1. [**JVM 参数大全**](http://dasheng.iteye.com/blog/1114639)

**Behavioral Options（行为参数）**

Option and Default Value Description

|  |
| --- |
|  |
| -XX:-AllowUserSignalHandlers | Do not complain if the application installs signal handlers. (Relevant to Solaris and Linux only.)  允许应用程序安装信号处理器（仅在solaris和linux起作用。） |
| -XX:AltStackSize=16384 | Alternate signal stack size (in Kbytes). (Relevant to Solaris only, removed from 5.0.)  备用信号栈大小（仅在solaris起作用，从5.0起已经删除） |
| -XX:-DisableExplicitGC | Disable calls to System.gc(), JVM still performs garbage collection when necessary.  禁止调用System.gc(), JVM 依然在必要时执行垃圾回收。 |
| -XX:+FailOverToOldVerifier | Fail over to old verifier when the new type checker fails. (Introduced in 6.)  当新的类型检查器失败时转移到旧的类型检查器上（从6.0开始引进） |
| -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.]  最年轻一代的垃圾收集不需要保证所有的活动对象完全升级。 |
| -XX:+MaxFDLimit | Bump the number of file descriptors to max. (Relevant  to Solaris only.)  文件描述符设置到最大。（只与solaris相关） |
| -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.)  在启用-XX:+UseSpinning 的情况下，设置旋转计数变量，来控制进入系统的线程同步代码之前，允许的最大旋转迭代次数。（在1.4.2引入） |
| -XX:-RelaxAccessControlCheck | Relax the access control checks in the verifier. (Introduced in 6.)  在验证器里放松访问控制的检查。（在6.0引入） |
| -XX:+ScavengeBeforeFullGC | Do young generation GC prior to a full GC. (Introduced in 1.4.1.)  最年轻一代GC优先于全局GC（在1.4.1引入） |
| -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.)  用备份信号代替JVM的SIGUSR1 和 SIGUSR2 （只和solaris相关） |
| -XX:+UseBoundThreads | Bind user level threads to kernel threads. (Relevant to Solaris only.)  绑定用户级别的线程到内核线程。（只与solaris有关） |
| -XX:-UseConcMarkSweepGC | Use concurrent mark-sweep collection for the old generation. (Introduced in 1.4.1)  对老一代对象区使用同步标记并摖除垃圾收集算法（从1.4.1引进） |
| -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.)  在OutOfMemory错误抛出之前，采用限制花费在GC上的时间在VM比例幅度的策略。（在6.0引入） |
| -XX:+UseLWPSynchronization | Use LWP-based instead of thread based synchronization. (Introduced in 1.4.0. Relevant to Solaris only.)  用基于LWP的同步来替代基于线程的同步。（从1.4.0引入，只和solaris相关） |
| -XX:-UseParallelGC | Use parallel garbage collection for scavenges. (Introduced in 1.4.1)  用并行垃圾收集的算法来清除对象。（从1.4.1起引入） |
| -XX:-UseParallelOldGC | Use parallel garbage collection for the full collections. Enabling this option automatically sets -XX:+UseParallelGC. (Introduced in 5.0 update 6.)  对整个收集使用并行垃圾收集算法，并自动设置-XX:+UseParallelGC。（从5.0引入升级到6.0） |
| -XX:-UseSerialGC | Use serial garbage collection. (Introduced in 5.0.)  使用系列垃圾收集算法（从5.0起引入） |
| -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]  在进入系统线程同步代码之前在JAVA监视器上使用原生的旋转。  （只与1.4.2和5.0相关，1.4.2只有在多处理器的windows平台才有效） |
| -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]  用本地线程对象分配内存（从1.4.0引入，UseTLE优先于该选项，- client: 无效） |
| -XX:+UseSplitVerifier | Use the new type checker with StackMapTable attributes. (Introduced in 5.0.)[5.0: false]  用新类型检查器里用StackMapTable属性。（从5.0开始引入） |
| -XX:+UseThreadPriorities | Use native thread priorities.  优先用本地线程 |
| -XX:+UseVMInterruptibleIO | Thread interrupt before or with EINTR for I/O operations results in OS\_INTRPT. (Introduced in 6. Relevant to Solaris only.)  在用I/O操作导致的OS\_INTRPT或者使用线程终断。  （在6.0引入，只与solaris有关） |

**Performance Options（性能参数）**

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.)  打开性能编译器优化（在5.0引入升级到6.0） |
| -XX:CompileThreshold=10000 | Number of method invocations/branches before compiling [-client: 1,500]  编译前方法调用/分支的数目【-client默认是1500】 |
| -XX:LargePageSizeInBytes=4m | Sets the large page size used for the Java heap. (Introduced in 1.4.0 update 1.) [amd64: 2m.]  设置用于Java堆的最大页的尺寸 |
| -XX:MaxHeapFreeRatio=70 | Maximum percentage of heap free after GC to avoid shrinking.  为了避免重叠，GC后最大空闲的堆的最大百分比。 |
| -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.]  新生代的最大尺寸。从1.4开始，MaxNewSize由NewRatio的函数计算而来。 |
| -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.]  永久生代的尺寸。 |
| -XX:MinHeapFreeRatio=40 | Minimum percentage of heap free after GC to avoid expansion.  为了避免扩张，GC后最小空闲堆的百分比。 |
| -XX:NewRatio=2 | Ratio of new/old generation sizes. [Sparc -client: 8; x86 -server: 8; x86 -client: 12.]-client: 4 (1.3) 8 (1.3.1+), x86: 12]  新生代/老生代的比分比。 |
| -XX:NewSize=2.125m | 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]  默认的新生代大小。 |
| -XX:ReservedCodeCacheSize=32m | Reserved code cache size (in bytes) - maximum code cache size. [Solaris 64-bit, amd64, and -server x86: 48m; in 1.5.0\_06 and earlier, Solaris 64-bit and and64: 1024m.]  保留的代码缓存大小-最大的代码缓存大小 |
| -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]  销毁/保留的空间比例 |
| -XX:TargetSurvivorRatio=50 | Desired percentage of survivor space used after scavenge.  清除后期望的保留空间比率 |
| -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.]  线程栈大小 |
| -XX:+UseBiasedLocking | Enable biased locking. For more details, see this [tuning example](http://dasheng.iteye.com/technetwork/java/tuning-139912.html#section4.2.5). (Introduced in 5.0 update 6.) [5.0: false]  使用偏锁 |
| -XX:+UseFastAccessorMethods | Use optimized versions of Get<Primitive>Field.  使用优化版本的Get<Primitive>方法。 |
| -XX:-UseISM | Use Intimate Shared Memory. [Not accepted for non-Solaris platforms.] For details, see [Intimate Shared Memory](http://dasheng.iteye.com/technetwork/java/ism-139376.html).  使用亲密共享空间【对非solaris平台无效】 |
| -XX:+UseLargePages | Use large page memory. (Introduced in 5.0 update 5.) For details, see [Java Support for Large Memory Pages](http://dasheng.iteye.com/technetwork/java/javase/tech/largememory-jsp-137182.html).  使用large page memory |
| -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]  使用Multiple Page Size Support |
| -XX:+UseStringCache | Enables caching of commonly allocated strings.  使用常用字符串缓存 |
| -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.  用预取指令分配一个对象，缓存代码的行数。缺省是1，数组是3. |
| -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.  预取指令的产生代码风格：  0-没有预取指令，  1-每一次分配内存就执行预取指令，  2-当执行预取代码指令时，用TLAB分配水印指针指向门。 |
| -XX:+UseCompressedStrings | Use a byte[] for Strings which can be represented as pure ASCII. (Introduced in Java 6 Update 21 Performance Release)  用byte[] 代替字符串（从Java6开始引入） |
| -XX:+OptimizeStringConcat | Optimize String concatenation operations where possible. (Introduced in Java 6 Update 20)  可能的话优化字符串连接操作（从Java6引入） |

**Debugging Options（dubug参数）**

Option and Default Value Description

|  |
| --- |
|  |
| -XX:-CITime | Prints time spent in JIT Compiler. (Introduced in 1.4.0.)  打印JIT编译器花费的时间（从1.4.0引入） |
| -XX:ErrorFile=./hs\_err\_pid<pid>.log | If an error occurs, save the error data to this file. (Introduced in 6.)  错误发生时，把错误保存到一个指定文件。 |
| -XX:-ExtendedDTraceProbes | Enable performance-impacting [dtrace](http://java.sun.com/javase/6/docs/technotes/guides/vm/dtrace.html) probes. (Introduced in 6. Relevant to Solaris only.)  启用performance-impacting [dtrace](http://java.sun.com/javase/6/docs/technotes/guides/vm/dtrace.html)跟踪（从6.0引入，只在solaris有效） |
| -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.)  堆dump文件的存放目录。（从1.4.2引入） |
| -XX:-HeapDumpOnOutOfMemoryError | Dump heap to file when java.lang.OutOfMemoryError is thrown. *Manageable*. (Introduced in 1.4.2 update 12, 5.0 update 7.)  当抛出异常java.lang.OutOfMemoryError 时，Dump heap 到文件。（从1.4.2引入） |
| -XX:OnError="<cmd args>;<cmd args>" | Run user-defined commands on fatal error. (Introduced in 1.4.2 update 9.)  在致命错误发生时运行用户定义命令（从1.4.2引入） |
| -XX:OnOutOfMemoryError="<cmd args>;  <cmd args>" | Run user-defined commands when an OutOfMemoryError is first thrown. (Introduced in 1.4.2 update 12, 6)  运行用户定义命令当一个OutOfMemoryError 异常抛出时。  （从1.4.2引入） |
| -XX:-PrintClassHistogram | Print a histogram of class instances on Ctrl-Break. *Manageable*. (Introduced in 1.4.2.) The [jmap -histo](http://java.sun.com/javase/6/docs/technotes/tools/share/jmap.html) command provides equivalent functionality.  Ctrl-Break打印类实例的直方图（从1.4.2引入） |
| -XX:-PrintConcurrentLocks | Print java.util.concurrent locks in Ctrl-Break thread dump. *Manageable*. (Introduced in 6.) The [jstack -l](http://java.sun.com/javase/6/docs/technotes/tools/share/jstack.html) command provides equivalent functionality.  打印java.util.concurrent 锁 |
| -XX:-PrintCommandLineFlags | Print flags that appeared on the command line. (Introduced in 5.0.)  打印在命令行中的标志（从5.0引入） |
| -XX:-PrintCompilation | Print message when a method is compiled.  当方法被编译时打印消息 |
| -XX:-PrintGC | Print messages at garbage collection. *Manageable*.  垃圾收集时打印消息 |
| -XX:-PrintGCDetails | Print more details at garbage collection. *Manageable*. (Introduced in 1.4.0.)  垃圾收集时打印具体信息 |
| -XX:-PrintGCTimeStamps | Print timestamps at garbage collection. *Manageable* (Introduced in 1.4.0.)  垃圾收集时打印时间戳（从1.40引入） |
| -XX:-PrintTenuringDistribution | Print tenuring age information.  打印任期年纪信息 |
| -XX:-TraceClassLoading | Trace loading of classes.  跟踪正在加载的类 |
| -XX:-TraceClassLoadingPreorder | Trace all classes loaded in order referenced (not loaded). (Introduced in 1.4.2.)  按顺序跟踪所有已经加载的类（从1.4.2引入） |
| -XX:-TraceClassResolution | Trace constant pool resolutions. (Introduced in 1.4.2.)  跟踪常量池分辨率（从1.4.2引入） |
| -XX:-TraceClassUnloading | Trace unloading of classes.  跟踪正在上传的类 |
| -XX:-TraceLoaderConstraints | Trace recording of loader constraints. (Introduced in 6.)  跟踪加载器的限制记录（从6.0引入） |
| -XX:+PerfSaveDataToFile | Saves jvmstat binary data on exit.  退出时保存jvmstat二进值数据 |
| -XX:ParallelGCThreads= | 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.  设置在青年和老年并行垃圾收集程序的收集线程的数目，缺省值根据JVM运行的平台变化而变化。 |
| -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.  使用compressed pointers |
| -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.  当JVM初始化时预先对Java堆进行预先摸底 |
| -XX:AllocatePrefetchDistance= | 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.  为对象分配设置预取距离。 |
| -XX:InlineSmallCode= | 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.  当生成本地代码的大小小于给定的值时，内联这个预编译方法。 |
| -XX:MaxInlineSize=35 | Maximum bytecode size of a method to be inlined.  内联一个方法的最大字节码大小 |
| -XX:FreqInlineSize= | Maximum bytecode size of a frequently executed method to be inlined. The default value varies with the platform on which the JVM is running.  经常执行方法内联的最大字节大小 |
| -XX:LoopUnrollLimit= | 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.  代表节点数目小于给定值时打开循环体 |
| -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.  为自适应GC设置保有年龄阀值。就是一个对象被升级到年老，中年的时代的次数。 |
| -XX:MaxTenuringThreshold= | 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.  设置最大年龄保有阀值。当前最大值是15. |