe java/lang/Runnable:

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

Referències a altres blocs de classes i les referències d'emmagatzematge dinàmic, incloent-hi les referències nul•les:

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

Bloc de classes, d'una longitud de 168 bytes, per a la classe 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
```

Registre de cua 1

El registre de cua 1 és un registre individual que conté recomptes de registres.

```
// 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>
```

Exemple:

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

Registre de cua 2

El registre de cua 2 és un registre individual que conté totals.

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

Exemple:

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

Signatures de tipus de màquina virtual Java

Les signatures de tipus de màquina virtual Java són les abreviacions dels tipus Java que es mostren a la taula següent:

Signatures de tipus de màquina virtual Java	**Tipus Java**
Z	boolean
B	byte
C	char
S	short
I	int
J	long
F	float
D	double
L <*classe totalment qualificada*> ;	<*classe totalment qualificada*>
[<*tipus*>	<*tipus*>[ ] (matriu de <*tipus*>)

Signatures de tipus de màquina virtual Java	Tipus Java
(<*tipus-arg*>) <*tipus-ret*>	mètode

Utilització dels abocaments de memòria del sistema i el visualitzador d'abocaments de memòria

La JVM pot generar abocaments de memòria del sistema natius, anomenats també abocaments de memòria del nucli, en condicions configurables. Els abocaments de memòria del sistema solen ser grans. La majoria de les eines que s'utilitzen per analitzar els abocaments de memòria del sistema també són específiques de la plataforma. Utilitzeu l'eina **gdb** per analitzar un abocament de memòria del sistema al Linux.

La guia de l'usuari de l'IBM SDK for Java V7 conté informació útil sobre l'ús dels abocaments de memòria del sistema i del visualitzador d'abocaments de memòria, com per exemple:

- Visió general dels abocaments de memòria del sistema
- Valors per defecte de l'abocament de memòria del sistema
- Utilització del visualitzador d'abocaments de memòria
  - Utilització de **jextract**
  - Problemes que cal abordar amb el visualitzador d'abocaments de memòria
  - Ordres disponibles a **jdmpview**
  - Exemple de sessió
  - Consulta ràpida de les ordres **jdmpview**

Podeu trobar aquesta informació aquí: IBM SDK for Java 7 - Utilització dels abocaments de memòria del sistema i el visualitzador d'abocaments de memòria.

Informació complementària per a l'IBM WebSphere Real Time for Linux:

Ordres disponibles a jdmpview

jdmpview és una eina interactiva de línia d'ordres que serveix per explorar la informació d'un abocament de memòria del sistema de la JVM i efectuar diverses funcions d'anàlisi.

info jitm
Mostra els mètodes compilats de JIT i AOT i les seves adreces:
- Nom del mètode i signatura
- Adreça inicial del mètode
- Adreça final del mètode

Per veure la resta d'opcions de l'ordre, consulteu la guia de l'usuari de l'IBM SDK for Java V7.

Traça de les aplicacions Java i la JVM

La traça de JVM és un recurs de traça que es proporciona en l'IBM WebSphere Real Time for Linux amb un efecte mínim sobre el rendiment. A la majoria dels casos, les dades de traça es mantenen en un format binari compacte que es pot formatar amb el formatador Java subministrat.

La traça està habilitada per defecte, juntament amb un petit conjunt de punts de traça que van a les memòries intermèdies. Podeu habilitar els punts de traça en temps d'execució mitjançant la utilització de nivells, components, noms de grup o identificadors de punts de traça individuals.

La guia de l'usuari d'IBM SDK for Java V7 conté informació detallada sobre les aplicacions de traça, per exemple:

- De què es pot fer el seguiment
- Tipus de punts de traça
- Traça per defecte
- Enregistrament de les dades de traça
- Control de la traça
- Traça de les aplicacions Java
- Traça dels mètodes Java

En fer la traça de l'IBM WebSphere Real Time for Linux, heu d'invocar correctament la JVM en temps real quan incloeu les opcions de traça. Per exemple, quan especifiqueu les opcions de traça, escriviu:

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

Podeu obtenir la informació de l'IBM SDK for Java V7 aquí: Traça de les aplicacions Java i la JVM.

Determinació de problemes de JIT i AOT

Podeu utilitzar les opcions de línia d'ordres perquè resulti més fàcil diagnosticar problemes de compilador JIT i AOT i ajustar el rendiment.

Tot i que l'IBM WebSphere Real Time for Linux comparteix diversos components comuns amb l'IBM SDK for Java V7, el comportament de JIT i AOT és diferent. En aquest apartat es descriu la resolució de problemes de JIT i AOT a l'IBM WebSphere Real Time for Linux.

Diagnosi d'un problema de JIT o AOT>

En ocasions els codis de bytes vàlids es compilen en codi natiu no vàlid, cosa que fa que el programa Java falli. Si determineu que el compilador JIT o AOT és defectuós i, si és així, , *on* hi ha el defecte, podeu proporcionar una ajuda valuosa a l'equip de servei tècnic de Java.

Quant a aquesta tasca

Per determinar els mètodes que es compilen quan s'omple la memòria cau de classes compartida, utilitzeu l'opció **-Xaot:verbose** a la línia de l'ordre admincache. Per exemple:

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

En aquest apartat es descriu com es pot determinar si el problema està relacionat amb el compilador. En aquest apartat també se suggereixen algunes solucions temporals i tècniques de depuració per resoldre els problemes relacionats amb el compilador.

Desactivació del compilador JIT o AOT:

Si sospiteu que es produeix algun problema al compilador JIT o AOT, desactiveu la compilació per comprovar si el problema es continua produint. Si el problema encara es produeix, ja sabreu que el compilador no n'és la causa.

Quant a aquesta tasca

El compilador JIT està habilitat per defecte. El compilador AOT també està habilitat, però, no està actiu si no s'han habilitat les classes compartides. Per motius d'eficàcia, no tots els mètodes d'una aplicació Java es compilen. La JVM manté un recompte de crides per a cada mètode de l'aplicació; cada vegada que es crida i s'interpreta un mètode, el recompte de crides del mètode en qüestió augmenta. Quan el recompte arriba al llindar de compilació, el mètode es compila i s'executa de forma nativa.

El mecanisme de recompte de crides estén la compilació de mètodes al llarg de tota la vida útil d'una aplicació i assigna una prioritat més alta als mètodes que s'utilitzen més sovint. És possible que alguns mètodes que no s'utilitzen gaire sovint no es compilin mai. Com a resultat d'això, quan un programa Java falla, el problema pot ser al compilador JIT o AOT o pot ser a qualsevol altre punt de la JVM.

El primer pas per diagnosticar l'error és determinar *on* és el problema. Per fer-ho, primer cal que executeu el programa Java en mode interpretat pur (és a dir, amb el compilador JIT i AOT desactivats.

Procediment

1. Elimineu les opcions **-Xjit**i **-Xaot** (i els paràmetres que les acompanyen) de la línia d'ordres.
2. Utilitzeu l'opció de línia d'ordres **-Xint** per desactivar els compiladors JIT i AOT. Per motius de rendiment, no utilitzeu l'opció **-Xint** en un entorn de producció.

Què cal fer posteriorment

L'execució del programa Java amb la compilació inhabilitada duu a una de les situacions següents:

- L'error no desapareix. El problema és als compiladors JIT o AOT. En alguns casos, el programa pot començar a fallar d'una altra manera; el problema, tot i això, no està relacionat amb el compilador.
- L'error desapareix. El més probable és que el problema sigui al compilador JIT o AOT.

  Si no utilitzeu classes compartides, el compilador JIT té un error. Si utilitzeu classes compartides, cal que determineu quin compilador té l'error executant l'aplicació només amb la compilació JIT habilitada. Executeu l'aplicació amb l'opció **-Xnoaot** en lloc de l'opció **-Xint**. Això porta a una de les situacions següents:
  - L'error no desapareix. El problema és al compilador JIT. També podeu utilitzar l'opció **-Xnojit** en lloc de l'opció **-Xnoaot** per assegurar-vos que només el compilador JIT tingui un error.
  - L'error desapareix. El problema és al compilador AOT.

Inhabilitació selectiva del compilador JIT:

Si l'error del programa Java apunta a un problema amb el compilador JIT, podeu provar de delimitar encara més el problema.

Quant a aquesta tasca

Per defecte, el compilador JIT optimitza mètodes a diferents nivells d'optimització. S'apliquen diferents seleccions d'optimitzacions a diferents mètodes, segons els seus recomptes de trucades. Els mètodes que es criden amb més freqüència s'optimitzen a nivells superiors. En canviar els paràmetres del compilador JIT podeu controlar el nivell d'optimització al quals s'optimitzen els mètodes. Podeu determinar si l'optimitzador té un error i, si el té, quina optimització és problemàtica.

Especifiqueu els paràmetres JIT com a llista separada per comes afegida a l'opció **-Xjit**. La sintaxi és **-Xjit**:*<paràm1>*,*<paràm2>*=*<valor>*. Per exemple:

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

executa el programa HelloWorld, habilita la sortida detallada del JIT i fa que el JIT generi el codi natiu sense realitzar optimitzacions.

Seguiu aquests passos per determinar quina part del compilador provoca l'error:

Procediment

1. Definiu el paràmetre JIT **count=0** per canviar el llindar de compilació a zero. Aquest paràmetre fa que cada mètode Java es compili abans que s'executi. Utilitzeu **count=0** només quan diagnostiqueu problemes, perquè es compilen molts més mètodes, incloent-hi els mètodes que s'utilitzen poc sovint. La compilació addicional utilitza més recursos informàtics i provoca que l'aplicació vagi més lenta. Amb **count=0**, l'aplicació falla immediatament quan s'arriba a una àrea de problemes. En alguns casos, si s'utilitza **count=1**, l'error es pot reproduir de manera més fiable.
2. Afegiu **disableInlining** als paràmetres de compilador JIT. **disableInlining** inhabilita la generació de codi més gran i complex. Si el problema ja no es produeix, utilitzeu **disableInlining** com a solució temporal mentre l'equip de servei tècnic de Java analitza i corregeix el problema del compilador.
3. Reduïu els nivells d'optimització afegint el paràmetre **optLevel** i executeu una altra vegada el paràmetre fins que ja no es produeixi l'error, o fins que arribeu al nivell "noOpt". En el cas d'un problema de compilador JIT, comenceu amb "scorching" i continueu en ordre descendent per la llista. Els nivells d'optimització són en ordre descendent:
   a. scorching
   b. veryHot
   c. hot
   d. warm
   e. cold
   f. noOpt

Què cal fer posteriorment

Si un d'aquests paràmetres fa que l'error desaparegui, teniu una solució temporal que podeu utilitzar. Aquesta solució és temporal mentre l'equip de servei de Java analitza i corregeix el problema del compilador. Si eliminant **disableInlining** de la

llista de paràmetres JIT l'error no torna a aparèixer, feu-ho per millorar el rendiment. Seguiu les instruccions de "Ubicació del mètode que falla" per millorar el rendiment de la solució temporal.

Si l'error encara es produeix al nivell d'optimització "noOpt", cal que inhabiliteu el compilador JIT com a solució temporal.

Ubicació del mètode que falla:

Quan hàgiu determinat el nivell d'optimització més baix al qual el compilador JIT o AOT ha de compilar els mètodes per activar l'error, podeu esbrinar quina part del programa Java, quan es compila, ocasiona l'error. Tot seguit, podeu donar instruccions al compilador per limitar la solució temporal per a un mètode, una classe o un paquet específic, cosa que permetrà que el compilador compili la resta del programa com es fa habitualment. En el cas d'errors de compilador JIT, si l'error es produeix amb **-Xjit:optLevel=noOpt**, també podeu donar instruccions al compilador perquè no compili els mètodes que ocasionen l'error.

Abans de començar

Si veieu una sortida d'error com la d'aquest exemple, podeu utilitzar-la per identificar el mètode que falla:

```
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;
```

Les línies importants són:

vmState=0x00000000
Indica que el codi que ha fallat no era codi d'execució de la JVM.

Module= o Module_base_address=
No és a la sortida (pot estar en blanc o amb un valor zero) perquè JIT ha compilat el codi, i fora de qualsevol DLL o biblioteca.

Compiled_method=
Indica el mètode Java per al qual s'ha generat el codi compilat.

Quant a aquesta tasca

Si la sortida no indica el mètode que falla, seguiu els passos per identificar el mètode que falla:

Procediment

1. Executeu el programa Java amb els paràmetres JIT **verbose** i **vlog=**<*nom_fitxer*> afegits a l'opció **-Xjit**o **-Xaot**. Amb aquests paràmetres el compilador enumera els mètodes compilats en un fitxer de registre anomenat <*nom_fitxer*>.<*data*>.<*hora*>.<*pid*>, anomenat també *fitxer de límit*. Un fitxer de límit habitual conté línies que corresponen als mètodes compilats, com ara:
   ```
   + (hot) java/lang/Math.max(II)I @ 0x10C11DA4-0x10C11DDD
   ```

El compilador ignora les línies que no comencen amb el signe més als passos següents i podeu eliminar-les del fitxer. Els mètodes compilats pel compilador AOT comencen amb `+ (AOT cold)`. Els mètodes per als quals es carrega el codi AOT de la memòria cau de classes compartida comencen amb `+ (AOT load)`.

2. Torneu a executar el programa amb el paràmetre JIT o AOT **limitFile=**(<*nom_fitxer*>,<*m*>,<*n*>), on <*nom_fitxer*> és el camí d'accés al fitxer de límit i <*m*> i <*n*> són números de línia que indiquen el primer i el darrer mètode del fitxer de límit que s'han de compilar. El compilador només compila els mètodes que s'enumeren a les línies de <*m*> a <*n*> al fitxer de límit. Els mètodes que no s'enumeren al fitxer de límit i els mètodes que s'enumeren a línies que no formen part de l'abast no es compilen i no es carrega el codi AOT de la memòria cau de dades compartides corresponent a aquests mètodes. Si el programa ja no falla, un o més dels mètodes que heu eliminat a la darrera iteració ha d'haver estat la causa de l'error.
3. Opcional: Si esteu diagnosticant un problema d'AOT, executeu el programa un altre cop amb les mateixes opcions per permetre que els mètodes compilats es carreguin de la memòria cau de dades compartides. També podeu afegir l'opció **–Xaot:scount=0** per garantir que els mètodes compilats en mode AOT emmagatzemats a la memòria cau de dades compartides s'utilitzin quan es cridi el mètode el primer cop. Alguns errors de compilació AOT només es produeixen quan es carrega codi compilat en mode AOT de la memòria cau de dades compartides. Per ajudar a diagnosticar aquests problemes, utilitzeu l'opció **–Xaot:scount=0** per garantir que els mètodes compilats en mode AOT emmagatzemats a la memòria cau de dades compartides s'utilitzin quan es cridi el mètode per primer cop, cosa que pot facilitar la reproducció del problema. Teniu en compte que, si definiu l'opció **scount** en 0, això forçarà la càrrega del codi AOT i una pausa a qualsevol fil d'aplicació en espera de l'execució d'aquest mètode. Per tant, això només s'ha d'utilitzar per fer diagnòstics. Pot haver-hi més temps de pausa més significatius amb l'opció **–Xaot:scount=0**.
4. Repetiu aquest procés amb diferents valors per a <*m*> i <*n*>, tantes vegades com calgui, per trobar el conjunt mínim de mètodes que es poden compilar per activar l'error. Dividint entre dos el nombre de línies seleccionades cada vegada podeu efectuar una cerca binària del mètode que falla. Sovint podeu reduir el fitxer a una sola línia.

Què cal fer posteriorment

Quan hàgiu localitzar el mètode que falla, podeu inhabilitar el compilador JIT o AOT només per al mètode que falla. Per exemple, si el mètode java/lang/Math.max(II)I fa que el programa falli quan es compila en mode JIT i **optLevel=hot**, podeu executar el programa amb:

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

per compilar només el mètode que falla en un nivell d'optimització “warm”, però compilar la resta de mètodes de la manera habitual.

Si un mètode falla quan es compila en mode JIT a “noOpt”, podeu excloure'l totalment de la compilació, utilitzant el paràmetre **exclude**={<*method*>}:

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

Si un mètode fa que el programa falli quan es compila o escarrega codi AOT de la memòria cau de dades compartides, excloeu el mètode de la compilació AOT i la càrrega AOT mitjançant el paràmetre **exclude**={<*method*>}:

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

Els mètodes AOT es compilen només al nivell d'optimització "cold". El millor per a aquests mètodes és evitar la compilació o la càrrega AOT.

Identificació d'errors de compilació JIT:

En cas d'error de compilador JIT, analitzeu la sortida d'error per determinar si es produeix un error quan el compilador JIT intenta compilar un mètode.

Si hi ha una fallada de la JVM i podeu veure que l'error s'ha produït a la biblioteca (`libj9jit26.so`), pot ser que el compilador JIT hagi fallat mentre s'intentava compilar un mètode.

Si veieu una sortida d'error com la d'aquest exemple, podeu utilitzar-la per identificar el mètode que falla:

```
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;)
```

Les línies importants són:

vmState=0x00050000
: Indica que el compilador JIT està compilant codi. Per obtenir una llista de números de codi de vmState, consulteu la taula d'etiquetes del Javadump a la guia de l'usuari de l'IBM SDK for 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 l'error s'ha produït a `libj9jit26.so`, el mòdul de compilador JIT.

Method_being_compiled=
: Indica que s'està compilant el mètode Java.

Si la sortida no indica el mètode que falla, utilitzeu l'opció **verbose** amb els paràmetres addicionals següents:

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

Aquests paràmetres **verbose** notifiquen quan JIT comença a compilar un mètode i quan acaba. Si JIT falla en un mètode concret (és a dir, comença la compilació però falla abans de poder acabar), utilitzeu el paràmetre **exclude** per excloure'l de la compilació (consulteu "Ubicació del mètode que falla" a la pàgina 62). Si l'exclusió del mètode impedeix la fallada, heu trobat una solució temporal que podeu utilitzar mentre l'equip de servei tècnic corregeix el problema.

Rendiment de les aplicacions d'execució breu

El compilador JIT d'IBM s'ha ajustat per a aplicació d'execució llarga que normalment s'utilitzen en un servidor. Podeu utilitzar l'opció de línia d'ordres

-Xquickstart per millorar el rendiment de les aplicacions d'execució breu, especialment per a les aplicacions a les quals el procés no es concentra en un nombre menor de mètodes.

-Xquickstart fa que el compilador JIT utilitzi un nivell d'optimització inferior per defecte i compili menys mètodes. La realització de menys compilacions més ràpidament pot millorar el temps d'inici de l'aplicació. Quan el compilador AOT està actiu (tant les classes compartides com la compilació AOT habilitades), **-Xquickstart** fa que tots els mètodes seleccionats per a la compilació es compilin en mode AOT, cosa que millora el temps d'inici de les execucions posteriors. Pot ser que **-Xquickstart** degradi el rendiment si s'utilitza amb aplicacions de llarga execució que contenen mètodes que utilitzen una gran quantitat de recursos de processament. La implementació de **-Xquickstart** està sotmesa a canvis a les futures versions.

També podeu provar de millorar els temps d'inici ajustant el llindar de JIT (amb proves i errors). Consulteu "Inhabilitació selectiva del compilador JIT" a la pàgina 61 per obtenir-ne més informació.

Comportament de la JVM durant períodes inactius

Podeu reduir els cicles de CPU que consumeix una JVM inactiva utilitzant l'opció **-XsamplingExpirationTime** per desactivar el fil de mostratge JIT.

El fil de mostratge JIT defineix el perfil de l'aplicació Java en execució per descobrir els mètodes que s'utilitzen habitualment. L'ús de memòria i processador del fil de mostratge es pot obviar i la freqüència de definició de perfils es redueix de forma automàtica quan la JVM està inactiva.

En algunes circumstàncies, és possible que vulgueu que una JVM inactiva no consumeixi cap cicle de CPU. Per fer-ho, especifiqueu l'opció **-XsamplingExpirationTime**<*temps*>. Definiu <*temps*> en el nombre de segons durant els quals voleu que s'executi el fil de mostratge. Utilitzeu aquesta opció amb cura; després que es desactivi, no podreu reactivar el fil de mostratge. Permeteu que el fil de mostratge s'executi durant prou temps per identificar optimitzacions importants.

Recollidor de diagnòstics

El recollidor de diagnòstics recopila els fitxers de diagnòstic de Java per a un esdeveniment de problemes.

Si es recullen els fitxers necessaris per al servei d'IBM, es pot reduir el temps que es triga en resoldre problemes notificats. La guia de l'usuari d'IBM SDK for Java V7 conté informació detallada sobre la utilització del Recollidor de diagnòstics.

Podeu trobar aquesta informació aquí: IBM SDK for Java 7 - The Diagnostics Collector.

Dades de diagnòstic del recollidor de deixalles

En aquest apartat es descriu com diagnosticar els problemes de la recollida de deixalles.

La guia de l'usuari de l'IBM SDK for Java V7 conté informació útil per diagnosticar problemes del recollidor de deixalles, com per exemple:

- Registre de la recollida de deixalles detallada
- Traça de la recollida de deixalles mitjançant **-Xtgc**

Podeu trobar aquesta informació aquí: IBM SDK for Java 7 - Dades de diagnòstic del recollidor de deixallescompartida.

Als apartats següents es proporciona informació complementària del Recollidor de deixalles Metronome de l'IBM WebSphere Real Time for Linux.

Resolució de problemes del recollidor de deixalles Metronome

Mitjançant les opcions de la línia d'ordres, podeu controlar la freqüència de la recollida de deixalles de Metronome, les excepcions de manca de memòria i el comportament de Metronome en crides explícites del sistema.

Utilització de la informació verbose:gc:

Podeu utilitzar l'opció **-verbose:gc** amb l'opció **-Xgc:verboseGCCycleTime=N** per escriure informació a la consola sobre l'activitat del recollidor de deixalles Metronome. No totes les propietats de la sortida **-verbose:gc** de la JVM estàndard es creen o s'apliquen a la sortida del Recollidor de deixalles Metronome.

Utilitzeu l'opció **-verbose:gc** per visualitzar l'espai mínim, màxim i de mitjana de l'emmagatzematge dinàmic. D'aquesta manera, podeu comprovar el nivell d'activitat i l'ús de l'emmagatzematge dinàmic i llavors ajustar els valors segons calgui. L'opció **-verbose:gc** escriu estadístiques de Metronome a la consola.

L'opció **-Xgc:verboseGCCycleTime=N** controla la freqüència de recuperació de la informació. Determina el temps en mil•lisegons en què s'aboquen a la memòria els resums. El valor per defecte per a N és 1000 mil•lisegons. El temps de cicle no vol dir que el resum s'aboqui a la memòria en aquell moment concret, sinó que indica quan passa el darrer esdeveniment de recollida de deixalles que compleix aquests criteris de temps. La recollida i visualització d'aquestes estadístiques pot augmentar els temps de pausa del recollidor de deixalles Metronome i, a mesura que N té un valor més petit, els temps de pausa poden esdevenir elevats.

Un quantum és un període únic d'activitat Recollidor de deixalles Metronome que provoca una interrupció o una pausa per a una aplicació.

Exemple de sortida verbose:gc

Introduïu:

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

Quan s'activa la recollida de deixalles, es produeix l'esdeveniment `trigger start`, seguit per qualsevol nombre d'esdeveniments `heartbeat`, i després, quan l'activador està satisfet, es produeix un esdeveniment `trigger end`. Aquest exemple mostra un cicle de recollida de deixalles com a sortida 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" />
```

Es poden produir els tipus següents d'esdeveniment:

<trigger-start ...>
: Inici d'un cicle de recollida de deixalles, quan la quantitat de memòria utilitzada s'ha fet més gran que el llindar de l'activador. El llindar per defecte és el 50% de l'emmagatzematge dinàmic. L'atribut `intervalms` és l'interval entre l'esdeveniment `trigger end` anterior (amb id 1) i aquest esdeveniment `trigger start`.

<trigger-end ...>
: Un cicle de recollida de deixalles ha reduït satisfactòriament la quantitat de memòria utilitzada per sota del llindar de l'activador. Si un cicle de recollida de deixalles ha finalitzat, però la memòria utilitzada no s'ha reduït per sota del llindar de l'activador, s'inicia un nou cicle de recollida de deixalles amb el mateix ID de context. Per a cada esdeveniment `trigger start`, hi ha l'esdeveniment `trigger end` corresponent amb el mateix ID de context. L'atribut `intervalms` és l'interval entre l'esdeveniment `trigger start` anterior i l'esdeveniment actual `trigger end`. Durant aquest temps, s'haurà completat un cicle de recollida de deixalles, o més d'un, fins que la memòria utilitzada caigui per sota del llindar de l'activador.

<gc-op id="28" type="heartbeat"...>
: Esdeveniment periòdic que recopila informació (de la memòria i el temps) sobre els quantums de recollida de deixalles per al període de temps que cobreix. Un esdeveniment heartbeat només es pot produir entre una parella d'esdeveniments `trigger start` i `trigger end`, és a dir, mentre hi ha un cicle de recollida de deixalles actiu. L'atribut `intervalms` és l'interval entre l'esdeveniment heartbeat anterior (amb `id` -1) i aquest esdeveniment heartbeat.

<gc-op id="29" type="syncgc"...>
: Esdeveniment de recollida de deixalles síncron (no determinista). Consulteu “Recollides de deixalles síncrones” a la pàgina 68

Tot seguit s'indica el significat de les etiquetes XML d'aquest exemple:

<quanta ...>
: Resum dels temps de pausa de quantum durant l'interval de heartbeat, que inclou la durada de les pauses en mil•lisegons.

<free-mem type="heap" ...>
: Resum de la quantitat d'espai lliure d'emmagatzematge dinàmic durant l'interval heartbeat, mostrat al final de cada quantum de recollida de deixalles.

<classunload-info classloadersunloaded=9 classesunloaded=156 />
Nombre de carregadors de classes i classes baixades durant l'interval heartbeat.

<references type="weak" cleared="660 />
Nombre i tipus d'objectes de referència Java que s'han esborrat durant l'interval heartbeat.

Nota:

- Si s'ha produït un quantum de recollida de deixalles a l'interval entre dos heartbeats, la memòria lliure només es mostreja al final d'aquest quantum. Per tant, les quantitats mínima, màxima i de mitjana donades al resum de heartbeat són totes iguals.
- Pot ser que l'interval entre dos esdeveniments heartbeat sigui significativament més gran que el temps de cicle especificat si l'emmagatzematge dinàmic no és prou ple per necessitar l'activitat de recollida de deixalles. Per exemple, si el vostre programa necessita una que l'activitat de recollida de deixalles s'executi només cada pocs segons, pot ser que veieu només un heartbeat cada pocs segons.
- Pot ser que l'interval sigui significativament més gran que el temps de cicle especificat perquè la recollida de deixalles no té feina en un emmagatzematge dinàmic que no està prou ple per garantir l'activitat de recollida de deixalles. Per exemple, si el vostre programa necessita una que l'activitat de recollida de deixalles s'executi només cada pocs segons, pot ser que veieu només un heartbeat cada pocs segons.

  Si es produeix un esdeveniment com ara una recollida de deixalles síncrona o un canvi de prioritat, els detalls de l'esdeveniment i els esdeveniments pendents, com ara esdeveniments heartbeat, es produeixen immediatament com a sortida.
- Si el quantum màxim de recollida de deixalles per a un període determinat és massa gran, és recomanable que reduïu la utilització objecti l'opció **-Xgc:targetUtilization**. Aquesta acció dóna al recollidor de deixalles més temps per treballar. També podeu incrementar la mida de l'emmagatzematge dinàmic mitjançant l'opció **-Xmx**. De la mateixa manera, si la vostra aplicació pot tolerar retards més llargs dels que es notifiquen actualment, podeu incrementar la utilització objectiu o reduir la mida de l'emmagatzematge dinàmic.
- La sortida es pot redirigir a un fitxer de registre, en lloc de redirigir-la a la consola, mitjançant l'opció **-Xverbosegclog:<fitxer>**; per exemple, **-Xverbosegclog:out** escriu la sortida de **-verbose:gc** al fitxer *out*.
- La prioritat que s'indica a `thread-priority` és la prioritat del fil del sistema operatiu subjacent, no una prioritat de fil Java.

Recollides de deixalles síncrones

També s'escriu una entrada al registre **-verbose:gc** quan es produeix una recollida de deixalles síncrona (no determinista). Aquest esdeveniment té tres possibles causes:

- Una crida `System.gc()` explícita al codi.
- La JVM es queda sense memòria i duu a terme una recollida de deixalles síncrona per evitar una condició OutOfMemoryError.
- La JVM es tanca durant una recollida de deixalles contínua. La JVM no pot cancel•lar la recollida; per tant, fa la recollida de manera asíncrona i llavors finalitza.

Tot seguit es mostra un exemple d'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>
```

L'exemple següent correspon a una entrada de recollida de deixalles síncrona com a conseqüència del tancament 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>
```

Les etiquetes XML i els atributs d'aquest exemple tenen aquest significat:

<gc-op id="9" type="syncgc" timems="6.439" ...
: Aquesta línia indica que el tipus d'esdeveniment és una recollida de deixalles síncrona. L'atribut `timems` és la duració de la recollida de deixalles síncrona en mil•lisegons.

<syncgc-info reason="..."/>
: Causa de la recollida de deixalles síncrona.

<free-mem-delta.../>
: Memòria d'emmagatzematge dinàmic Java lliure abans i després de la recollida de deixalles síncrona, en bytes.

<finalization .../>
: Nombre d'objectes que esperen la finalització.

<classunload-info .../>
: Nombre de carregadors de classes i classes baixades durant l'interval heartbeat.

<references type="weak" cleared="53" .../>
: Nombre i tipus d'objectes de referència Java que s'han esborrat durant l'interval heartbeat.

La recollida de deixalles sincrònica que es produeix a causa de condicions de manca de memòria o del tancament de la VM només es pot produir quan el recollidor de deixalles està actiu. Ha d'anar precedida per un esdeveniment `trigger start`, tot i que no es necessari que sigui de manera immediata. Pot ser que es produeixin alguns esdeveniments heartbeat entre un esdeveniment `trigger start` i l'esdeveniment `synchgc`. La recollida de deixalles causada per `System.gc()` es pot produir en qualsevol moment.

Traça de tots els quantums de recollida de deixalles

La traça dels quantums de recollida de deixalles individuals es pot fer habilitant els punts de traça `GlobalGCStart` i `GlobalGCEnd`. Aquests punts de traça es produeixen el principi i al final de tota l'activitat del recollidor de deixalles Metronome, incloent-hi recollides de deixalles sincròniques. La sortida dels punts de traça serà semblant a aquesta:

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

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

Entrades de manca de memòria

Quan l'emmagatzematge dinàmic es queda sense espai, s'escriu una entrada al registre **-verbose:gc** abans que es generi l'excepció OutOfMemoryError. Exemple d'aquesta sortida:

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

Per defecte, es produeix un Javadump com a conseqüència d'una excepció OutOfMemoryError. Aquest abocament de memòria inclou informació sobre la memòria utilitzada pel programa.

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

Comportament del recollidor de deixalles Metronome en condicions de manca de memòria:

Per defecte, el recollidor de deixalles Metronome activa una recollida de deixalles il•limitada i no determinista quan la JVM es queda sense memòria. Per evitar el comportament no determinista, utilitzeu l'opció **-Xgc:noSynchronousGCOnOOM** per generar un error OutOfMemoryError quan la JVM es quedi sense memòria.

La recollida il•limitada per defecte s'executa fins que s'han recollit totes les deixalles possibles en una única operació. El temps de pausa necessari sol ser molts mil•lisegons més gran que el d'un quantum incremental de Metronome normal.

Informació relacionada:

Ús de -Xverbose:gc per analitzar recollides de deixalles síncrones

Comportament del recollidor de deixalles Metronome en crides System.gc() explícites:

Si hi ha un cicle de recollida de deixalles en curs, el recollidor de deixalles Metronome completa el cicle de manera síncrona quan es crida `System.gc()`. Si no hi ha cap cicle de recollida de deixalles en curs, es duu a terme un cicle síncron complet quan es crida `System.gc()`. Utilitzeu `System.gc()` per netejar l'emmagatzematge dinàmic de manera controlada. Es tracta d'una operació no determinista perquè realitza una recollida de deixalles completa abans de tornar.

Algunes aplicacions criden programari de proveïdor que té crides System.gc() en els quals no es permet crear aquests retards no deterministes. Per inhabilitar totes les crides System.gc(), utilitzeu l'opció **-Xdisableexplicitgc**.

La sortida detallada de la recollida de deixalles per a una crida System.gc() té com a motiu “system garbage collect” i pot ser que sigui llarga:

```
<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>
```

Dades de diagnòstic de classes compartides

Entendre com es poden produir els problemes de diagnòstic us ajuda a utilitzar el mode de classes compartides.

La guia de l'usuari de l'IBM SDK for Java V7 conté informació útil per diagnosticar problemes de classes compartides, com per exemple:

- Desplegament de classes compartides
- Tractament de la modificació del codi de bytes de temps d'execució
- Descripció de les actualitzacions dinàmiques
- Utilització de l'API d'ajuda de Java
- Descripció de la sortida dels diagnòstics de classes compartides
- Depuració de problemes amb les classes compartides

Podeu trobar aquesta informació aquí: IBM SDK for Java 7 - Dades de diagnòstic de classe compartida.

Utilització de la JVMTI

JVMTI és una interfície de dos sentits que permet la comunicació entre la JVM i un agent natiu. Substitueix les interfícies JVMDI i JVMPI.

La JVMTI permet que altres proveïdors desenvolupin eines de depuració, definició de perfils i supervisió per a la JVM. La interfície conté mecanismes perquè l'agent notifiqui a la JVM els tipus d'informació que li calen. La interfície també proporciona un mitjà per rebre les notificacions rellevants. Es poden connectar diversos agents a una JVM en qualsevol moment.

La guia de l'usuari d'IBM SDK for Java V7 conté informació detallada sobre la utilització de la JVMTI, que inclou un apartat de referència de l'API sobre les extensions de JVMTI d'IBM.

Podeu trobar aquesta informació aquí: IBM SDK for Java 7 - Utilització de la JVMTI.

Utilització de l'estructura d'eines de diagnòstic per a Java

L'estructura d'eina de diagnòstic per a Java (DTFJ) és una interfície de programació d'aplicacions Java (API) d'IBM que s'utilitza per admetre la creació d'eines de diagnòstic Java. DTFJ funciona amb dades d'un abocament de memòria del sistema o Javadump.

La guia de l'usuari d'IBM SDK for Java V7 conté informació detallada sobre DTFJ. Seguiu aquest enllaç: Utilització de l'estructura d'eines de diagnòstic per a Java

Capítol 10. Referència

En aquests temes s'indiquen les opcions i les biblioteques de classes que es poden utilitzar amb l'WebSphere Real Time for Linux

Opcions de línia d'ordres

Podeu especificar opcions a la línia d'ordres mentre inicieu Java. Les opcions per defecte s'han triat per al seu millor ús general.

Especificació d'opcions i propietats del sistema Java

Hi ha tres maneres d'especificar les propietats Java i les propietats del sistema.

Quant a aquesta tasca

Podeu especificar opcions Java i propietats del sistema segons diferents mètodes. Per ordre de preferència, són:

1. Especificar l'opció o propietat a la línia d'ordres. Per exemple:

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

2. Crear un fitxer que inclogui les opcions i especificar-ho a la línia d'ordres mitjançant l'opció **-Xoptionsfile=**<*nom_fitxer*>.

   Al fitxer d'opcions, especifiqueu cada opció en una línia nova; podeu utilitzar el caràcter '\' com a caràcter de continuació si voleu que una opció individual abasti diverses línies. Utilitzeu el caràcter '#' per definir les línies de comentari. No podeu especificar **-classpath** en un fitxer d'opcions. Tot seguit es mostra un exemple de fitxer d'opcions:

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

3. Crear una variable d'entorn anomenada **IBM_JAVA_OPTIONS** que conté les opcions. Per exemple:

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

L'última opció que especifiqueu a la línia d'ordres preval per sobre de la primera opció. Per exemple, si especifiqueu les opcions **-Xint -Xjit myClass**, l'opció **-Xjit** té preferència sobre **-Xint**.

Propietats del sistema

Les propietats del sistema estan disponibles per a les aplicacions, i ajuden a proporcionar informació sobre l'entorn d'execució.

com.ibm.jvm.realtime

Aquesta propietat permet que les aplicacions Java determinen si s'executen en un entorn WebSphere Real Time for Linux.

Si la vostra aplicació s'executa dins de l'entorn de temps d'execució del IBM WebSphere Real Time for RT Linux, i s'ha iniciat amb l'opció **-Xrealtime**, la propietat **com.ibm.jvm.realtime** té el valor "hard".

Si la vostra aplicació s'executa dins de l'entorn de temps d'execució de l'IBM WebSphere Real Time for RT Linux, però no s'ha iniciat amb l'opció **-Xrealtime**, la propietat **com.ibm.jvm.realtime** no es defineix.

Si la vostra aplicació s'executa dins de l'entorn del temps d'execució del IBM WebSphere Real Time, la propietat **com.ibm.jvm.realtime** té el valor "soft".

Opcions estàndard

Les definicions per a les opcions estàndard.

-agentlib:<*nom_biblioteca*>**[=**<*opcions*>**]**
Carrega una biblioteca d'agents nadius <*nom_biblioteca*>; per exemple **-agentlib:hprof**. Per obtenir més informació, especifiqueu **-agentlib:jdwp=help** i **-agentlib:hprof=help** a la línia d'ordres.

-agentpath:*nom_biblioteca***[=**<*opcions*>**]**
Carrega una biblioteca d'agents nadius pel nom de camí d'accés complet.

-assert Imprimeix l'ajuda sobre les opcions relacionades amb assert.

-cp o -classpath <*directoris i fitxers .zip o .jar separats per* **:**>
Estableix el camí de cerca per a les classes i recursos de l'aplicació. Si no utilitzeu **-classpath** i **-cp** i no s'ha definit **CLASSPATH**, la variable classpath de l'usuari és, per defecte, el directori actual (.).

-D<nom_propietat>=<*valor*>
Estableix una propietat del sistema.

-help or -?
Imprimeix un missatge d'ús.

-javaagent:<*camí_accés_jar*>**[=**<*opcions*>**]**
Carrega un agent de llenguatge de programació Java. Per obtenir més informació, consulteu la documentació de l'API java.lang.instrument.

-jre-restrict-search
Inclou els JRE privats en la cerca de la versió.

-no-jre-restrict-search
Exclou els JRE privats d'usuari a la cerca de versió.

-showversion
Imprimeix la versió del producte i continua.

-verbose:[*class***,***gc***,***dynload***,***sizes***,***stack***,***jni***]**
Habilita la sortida detallada.

-verbose:class
Escriu una entrada a l'stderr per a cada classe que està carregada.

-verbose:gc
Consulteu "Utilització de la informació verbose:gc" a la pàgina 66.

-verbose:dynload
Proporciona informació detallada mentre la JVM carrega cada classe, incloent-hi:
- El nom de classe i el paquet
- Per als fitxers de classes que són al fitxer `.jar`, nom i camí d'accés del directori del fitxer `.jar`.
- Detalls sobre la mida de la classe i el temps que es triga a carregar la classe

Les dades s'escriuen a stderr. Tot seguit es mostra un exemple de la sortida:

```
<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: Les classes que es carreguen de la memòria cau de classe compartida no apareixen a la sortida de **-verbose:dynload**. Utilitzeu **-verbose:class** per obtenir informació sobre aquestes classes.

-verbose:sizes
Escriu informació a la sortida estàndard (stderr) que descriu la quantitat de memòria per a les piles i els emmagatzematges dinàmics de la JVM.

-verbose:stack
Escriu informació a la sortida estàndard que descriu l'ús de la pila Java i C.

-verbose:jni
Escriu informació a l'stderr que descriu els serveis JNI que crida l'aplicació i la JVM.

-version
Imprimeix la informació de la versió del mode de temps no real.

-version:<*valor*>
Requereix que s'executi la versió especificada.

-X Imprimeix l'ajuda sobre les opcions no estàndard.

Opcions no estàndard

Les opcions que tenen el prefix **-X** no són estàndard i estan subjectes a canvis sense notificació prèvia.

La guia de l'usuari d'IBM SDK for Java V7 conté informació detallada sobre les opcions no estàndard. Podeu trobar aquesta informació aquí: IBM SDK for Java 7 - Opcions de línia d'ordres.

Als apartats següents es proporciona informació complementària de l'IBM WebSphere Real Time for Linux.

Opcions del recollidor de deixalles Metronome

Definicions de les opcions del recollidor de deixalles Metronome.

-Xgc:synchronousGCOnOOM | -Xgc:nosynchronousGCOnOOM
Un dels casos en què es produeix la recollida de deixalles és quan l'emmagatzematge dinàmic es queda sense memòria. Si no hi ha més espai lliure a l'emmagatzematge dinàmic, l'ús **-Xgc:synchronousGCOnOOM** atura l'aplicació mentre el procés de recollida de deixalles elimina els objectes no utilitzats. Si l'espai lliure es torna a exhaurir, considereu la possibilitat de reduir l'ús objectiu per permetre més temps perquè finalitzi la recollida de deixalles. Definir **-Xgc:nosynchronousGCOnOOM** implica que, quan la memòria de l'emmagatzematge dinàmic estigui plena, l'aplicació s'aturarà i emetrà un missatge de manca de memòria. El valor per defecte és **-Xgc:synchronousGCOnOOM**.

-Xnoclassgc
Inhabilita la recollida de deixalles de classe. Aquesta opció desactiva la

recollida de deixalles d'emmagatzematge associada amb les classes Java que ja no utilitza la JVM. El comportament per defecte és **-Xnoclassgc**.

-Xgc:targetPauseTime=*N*
Defineix el temps de pausa de la recollida de deixalles, on *N* és el temps en mil•lisegons. Quan s'especifica aquesta opció la recollida de deixalles es realitza amb pauses que no superen el valor indicat. Si aquesta opció no s'especifica, el temps de pausa per defecte és de 3 mil•lisegons. Per exemple, l'execució amb **-Xgc:targetPauseTime=20** fa que el recollidor de deixalles faci una pausa no superior als 20 mil•lisegons durant les operacions de recollida de deixalles.

-Xgc:targetUtilization=*N*
Defineix l'ús de l'aplicació a N%; el recollidor de deixalles prova d'utilitzar com a màxim (100-N)% de cada interval de temps. Els valors raonables corresponen a l'interval 50-80%. Les aplicacions amb índex d'assignació més baixos pot ser que es puguin executar al 90%. El valor per defecte és 70%.

Aquest exemple mostra que la mida màxima de la memòria d'emmagatzematge dinàmic és 30 MB. El recollidor de deixalles intenta utilitzar el 25% de cada interval de temps perquè l'ús objectiu per a l'aplicació és el 75%.

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

-Xgc:threads=*N*
Especifica el nombre de fils de recollida de deixalles que cal executar. El valor per defecte és el nombre de nuclis de processador disponibles per al procés. El valor màxim que podeu especificar és el nombre de processadors disponibles per al sistema operatiu.

-Xgc:verboseGCCycleTime=N
N és el temps en mil•lisegons en què s'hauria d'abocar a memòria la informació de resum.

Nota: El temps de cicle no vol dir que la informació de resum s'aboqui a la memòria en aquell moment concret, sinó que indica quan passa el darrer esdeveniment de recollida de deixalles que compleix aquests criteris de temps.

-Xmx<*mida*>
Especifica la mida de l'emmagatzematge dinàmic Java. A diferència d'altres estratègies de recollida de deixalles, la recollida de deixalles Metronome en temps real no admet l'expansió de l'emmagatzematge dinàmic. No hi ha cap opció per a la mida inicial o màxima de l'emmagatzematge dinàmic. Podeu especificar només la mida d'emmagatzematge dinàmic màxima.

Paràmetres per defecte de la JVM

Els valors per defecte s'apliquen a la JVM en temps real quan no es fa cap canvi a l'entorn en el qual s'executa la JVM. Es mostren valors comuns com a referència.

Els valors per defecte es poden modificar utilitzant variables d'entorn o paràmetres de la línia d'ordres en arrencar al JVM. A la taula es mostren alguns dels valors comuns de la JVM. La darrera columna indica com podeu canviar el comportament, on s'apliquen aquestes claus:

- **e**: paràmetre que només controla la variable d'entorn.
- **c**: paràmetre que només controla un paràmetre de línia d'ordres.

- **ec**: paràmetre controlat per la variable d'entorn i el paràmetre de la línia d'ordres, on el paràmetre de la línia d'ordres té preferència.

La informació es proporciona com a referència ràpida i no és completa.

Paràmetre de la JVM	Valor per defecte	Paràmetre afectat per
Javadumps	Habilitat	ec
Javadumps amb manca de memòria	Habilitat	ec
Heapdumps	Inhabilitat	ec
Heapdumps amb manca de memòria	Habilitat	ec
Sysdumps	Habilitat	ec
Ubicació on es generen els fitxers d'abocament de memòria	Directori actual	ec
Sortida detallada	Inhabilitat	c
Cerca a la variable classpath d'arrencada	Inhabilitat	c
Comprovacions de JNI	Inhabilitat	c
Depuració remota	Inhabilitat	c
Comprovacions de compliment estricte	Inhabilitat	c
Inici ràpid	Inhabilitat	c
Servidor d'informació de depuració remot	Inhabilitat	c
Senyalització reduïda	Inhabilitat	c
Encadenament de gestors de senyals	Habilitat	c
Classpath	No definit	ec
Compartició de dades de classe	Inhabilitat	c
Suport per a accessibilitat	Habilitat	e
compilador JIT	Habilitat	ec
Compilador d'AOT (la JVM no utilitza AOT si no s'habiliten les classes compartides)	Habilitat	c
Opcions de depuració de JIT	Inhabilitat	c
Mida màxima Java2D dels tipus de lletra amb negreta algorítmica	14 punts	e
Mapes de bits representats Java2D en tipus de lletra escalables	Habilitat	e
Rasterització de tipus de lletra lliure Java2D	Habilitat	e
Tipus de lletra AWT Java2D	Inhabilitat	e
Entorn local per defecte	Cap	e
Temps d'espera abans d'iniciar el connector	zero	e
Directori temporal	/tmp	e
Redirecció de connector	Cap	e
Commutació d'IM	Inhabilitat	e
Modificadors d'IM	Inhabilitat	e
Model de fil	N/D	e
Mida de pila inicial per a fils Java de 32 bits. Utilitzeu: **-Xiss<mida>**	2 KB	c

Paràmetre de la JVM	Valor per defecte	Paràmetre afectat per
Mida de pila màxima per a fils Java de 32 bits. Utilitzeu: **-Xss<mida>**	256 KB	c
Mida de pila per a fils de sistema operatiu de 32 bits. Utilitzeu: **-Xmso<mida>**	256 KB	c
Mida d'emmagatzematge dinàmic inicial. Utilitzeu: **-Xms<mida>**	64 MB	c
Mida màxima d'emmagatzematge dinàmic Java. Utilitzeu: **-Xmx<mida>**	La meitat de la memòria disponible amb un mínim de 16 MB i un màxim de 512 MB	c
Utilització de l'interval de temps de destinació per a una aplicació. El recollidor de deixalles intenta utilitzar la resta. Utilitzeu **-Xgc:targetUtilization=<percentatge>**.	70%	c
Nombre de fils del recollidor de deixalles que cal executar. Utilitzeu **-Xgc:threads=<valor>**.	Nombre de nuclis de processador disponibles per al procés.	c
Quantitat màxima de memòria que es pot assignar a memòries amb àmbit al mode **-Xrealtime**. Utilitzeu **-Xgc:scopedMemoryMaximumSize=<mida>**.	8 MB	c
Defineix la mida de l'àrea de memòria immortal al mode **-Xrealtime**. Utilitzeu **-Xgc:immortalMemorySize=<mida>**.	16 MB	c

Nota: "memòria disponible" és la quantitat de memòria real (física) o el valor **RLIMIT_AS**, el que sigui el valor més baix.

Avisos

Aquesta informació s'ha desenvolupat per a productes i serveis oferts als EUA. És possible que IBM no ofereixi els productes, serveis o característiques que es mencionen dins aquest document en altres països. Consulteu el vostre representant local d'IBM si voleu obtenir més informació sobre els productes i els serveis disponibles actualment a la vostra zona. Qualsevol referència a un producte, programa o servei d'IBM no té la intenció de declarar o implicar que només es pot utilitzar aquell producte, programa o servei. En el seu lloc es podria utilitzar qualsevol producte, programa o servei equivalent que no infringeixi cap llei de propietat intel•lectual d'IBM. Tanmateix, és responsabilitat de l'usuari avaluar i verificar el funcionament de qualsevol producte, programa o servei que no sigui d'IBM.

IBM pot tenir patents o sol•licituds pendents de patent que tractin el tema d'aquest document. El fet de disposar d'aquest document no us dóna cap llicència sobre aquestes patents. Podeu enviar per escrit al fabricant les consultes referents a les llicències.

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
EUA

Per fer consultes sobre llicències pel que fa a la informació de DBCS (doble byte), poseu-vos en contacte amb el Departament de Propietat Intel•lectual d'IBM del vostre país, o envieu les consultes, per escrit, a:

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

El paràgraf següent no s'aplica al Regne Unit ni a cap altre país on aquestes disposicions entrin en conflicte amb la legislació local.

INTERNATIONAL BUSINESS MACHINES CORPORATION PROPORCIONA AQUESTA PUBLICACIÓ "TAL QUAL" SENSE CAP GARANTIA, NI EXPLÍCITA NI IMPLÍCITA, INCLOENT-HI, ENTRE D'ALTRES, LES GARANTIES RELATIVES AL NO INFRINGIMENT, A LA COMERCIALITZACIÓ O A L'ADEQUACIÓ PER A UNA FINALITAT DETERMINADA. Alguns països no permeten la renúncia de les garanties implícites o explícites en determinades transaccions i, per tant, pot ser que el paràgraf anterior no s'apliqui en el vostre cas.

Pot ser que la publicació inclogui incorreccions tècniques o errors tipogràfics. Es realitzaran modificacions periòdiques pel que fa a la informació de la publicació; aquestes modificacions s'incorporaran a les noves edicions de la informació. IBM pot efectuar millores i/o canvis en els productes i/o programes descrits en aquesta informació en qualsevol moment sense cap avís previ.

Les referències que apareixen en aquesta documentació a llocs Web que no són d'IBM es proporcionen a tall informatiu i de cap manera no es pretén aprovar aquests llocs Web. El material d'aquests llocs web no forma part del material per a aquest producte d'IBM i la utilització d'aquests llocs web és responsabilitat de l'usuari.

IBM pot utilitzar o distribuir la informació que proporcioneu de la manera que consideri oportuna sense que, per això, incorri en cap obligació envers el client.

Els usuaris que hagin rebut llicència per a aquest programa que vulguin obtenir informació per a habilitar (i) l'intercanvi d'informació entre programes creats de manera independent i altres programes (incloent-hi aquest) i (ii) la utilització mútua de la informació que s'ha intercanviat, s'hauran de posar en contacte amb:

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

Aquesta informació pot estar disponible, segons les condicions corresponents, incloent-hi, en alguns casos, el pagament d'una tarifa.

El programa amb llicència que es descriu en aquest document i tot el material amb llicència disponible per a aquest programa els proporciona IBM sota les condicions del Contracte de client d'IBM, el Contracte de llicència de programa internacional d'IBM, o qualsevol altre contracte equivalent.

Les dades de rendiment contingudes aquí s'han obtingut en un entorn controlat. Per tant, els resultats obtinguts en altres sistemes operatius poden variar significativament. És possible que algunes mesures s'hagin realitzat en sistemes a nivell de desenvolupament; no hi ha cap garantia que aquestes mesures siguin iguals en sistemes disponibles per a tothom. A més, és possible que algunes mesures s'hagin calculat mitjançant l'extrapolació. Els resultats reals poden variar. Els usuaris d'aquest document haurien de verificar les dades aplicables al seu entorn específic.

La informació relacionada amb productes que no són d'IBM s'ha obtingut dels proveïdors d'aquests productes, dels seus anuncis publicats o d'altres fonts disponibles públicament. IBM no ha provat aquests productes i no pot confirmar la precisió del rendiment, la compatibilitat ni cap altra reclamació relacionada amb els productes que no són d'IBM. Les preguntes relacionades amb les capacitats dels productes que no són d'IBM s'hauran de dirigir als proveïdors d'aquests productes.

Consideracions de la política de privadesa

Els productes d'IBM Software, incloent les solucions de programari com a servei, (“Ofertes de programari”) poden utilitzar galetes o altres tecnologies per recollir informació de l'ús del producte, per ajudar a millorar l'experiència de l'usuari final, per adaptar les interaccions amb l'usuari final o per altres objectius. En molts casos, les Ofertes de Programari no recopilen informació d'identificació personal. En alguns casos, aquestes Ofertes us ajuden a recopilar informació d'identificació personal. En el cas que utilitzin galetes per recopilar aquest tipus d'informació, tot seguit s'indica informació específica sobre l'ús de les galetes que fan aquestes ofertes.

Aquesta Oferta de Programari no utilitza cookies o altres tecnologies per recopilar informació d'identificació personal.

Si les configuracions desplegades per a aquesta Oferta de programari us proporciona com a client la possibilitat de recollir informació personalment

identificable dels usuaris finals mitjançant galetes i altres tecnologies, heu de cercar assessorament legal sobre les lleis vigents en relació a aquesta recollida de dades, incloent tots els requisits d'avís i consentiment.

Per obtenir més informació sobre l'ús de diverses tecnologies, per exemple les galetes, per a aquest objectiu, consulteu: (i) La Política de privades d'IBM a http://www.ibm.com/privacy ; (ii) la Declaració de privadesa en línia d'IBM a http://www.ibm.com/privacy/details (en concret, la secció titulada “Galetes, balises web i altres tecnologies"); i (iii) la Declaració de la política de productes de programari i de programari com a servei d'IBM a http://www.ibm.com/software/info/product-privacy.

Marques registrades

IBM, el logotip d'IBM i ibm.com són marques comercials o marques registrades d'International Business Machines Corporation als Estats Units o a altres països. Si aquests i altres termes de marques registrades d'IBM estan marcats amb un símbol de marca registrada (® o ™) la primera vegada que apareixen, aquests símbols indiquen marques comercials o marques registrades dels EUA que són propietat d'IBM en el moment de publicar-se aquesta informació. Aquestes marques comercials també poden ser marques comercials o registrades a altres països. Una llista actualitzada de marques registrades d'IBM està disponible a la web a l'apartat "Informació de copyright i marques registrades" de http://www.ibm.com/legal/copytrade.shtml.

Adobe, el logotip d'Adobe logo, PostScript i el logotip de PostScript són marques registrades d'Adobe Systems Incorporated als Estats Units i/o a altres països.

Intel i Itanium són marques registrades d'Intel Corporation o de les seves filials als Estats Units i a altres països.

Linux és una marca registrada de Linus Torvalds als Estats Units i/o a altres països.

Java i totes les marques registrades i logotips basats en Java són marques comercials o marques registrades d'Oracle o els seus afiliats.

Altres noms d'empreses, productes o serveis poden ser marques registrades o marques de serveis d'altres empreses.

Índex

Caràcters Especials

A

B

C

D

E

F

G

H

I

J

L

M

O

P

R

S

T

U

V

Imprès a Espanya