



JIPDay Roma 30 Settembre 2005

Tuning, Performances & Problem Solving sugli Application Server

Fabrizio Marini

fabrizio.marini@bea.com



Problema:

Spesso quando un Application Server risponde lentamente o si blocca si da subito la colpa al produttore, ma ci sono molte cose che devono essere controllate e fatte prima durante e dopo l'installazione, vediamole e commentiamole insieme



Controlli sul Sistema Operativo

- Il sistema operativo DEVE essere aggiornato con gli opportuni livelli di Patches
- Se si lavora in ambiente CLUSTER le macchine devono avere gli stessi livelli di Patches



Controlli sul JDK

- Identificare il JDK in utilizzo (verificare che rientri nelle release notes dell'AS in uso)
- Riconoscere il livello di PATCHES per Java
- Identificare il corretto livello di Patches per specifico JDK (Molti crash della JVM dipendono dalle Patches)
- Alcune PATCHES sono specifiche per combinazioni fra Sistema Operativo e JDK utilizzato



Esempio di Patches su HP:

HP-UX	Java 1.1	Java 1.2	Java 1.3	Java 1.4	Java 1.5
10.20 PA	Base 10.20 (5)	not supported	not supported	not supported	not supported
11.00 PA	March '04 (none) Sept '03 (none) March '03 (none) Sept '02 (none) March '02 (none) Sept '01 (1) Base 11.00 (7)	March '04 (1) Sept '03 (1) March '03 (1) Sept '02 (2) March '02 (2) Sept '01 (4) Base 11.00 (10)	March '04 (2) Sept '03 (3) March '03 (4) Sept '02 (6) March '02 (7) Sept '01 (12) Base 11.00 (26)	March '04 (3) Sept '03 (4) March '03 (6) Sept '02 (8) March '02 (9) Sept '01 (15) Base 11.00 (31)	not supported
11.11 PA (11iv1)		Dec '04 (none) June '04 (none) Dec '03 (none) June '03 (none) Dec '02 (1) June '02 (1) Dec '01 (2) June '01 (2) Base 11.11 (2)	Dec '04 (1) June '04 (3) Dec '03 (4) June '03 (4) Dec '02 (6) June '02 (8) Dec '01 (19) June '01 (26) Base 11.11 (26)	Dec '04 (3) June '04 (6) Dec '03 (7) June '03 (8) Dec '02 (10) June '02 (13) Dec '01 (25) June '01 (32) Base 11.11 (32)	Dec '04 (3) June '04 (6) Dec '03 (7) June '03 (8) Dec '02 (10) June '02 (13) Dec '01 (25) June '01 (32) Base 11.11 (32)
11.20 IA (11iv1.5)			Base 11.20 (11)	Base 11.20 (11)	not supported
11.22 IA (11iv1.6)			Base 11.22 (2)	Base 11.22 (2)	not supported
HP-UX 11.23 IA			March '04 (3) Base 11.23 (3)	March '04 (4) Base 11.23 (4)	March '04 (4) Base 11.23 (4)
11.23 IA (11iv2)					



Esempio di Patches su SUN:

[Java](#) | [Solaris](#) | [Communities](#) | [Partners](#) | [My Sun](#) | [Sun Store](#)

[United States](#) | [Worldwide](#)



[Products](#) | [Downloads](#) | [Services & Solutions](#) | [Support](#) | [Training](#) | [Research](#)

[Home](#) > [Support](#) >



Patches & Updates from the Sun Update Connection

Deploy and monitor updates to all of your systems through Sun from anywhere you have an Internet connection.

Start today with the [Intelligent Software Update Services](#)

Sun support customers: more support information is available after you sign in.

User Name

Password

[Sign In](#) » Standard

» [Secure Login](#)

» [Register](#)

» [Forgot Username?](#)

» [Forgot your Password?](#)

J2SE Cluster Patches for Solaris

Cluster Name	Updated	Download	Readme	Size
J2SE Solaris 10	Sep/14/05	HTTP FTP	Readme	1.3M
J2SE Solaris 10 x86	Sep/19/05	HTTP FTP	Readme	2.6M
J2SE Solaris 9	Sep/19/05	HTTP FTP	Readme	41.0M
J2SE Solaris 9 x86	Sep/19/05	HTTP FTP	Readme	5.8M
J2SE Solaris 8	Sep/22/05	HTTP FTP	Readme	135.4M
J2SE Solaris 8 x86	Sep/22/05	HTTP FTP	Readme	32.8M
J2SE Solaris 7	Jul/26/05	HTTP FTP	Readme	36.2M
J2SE Solaris 7 x86	Jul/26/05	HTTP FTP	Readme	18.7M
J2SE Solaris 2.6	Mar/23/05	HTTP FTP	Readme	28.8M
J2SE Solaris 2.6 x86	Mar/23/05	HTTP FTP	Readme	18.5M
J2SE Solaris 2.5.1	Mar/22/05	HTTP FTP	Readme	14.5M
J2SE Solaris 2.5.1 x86	Mar/22/05	HTTP FTP	Readme	8.2M

Support

» [Patches and Updates](#)

» [Support Forums](#)

» [Security Resources](#)

» [System Admin
Community](#)

» [Sun System Handbook](#)



Esempio di Patches su IBM

For AIX 4.3.3, Java 1.3.1 requires the **AIX 4330-10 Recommended** Maintenance Level. This maintenance package is intended for customers who already have AIX 4.3.3 installed. The AIX 4330-10 maintenance package can be downloaded from <http://techsupport.services.ibm.com/server/fixes/>, using **APAR number IY28107**. If you are a licensee of AIX 4.3.3, you can obtain an Update CD by contacting your point of sale and requesting feature code 0838.

Recommended Fixes: IY40811: Prevents a hang/deadlock in the JVM whilst performing informational calls to setlocale by proceeding without taking a lock.



Controllare TCP/IP

- Controllare i valori del TCP/IP
- Verificare il numero di CLOSE WAIT
- Connessioni “appese”
- Configurazione del numero dei file descrittori



Esempio valori TCP/IP

Parameter	Customer Actual Value	Vendor Suggested Value
tcp_time_wait_interval	15000	60000
tcp_conn_req_max_q	not HP UX support	16384
tcp_conn_req_max_q0	not HP UX support	16384
tcp_ip_abort_interval	600000	60000
tcp_keepalive_interval	60000	7200000
tcp_rexmit_interval_initial	3000	4000
tcp_rexmit_interval_max	60000	10000
tcp_rexmit_interval_min	500	3000
tcp_smallest_anon_port	49152	32768
tcp_xmit_hiwat	not HP UX support	131072
tcp_recv_hiwat	not HP UX support	131072
tcp_naglim_def	65535	1



Ora possiamo installare ...

- Installiamo l'Application Server
- Installiamo le Service Packs richieste dall'AS
- Convinciamo il cliente a mettere le ultime Service Packs 😊



Alcuni controlli:

- Abilitare o disabilitare a seconda dei casi il Production Mode
- Controllare il CLASSPATH
 - Ordine elementi
 - Posizione Patches
 - Riconoscere le librerie di Default
 - Attenzione ai driver JDBC



Controlli relativi al DB:

- Verificare il tipo di driver JDBC che si utilizza, per esempio leggendo il file MANIFEST
- Dimensionare bene i pool di connessione, soprattutto rispetto ai Thread di esecuzione
- Verificare i parametri per XA Driver, importanti sono i TimeOut (JTA < Driver < DB) per evitare errori tipo XA ERR NOTA



Controlli sui descrittori degli EJB:

- Dimensionamento del Pool degli EJB
- Evitare che gli ejb in pool crescano a dismisura
- Impostiamo valori come i MAX Bean in Cache
- Gli MDB se non sono settati possono assumere come numero MAX il numero degli Execute Thread (o la metà + 1)



Execute Thread degli AS:

- Verificarne bene la corretta dimensione (magari dopo opportuni stress Test)
- Suddividiamoli per tipologie di protocollo (HTML, RMI) e/o di lavoro
- Usiamo il KILL -3



KILL -3

```
"ExecuteThread: '7' for queue: 'JmsDispatcher'" daemon prio=5 tid=0x64af70 nid=0x34
  waiting on monitor [0xb1d81000..0xb1d819d8]
  at java.lang.Object.wait(Native Method)
  at java.lang.Object.wait(Object.java:415)
  at weblogic.kernel.ExecuteThread.waitForRequest(ExecuteThread.java:95)
  at weblogic.kernel.ExecuteThread.run(ExecuteThread.java:119)
```

```
"ExecuteThread: '6' for queue: 'JmsDispatcher'" daemon prio=5 tid=0x64ae30 nid=0x33
  waiting on monitor [0xb1e81000..0xb1e819d8]
  at java.lang.Object.wait(Native Method)
  at java.lang.Object.wait(Object.java:415)
  at weblogic.kernel.ExecuteThread.waitForRequest(ExecuteThread.java:95)
  at weblogic.kernel.ExecuteThread.run(ExecuteThread.java:119)
```

```
"ExecuteThread: '5' for queue: 'JmsDispatcher'" daemon prio=5 tid=0x51f6a0 nid=0x32
  waiting on monitor [0xb1f81000..0xb1f819d8]
  at java.lang.Object.wait(Native Method)
  at java.lang.Object.wait(Object.java:415)
  at weblogic.kernel.ExecuteThread.waitForRequest(ExecuteThread.java:95)
  at weblogic.kernel.ExecuteThread.run(ExecuteThread.java:119)
```



Analisi dei Thread Dump:

Samurai

File Edit Help

log Thread Dumps

ListenThread

- JmsDispatcher*{14}
- JmsDispatcher*{13}
- JmsDispatcher*{12}
- JmsDispatcher*{11}
- JmsDispatcher*{10}
- JmsDispatcher*{9}
- JmsDispatcher*{8}
- JmsDispatcher*{7}
- JmsDispatcher*{6}
- JmsDispatcher*{5}
- JmsDispatcher*{4}
- JmsDispatcher*{3}
- JmsDispatcher*{2}
- JmsDispatcher*{1}
- JmsDispatcher*{0}
- weblogic.transaction.AsyncQueue*{2}
- weblogic.transaction.AsyncQueue*{1}
- weblogic.transaction.AsyncQueue*{0}
- Multicast*{0}

[TimeEventGenerator](#)

[_weblogic_dgc_queue](#)[1]

[_weblogic_dgc_queue](#)[0]

[SpinnerRandomSource](#)

[default](#)[14]

[default](#)[13]

[default](#)[12]

[default](#)[11]

[default](#)[10]

[default](#)[9]

[default](#)[8]

monitoring : "thread_dump"



Directory & File

- Controllare la Dimensione del File System.
- Ci deve essere spazio per i core dump
- Directory dedicate per LOG e Code persistenti



Non sottovalutiamo i Log:

- In produzione abbassare il livello di log
- Evitare di reindirizzare l'output su file system
- Non mettere `System.out()` nel codice
- Usare `log4j` ma non esageriamo.
- Durante gli stress test i livelli di log devono essere abbassati



JMS

- JMS: code persistenti su file system o su DB?
- Controllare le properties di JMS come:
 - Tempo di vita
 - Destinazione errori
 - Redelivery



CLUSTER

- Verifiche sul Multicast (237.0.0.1)
- Attivare il Load Balancing (HTTP, RMI)
- Fail Over
- Cosa serve fra DNS, bilanciatori Hardware e/o software
- Non facciamo soffrire JNDI



Configuriamo la memoria per la JVM

- Jdk131 - jdk141 – jdk 1.5
- Attenzione alle grosse quantità di memoria, spesso ci sono dei limiti e dei valori di sistema da dover impostare come la paginazione.
- Permanent size? (se l'applicazione richiede di elevarla troppo fermiamoci a pensare)



Impostiamo GC

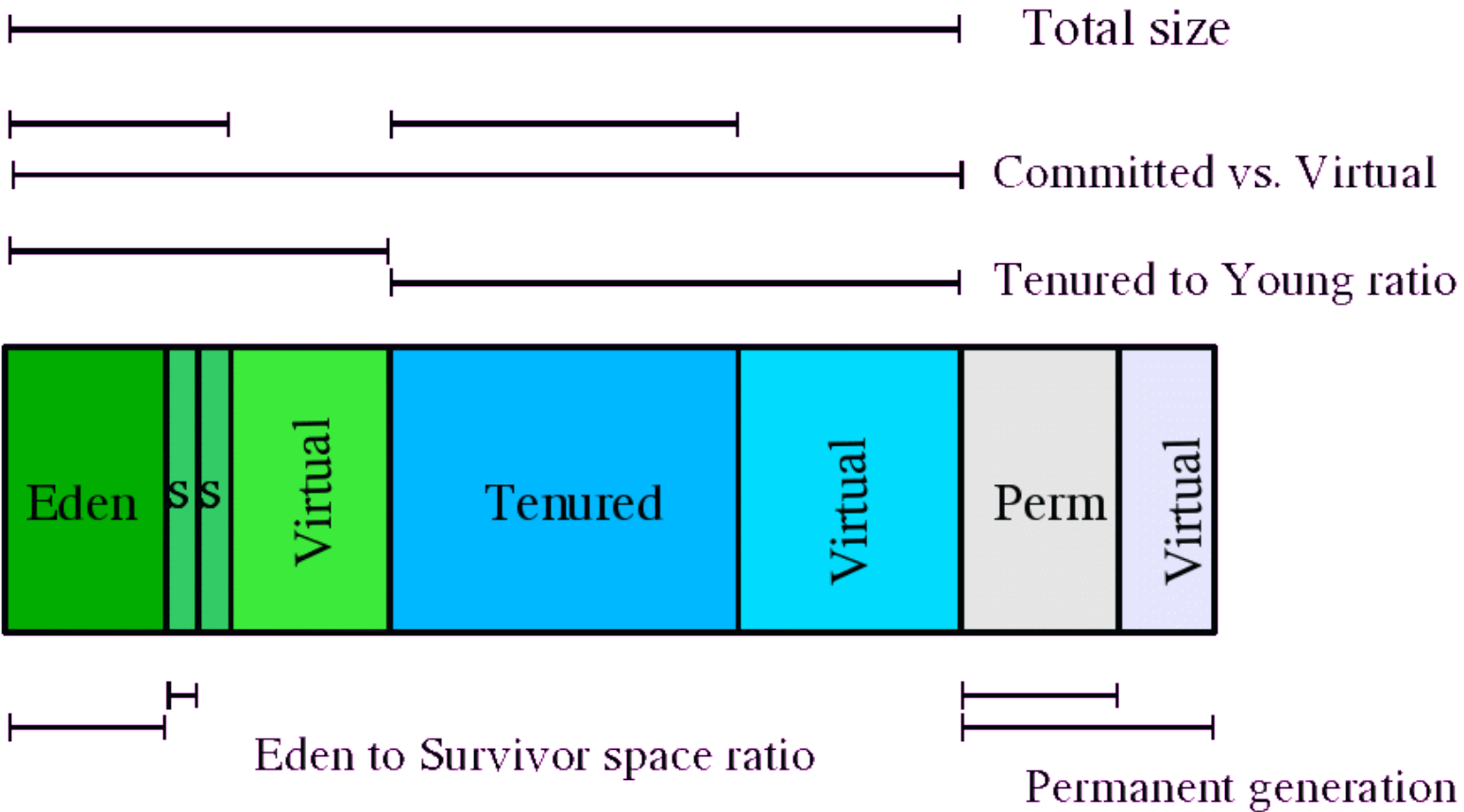
Types of Collectors

In the J2SE platform, version 1.4.2 there are three additional collectors. Each is a generational collector which has been implemented to emphasize the throughput of the application or low garbage collection pause times.

4. The *throughput* collector: this collector uses a parallel version of the *young* generation collector. It is used if the `-XX:+UseParallelGC` option is passed on the command line. The *tenured* generation collector is the same as the default collector.
5. The *concurrent* low pause collector: this collector is used if the `-XX:+UseConcMarkSweepGC` is passed on the command line. The concurrent collector is used to collect the *tenured* generation and does most of the collection concurrently with the execution of the application. The application is paused for short periods during the collection. A parallel version of the *young* generation copying collector is used with the concurrent collector if the combination `-XX:+UseConcMarkSweepGC -XX:+UseParNewGC` is passed on the command line.
6. The *incremental* (sometimes called *train*) low pause collector: this collector is used only if `-Xincgc` is passed on the command line. By careful bookkeeping, the incremental garbage collector collects just a portion of the *tenured* generation at each minor collection, trying to spread the large pause of a major collection over many minor collections. However, it is even slower than the default *tenured* generation collector when considering overall throughput.



Memoria JVM





Parametri da passare alla JVM

- Leggere sempre le release notes
- `-XX:+DisableExplicitGC`
- Attenzione a JNI
- Loop?
 - - `Xss<val>K`
 - - `XX:ThreadStackSize=<val>k`



Parliamo di Codice

- Web Application, JSP/Servlet e variabili di classe
- Web App & Multi Threading
- Singleton & Cluster
- Variabili Statiche
- EJB e Gestione Transazioni
- Sessioni ed oggetti Serializzabili
- Evitare stringhe scolpite nel codice
- Evitare file di properties
- XML Identificare il giusto parser
- EJB ed accesso al disco
- EJB e gestione dei Thread
- Utilizzo di classe Timer (Parliamo di CommonJ)
- Non settiamo Time Out nel codice in contesti XA



Specifiche JVM

1) <http://java.sun.com/docs/books/vmspec/2nd-edition/html/Concepts.doc.html#33308>

"If two or more concurrent threads act on a shared variable, there is a possibility that the actions on the variable will produce timing-dependent results."

2) <http://java.sun.com/docs/books/vmspec/2nd-edition/html/Threads.doc.html#21294>

8.9 Discussion

Any association between locks and variables is purely conventional. Locking any lock conceptually flushes all variables from a thread's working memory, and unlocking any lock forces the writing out to main memory of all variables that the thread has assigned. That a lock may be associated with a particular object or a class is purely a convention. For example, in some applications it may be appropriate always to lock an object before accessing any of its instance variables; synchronized methods are a convenient way to follow this convention. In other applications, it may suffice to use a single lock to synchronize access to a large collection of objects.

If a thread uses a particular shared variable only after locking a particular lock and before the corresponding unlocking of that same lock, then the thread will read the shared value of that variable from main memory after the lock operation, if necessary, and will copy back to main memory the value most recently assigned to that variable before the unlock operation. This, in conjunction with the mutual exclusion rules for locks, suffices to guarantee that values are correctly transmitted from one thread to another through shared variables.



E poi ...

- Utilizziamo Top, controlliamo Ram e CPU
- JMX
- JVMStat
- Jrat



Domande ?