Java HotSpot VM Options

Please note that this page only applies to JDK 7 and earlier releases. For JDK 8 please see the Windows, Solaris reference pages.

This document provides information on typical command-line options and environment variables that can affect the performance characteristics of the Java HotSpot Virtual Machine. Unless otherwise noted, all information in this document pertains to both the Java HotSpot Client VM and the Java HotSpot Server VM.

Categories of Java HotSpot VM Options

Standard options recognized by the Java HotSpot VM are described on the Java Application Launcher reference pages for Windows and Solaris & Linux. This document deals exclusively with non-standard options recognized by the Java HotSpot

VM:

- Options that begin with `-X` are non-standard (not guaranteed to be supported on all VM implementations), and are subject to change without notice in subsequent releases of the JDK.
- Options that are specified with `-XX` are not stable and are subject to change without notice.

Users of JDKs older than 1.3.0 who wish to port to a Java HotSpot VM, should see Java HotSpot Equivalents of Exact VM flags.

Some Useful -XX Options

Default values are listed for Java SE 6 for Solaris Sparc with -server. Some options may vary per architecture/OS/JVM version. Platforms with a differing default value are listed in the description.

- Boolean options are turned on with `-XX:+<option>` and turned off with `-XX:-<option>`.Disa
- Numeric options are set with `-XX:<option>=<number>`. Numbers can include 'm' or 'M' for megabytes, 'k' or 'K' for kilobytes, and 'g' or 'G' for gigabytes (for example, 32k is the same as 32768).
- String options are set with `-XX:<option>=<string>`, are usually used to specify a file, a path, or a list of commands

Flags marked as *manageable* are dynamically writeable through the JDK management interface (com.sun.management.HotSpotDiagnosticMXBean API) and also through JConsole. In Monitoring and Managing Java SE 6 Platform Applications, Figure 3 shows an example. The manageable flags can also be set through jinfo -flag. The options below are loosely grouped into categories.

Behavioral options change the basic behavior of the VM.

Garbage First (G1) Garbage Collection Options

Performance tuning options are knobs which can be used to tune VM performance.

Debugging options generally enable tracing, printing, or output of VM information.

Behavioral Options

Option and Default Value	Description
-XX:-AllowUserSignalHandlers	Do not complain if the application installs signal handlers. (Relevant to Solaris and Linux only.)
-XX:AltStackSize=16384	Alternate signal stack size (in Kbytes). (Relevant to Solaris only, removed from 5.0.)
-XX:-DisableExplicitGC	By default calls to System.gc() are enabled (-XX:-DisableExplicitGC). Use -XX:+DisableExplicitGC to disable calls to System.gc(). Note that the JVM still performs garbage collection when necessary.
-XX:+FailOverToOldVerifier	Fail over to old verifier when the new type checker fails. (Introduced in 6.)
-XX:+HandlePromotionFailure	The youngest generation collection does not require a guarantee of full promotion of all live objects. (Introduced in 1.4.2 update 11) [5.0 and earlier: false.]
-XX:+MaxFDLimit	Bump the number of file descriptors to max. (Relevant to Solaris only.)
-XX:PreBlockSpin=10	Spin count variable for use with -XX:+UseSpinning. Controls the maximum spin iterations allowed

	before entering operating system thread synchronization code. (Introduced in 1.4.2.)
-XX:-RelaxAccessControlCheck	Relax the access control checks in the verifier. (Introduced in 6.)
-XX:+ScavengeBeforeFullGC	Do young generation GC prior to a full GC. (Introduced in 1.4.1.)
-XX:+UseAltSigs	Use alternate signals instead of SIGUSR1 and SIGUSR2 for VM internal signals. (Introduced in 1.3.1 update 9, 1.4.1. Relevant to Solaris only.)
-XX:+UseBoundThreads	Bind user level threads to kernel threads. (Relevant to Solaris only.)
-XX:-UseConcMarkSweepGC	Use concurrent mark-sweep collection for the old generation. (Introduced in 1.4.1)
-XX:+UseGCOverheadLimit	Use a policy that limits the proportion of the VM's time that is spent in GC before an OutOfMemory error is thrown. (Introduced in 6.)
-XX:+UseLWPSynchronization	Use LWP-based instead of thread based synchronization. (Introduced in 1.4.0. Relevant to Solaris only.)
-XX:-UseParallelGC	Use parallel garbage collection for scavenges. (Introduced in 1.4.1)

-XX:-UseParallelOldGC	Use parallel garbage collection for the full collections. Enabling this option automatically sets -XX:+UseParallelGC. (Introduced in 5.0 update 6.)
-XX:-UseSerialGC	Use serial garbage collection. (Introduced in 5.0.)
-XX:-UseSpinning	Enable naive spinning on Java monitor before entering operating system thread synchronizaton code. (Relevant to 1.4.2 and 5.0 only.) [1.4.2, multi-processor Windows platforms: true]
-XX:+UseTLAB	Use thread-local object allocation (Introduced in 1.4.0, known as UseTLE prior to that.) [1.4.2 and earlier, x86 or with -client: false]
-XX:+UseSplitVerifier	Use the new type checker with StackMapTable attributes. (Introduced in 5.0.)[5.0: false]
-XX:+UseThreadPriorities	Use native thread priorities.
-XX:+UseVMInterruptibleIO	Thread interrupt before or with EINTR for I/O operations results in OS_INTRPT. (Introduced in 6. Relevant to Solaris only.)

Garbage First (G1) Garbage Collection Options

Option and Default Value	**Description**
-XX:+UseG1GC	Use the Garbage First (G1) Collector
-XX:MaxGCPauseMillis=n	Sets a target for the maximum GC pause time. This is a soft goal, and the JVM will make its best effort to achieve it.
-XX:InitiatingHeapOccupancyPercent=n	Percentage of the (entire) heap occupancy to start a concurrent GC cycle. It is used by GCs that trigger a concurrent GC cycle based on the occupancy of the entire heap, not just one of the generations (e.g., G1). A value of 0 denotes 'do constant GC cycles'. The default value is 45.
-XX:NewRatio=n	Ratio of old/new generation sizes. The default value is 2.
-XX:SurvivorRatio=n	Ratio of eden/survivor space size. The default value is 8.
-XX:MaxTenuringThreshold=n	Maximum value for tenuring threshold. The default value is 15.
-XX:ParallelGCThreads=n	Sets the number of threads used during parallel phases of the garbage collectors. The default value varies with the platform on which the JVM is running.

-XX:ConcGCThreads=n	Number of threads concurrent garbage collectors will use. The default value varies with the platform on which the JVM is running.
-XX:G1ReservePercent=n	Sets the amount of heap that is reserved as a false ceiling to reduce the possibility of promotion failure. The default value is 10.
-XX:G1HeapRegionSize=n	With G1 the Java heap is subdivided into uniformly sized regions. This sets the size of the individual sub-divisions. The default value of this parameter is determined ergonomically based upon heap size. The minimum value is 1Mb and the maximum value is 32Mb.

Performance Options

Option and Default Value	**Description**
-XX:+AggressiveOpts	Turn on point performance compiler optimizations that are expected to be default in upcoming releases. (Introduced in 5.0 update 6.)
-XX:CompileThreshold=10000	Number of method invocations/branches before compiling [-client: 1,500]

-XX:LargePageSizeInBytes=4m	Sets the large page size used for the Java heap. (Introduced in 1.4.0 update 1.) [amd64: 2m.]
-XX:MaxHeapFreeRatio=70	Maximum percentage of heap free after GC to avoid shrinking.
-XX:MaxNewSize=size	Maximum size of new generation (in bytes). Since 1.4, MaxNewSize is computed as a function of NewRatio. [1.3.1 Sparc: 32m; 1.3.1 x86: 2.5m.]
-XX:MaxPermSize=64m	Size of the Permanent Generation. [5.0 and newer: 64 bit VMs are scaled 30% larger; 1.4 amd64: 96m; 1.3.1 -client: 32m.]
-XX:MinHeapFreeRatio=40	Minimum percentage of heap free after GC to avoid expansion.
-XX:NewRatio=2	Ratio of old/new generation sizes. [Sparc -client: 8; x86 -server: 8; x86 -client: 12.]-client: 4 (1.3) 8 (1.3.1+), x86: 12]
-XX:NewSize=2m	Default size of new generation (in bytes) [5.0 and newer: 64 bit VMs are scaled 30% larger; x86: 1m; x86, 5.0 and older: 640k]
-XX:ReservedCodeCacheSize=32m	Reserved code cache size (in bytes) - maximum code cache size. [Solaris 64-bit, amd64, and -server x86: 2048m; in 1.5.0_06 and earlier, Solaris 64-bit and amd64: 1024m.]

-XX:SurvivorRatio=8	Ratio of eden/survivor space size [Solaris amd64: 6; Sparc in 1.3.1: 25; other Solaris platforms in 5.0 and earlier: 32]
-XX:TargetSurvivorRatio=50	Desired percentage of survivor space used after scavenge.
-XX:ThreadStackSize=512	Thread Stack Size (in Kbytes). (0 means use default stack size) [Sparc: 512; Solaris x86: 320 (was 256 prior in 5.0 and earlier); Sparc 64 bit: 1024; Linux amd64: 1024 (was 0 in 5.0 and earlier); all others 0.]
-XX:+UseBiasedLocking	Enable biased locking. For more details, see this tuning example. (Introduced in 5.0 update 6.) [5.0: false]
-XX:+UseFastAccessorMethods	Use optimized versions of Get<Primitive>Field.
-XX:-UseISM	Use Intimate Shared Memory. [Not accepted for non-Solaris platforms.]
-XX:+UseLargePages	Use large page memory. (Introduced in 5.0 update 5.) For details, see Java Support for Large Memory Pages.
-XX:+UseMPSS	Use Multiple Page Size Support w/4mb pages for the heap. Do not use with ISM as this replaces the need for ISM. (Introduced in 1.4.0 update 1, Relevant to Solaris 9 and newer.) [1.4.1 and earlier: false]

-XX:+UseStringCache	Enables caching of commonly allocated strings.
-XX:AllocatePrefetchLines=1	Number of cache lines to load after the last object allocation using prefetch instructions generated in JIT compiled code. Default values are 1 if the last allocated object was an instance and 3 if it was an array.
-XX:AllocatePrefetchStyle=1	Generated code style for prefetch instructions. 0 - no prefetch instructions are generate*d*, 1 - execute prefetch instructions after each allocation, 2 - use TLAB allocation watermark pointer to gate when prefetch instructions are executed.
-XX:+UseCompressedStrings	Use a byte[] for Strings which can be represented as pure ASCII. (Introduced in Java 6 Update 21 Performance Release)
-XX:+OptimizeStringConcat	Optimize String concatenation operations where possible. (Introduced in Java 6 Update 20)

Debugging Options

Option and Default Value	**Description**

-XX:-CITime	Prints time spent in JIT Compiler. (Introduced in 1.4.0.)
-XX:ErrorFile=./hs_err_pid<pid>.log	If an error occurs, save the error data to this file. (Introduced in 6.)
-XX:-ExtendedDTraceProbes	Enable performance-impacting dtrace probes. (Introduced in 6. Relevant to Solaris only.)
-XX:HeapDumpPath=./java_pid<pid>.hprof	Path to directory or filename for heap dump. *Manageable*. (Introduced in 1.4.2 update 12, 5.0 update 7.)
-XX:-HeapDumpOnOutOfMemoryError	Dump heap to file when java.lang.OutOfMemoryError is thrown. *Manageable*. (Introduced in 1.4.2 update 12, 5.0 update 7.)
-XX:OnError="<cmd args>;<cmd args>"	Run user-defined commands on fatal error. (Introduced in 1.4.2 update 9.)
-XX:OnOutOfMemoryError="<cmd args>; <cmd args>"	Run user-defined commands when an OutOfMemoryError is first thrown. (Introduced in 1.4.2 update 12, 6)
-XX:-PrintClassHistogram	Print a histogram of class instances on Ctrl-Break. *Manageable*. (Introduced in 1.4.2.) The jmap -histo command provides equivalent functionality.

-XX:-PrintConcurrentLocks	Print java.util.concurrent locks in Ctrl-Break thread dump. *Manageable*. (Introduced in 6.) The jstack -l command provides equivalent functionality.
-XX:-PrintCommandLineFlags	Print flags that appeared on the command line. (Introduced in 5.0.)
-XX:-PrintCompilation	Print message when a method is compiled.
-XX:-PrintGC	Print messages at garbage collection. *Manageable*.
-XX:-PrintGCDetails	Print more details at garbage collection. *Manageable*. (Introduced in 1.4.0.)
-XX:-PrintGCTimeStamps	Print timestamps at garbage collection. *Manageable* (Introduced in 1.4.0.)
-XX:-PrintTenuringDistribution	Print tenuring age information.
-XX:-PrintAdaptiveSizePolicy	Enables printing of information about adaptive generation sizing.
-XX:-TraceClassLoading	Trace loading of classes.
-XX:-TraceClassLoadingPreorder	Trace all classes loaded in order referenced (not loaded). (Introduced in 1.4.2.)

-XX:-TraceClassResolution	Trace constant pool resolutions. (Introduced in 1.4.2.)
-XX:-TraceClassUnloading	Trace unloading of classes.
-XX:-TraceLoaderConstraints	Trace recording of loader constraints. (Introduced in 6.)
-XX:+PerfDataSaveToFile	Saves jvmstat binary data on exit.
-XX:ParallelGCThreads=n	Sets the number of garbage collection threads in the young and old parallel garbage collectors. The default value varies with the platform on which the JVM is running.
-XX:+UseCompressedOops	Enables the use of compressed pointers (object references represented as 32 bit offsets instead of 64-bit pointers) for optimized 64-bit performance with Java heap sizes less than 32gb.
-XX:+AlwaysPreTouch	Pre-touch the Java heap during JVM initialization. Every page of the heap is thus demand-zeroed during initialization rather than incrementally during application execution.
-XX:AllocatePrefetchDistance=n	Sets the prefetch distance for object allocation. Memory about to be written with the value of new objects is prefetched into cache at this distance (in bytes) beyond the address of the last allocated

	object. Each Java thread has its own allocation point. The default value varies with the platform on which the JVM is running.
-XX:InlineSmallCode=n	Inline a previously compiled method only if its generated native code size is less than this. The default value varies with the platform on which the JVM is running.
-XX:MaxInlineSize=35	Maximum bytecode size of a method to be inlined.
-XX:FreqInlineSize=n	Maximum bytecode size of a frequently executed method to be inlined. The default value varies with the platform on which the JVM is running.
-XX:LoopUnrollLimit=n	Unroll loop bodies with server compiler intermediate representation node count less than this value. The limit used by the server compiler is a function of this value, not the actual value. The default value varies with the platform on which the JVM is running.
-XX:InitialTenuringThreshold=7	Sets the initial tenuring threshold for use in adaptive GC sizing in the parallel young collector. The tenuring threshold is the number of times an object survives a young collection before being promoted to the old, or tenured, generation.
-XX:MaxTenuringThreshold=n	Sets the maximum tenuring threshold for use in adaptive GC sizing. The current largest value is 15. The default value is 15 for the parallel collector and

	is 4 for CMS.
-Xloggc:<filename>	Log GC verbose output to specified file. The verbose output is controlled by the normal verbose GC flags.
-XX:-UseGCLogFileRotation	Enabled GC log rotation, requires -Xloggc.
-XX:NumberOfGClogFiles=1	Set the number of files to use when rotating logs, must be >= 1. The rotated log files will use the following naming scheme, <filename>.0, <filename>.1, ..., <filename>.n-1.
-XX:GCLogFileSize=8K	The size of the log file at which point the log will be rotated, must be >= 8K.

Resources for

Developers

Startups

Students and Educators

Partners

Oracle PartnerNetwork

Find a Partner

Log in to OPN

Solutions

Artificial Intelligence

Internet of Things

Blockchain

What's New

Oracle's response to COVID-19

Java SE14 download

Try Oracle Cloud Free Tier

Contact Us

US Sales: +1.800.633.0738

How can we help?

Subscribe to emails

Country/Region © 2020 Oracle Site Map Privacy / Do Not Sell My Info Cookie 喜好设置 Ad Choices Careers