HotSpot JVM options cheatsheet

Java 8

All concrete numbers in JVM options in this card are for illustrational purposes only!

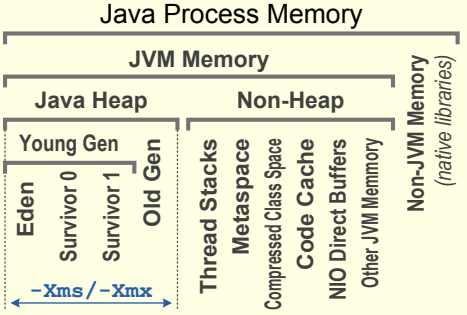
Alexey Ragozin – http://blog.ragozin.info

Available combinations of garbage collection algorithms in HotSpot JVM

Young collector	Old collector	JVM Flags
Serial (DefNew)	Serial Mark Sweep Compact	`-XX:+UseSerialGC`
Parallel scavenge (PSYoungGen)	Serial Mark Sweep Compact (PSOldGen)	`-XX:+UseParallelGC`
Parallel scavenge (PSYoungGen)	Parallel Mark Sweep Compact (ParOldGen)	`-XX:+UseParallelOldGC`
Parallel (ParNew)	Serial Mark Sweep Compact	`-XX:+UseParNewGC`
Serial (DefNew)	Concurrent Mark Sweep	`-XX:-UseParNewGC`[1] `-XX:+UseConcMarkSweepGC`
Parallel (ParNew)	Concurrent Mark Sweep	`-XX:+UseParNewGC` `-XX:+UseConcMarkSweepGC`
Garbage First (G1)		`-XX:+UseG1GC`

1 - Notice minus before UseParNewGC, which is explicitly disables parallel mode

Java Process Memory



GC log detail options

- `-verbose:gc` or `-XX:+PrintGC` — Print basic GC info 👍
- `-XX:+PrintGCDetails` — Print more details GC info 👍
- `-XX:+PrintGCTimeStamps` — Print timestamps for each GC event (seconds count from start of JVM)
- `-XX:+PrintGCDateStamps` — Print date stamps at garbage collection events: `2011-09-08T14:20:29.557+0400: [GC...` 👍
- `-XX:+PrintReferenceGC` — Print times for special (weak, JNI, etc) reference processing during STW pause 👍
- `-XX:+PrintJNIGCStalls` — Reports if GC is waiting for native code to unpin object in memory
- `-XX:+PrintGCCause` — Add cause of GC in log 👍
- `-XX:+PrintAdaptiveSizePolicy` — Print young space sizing decisions
- `-XX:+PrintPromotionFailure` — Print additional information for promotion failure 👍
- `-XX:+PrintGCApplicationStoppedTime` — Print summary after each JVM safepoint (including non-GC)
- `-XX:+PrintGCApplicationConcurrentTime` — Print time for each concurrent phase of GC

👍 - Highly recommended option

GC Log rotation

- `-Xloggc:<file>` — Redirects GC output to a file instead of console
- `-XX:+UseGCLogFileRotation` — Enable GC log rotation
- `-XX:GCLogFileSize=512m` — Size threshold for GC log file
- `-XX:NumberOfGCLogFiles=5` — Number GC log files

More logging options

- `-XX:+PrintTenuringDistribution` — Print detailed demography of young space after each collection
- `-XX:+PrintTLAB` — Print TLAB allocation statistics
- `-XX:+PrintPLAB` — Print survivor PLAB details
- `-XX:+PrintOldPLAB` — Print old space PLAB details
- `-XX:+PrintGCTaskTimeStamps` — Print timestamps for individual GC worker thread tasks (very verbose)
- `-XX:+PrintHeapAtGC` — Print heap details on GC
- `-XX:+PrintHeapAtSIGBREAK` — Print heap details on signal
- `-XX:+PrintClassHistogramAfterFullGC` — Prints class histogram after full GC
- `-XX:+PrintClassHistogramBeforeFullGC` — Prints class histogram before full GC

Memory sizing options

- `-Xms256m` or `-XX:InitialHeapSize=256m` — Initial size of JVM heap (young + old) 👍
- `-Xmx2g` or `-XX:MaxHeapSize=2g` — Max size of JVM heap (young + old) 👍
- `-XX:NewSize=64m` `-XX:MaxNewSize=64m` — Absolute (initial and max) size of young space (Eden + 2 Survivours) 👍
- `-XX:NewRatio=3` — Alternative way to specify size of young space. Sets ratio of young vs old space (e.g. `-XX:NewRatio=2` means that young space will be 2 time smaller than old space, i.e. 1/3 of heap size).
- `-XX:SurvivorRatio=15` — Sets size of single survivor space relative to Eden space size (e.g. `-XX:NewSize=64m -XX:SurvivorRatio=6` means that each Survivor space will be 8m and Eden will be 48m).
- `-XX:MetaspaceSize=512m` `-XX:MaxMetaspaceSize=1g` — Initial and max size of JVM's metaspace space
- `-Xss256k` (size in bytes) or `-XX:ThreadStackSize=256` (size in Kbytes) — Thread stack size
- `-XX:CompressedClassSpaceSize=1g` — Memory reserved for compressed class space (64bit only)
- `-XX:InitialCodeCacheSize=256m` `-XX:ReservedCodeCacheSize=512m` — Initial size and max size of code cache area
- `-XX:MaxDirectMemorySize=2g` — Maximum amount of memory available for NIO off-heap byte buffers

👍 - Highly recommended option

Young space tenuring

- `-XX:InitialTenuringThreshold=8` — Initial value for tenuring threshold (number of collections before object will be promoted to old space)
- `-XX:MaxTenuringThreshold=15` — Max value for tenuring threshold
- `-XX:PretenureSizeThreshold=2m` — Max object size allowed to be allocated in young space (large objects will be allocated directly in old space). Thread local allocation bypasses this check, so if TLAB is large enough object exciding size threshold still may be allocated in young space.
- `-XX:+AlwaysTenure` — Promote all objects surviving young collection immediately to tenured space (equivalent of `-XX:MaxTenuringThreshold=0`)
- `-XX:+NeverTenure` — Objects from young space will never get promoted to tenured space unless survivor space is not enough to keep them

Thread local allocation

- `-XX:+UseTLAB` — Use thread local allocation blocks in eden
- `-XX:+ResizeTLAB` — Let JVM resize TLABs per thread
- `-XX:TLABSize=1m` — Initial size of thread's TLAB
- `-XX:MinTLABSize=64k` — Min size of TLAB

Parallel processing

- `-XX:ConcGCThreads=2` — Number of parallel threads used for concurrent phase.
- `-XX:ParallelGCThreads=16` — Number of parallel threads used for stop-the-world phases.
- `-XX:+ParallelRefProcEnabled` — Enable parallel processing of references during GC pause

- `-XX:+DisableExplicitGC` — JVM will ignore application calls to `System.gc()`
- `-XX:+ExplicitGCInvokesConcurrent` — Let `System.gc()` trigger concurrent collection instead of full GC
- `-XX:+ExplicitGCInvokesConcurrentAndUnloadsClasses` — Same as above but also triggers permanent space collection.
- `-XX:SoftRefLRUPolicyMSPerMB=1000` — Factor for calculating soft reference TTL based on free heap size
- `-XX:OnOutOfMemoryError=...` — Command to be executed in case of out of memory. E.g. "kill -9 %p" on Unix or "taskkill /F /PID %p" on Windows.

Concurrent Mark Sweep (CMS)

CMS initiating options

- `-XX:+UseCMSInitiatingOccupancyOnly` — Only use predefined occupancy as only criterion for starting a CMS collection (disable adaptive behaviour) ⏱
- `-XX:CMSInitiatingOccupancyFraction=70` — Percentage CMS generation occupancy to start a CMS cycle. A negative value means that `CMSTriggerRatio` is used. ⏱
- `-XX:CMSBootstrapOccupancy=50` — Percentage CMS generation occupancy at which to initiate CMS collection for bootstrapping collection stats.
- `-XX:CMSTriggerRatio=70` — Percentage of MinHeapFreeRatio in CMS generation that is allocated before a CMS collection cycle commences.
- `-XX:CMSTriggerInterval=60000` — Periodically triggers CMS collection. Useful for deterministic object finalization. ⏱

CMS Stop-the-World pauses tuning

- `-XX:CMSWaitDuration=30000` — Once CMS collection is triggered, it will wait for next young collection to perform initial mark right after. This parameter specifies how long CMS can wait for young collection ⏱
- `-XX:+CMSScavengeBeforeRemark` — Force young collection before remark phase ⏱
- `-XX:+CMSScheduleRemarkEdenSizeThreshold` — If Eden used is below this value, don't try to schedule remark
- `-XX:CMSScheduleRemarkEdenPenetration=20` — Eden occupancy % at which to try and schedule remark pause
- `-XX:CMSScheduleRemarkSamplingRatio=4` — Start sampling Eden top at least before young generation occupancy reaches 1/ of the size at which we plan to schedule remark

CMS Concurrency options

- `-XX:+CMSParallelInitialMarkEnabled` — Whether parallel initial mark is enabled (enabled by default)
- `-XX:+CMSParallelRemarkEnabled` — Whether parallel remark is enabled (enabled by default)
- `-XX:+CMSParallelSurvivorRemarkEnabled` — Whether parallel remark of survivor space enabled, effective only with option above (enabled by default)
- `-XX:+CMSConcurrentMTEnabled` — Use multiple threads for concurrent phases.

Garbage First (G1)

- `-XX:G1HeapRegionSize=32m` — Size of heap region
- `-XX:G1ReservePercent=10` — Percentage of heap to keep free. Reserved memory is used as last resort to avoid promotion failure.
- `-XX:InitiatingHeapOccupancyPercent=45` — Percentage of (entire) heap occupancy to trigger concurrent GC
- `-XX:G1MixedGCCountTarget=8` — Target number of mixed collections after a marking cycle
- `-XX:G1HeapWastePercent=10` — If garbage level is below threshold, G1 will not attempt to reclaim memory further
- `-XX:G1ConfidencePercent=50` — Confidence level for MMU/pause prediction
- `-XX:MaxGCPauseMillis=500` — Target GC pause duration. G1 is not deterministic, so no guaranties for GC pause to satisfy this limit.

CMS Diagnostic options

- `-XX:PrintCMSStatistics=1` — Print additional CMS statistics. Very verbose if n=2.
- `-XX:+PrintCMSInitiationStatistics` — Print CMS initiation details
- `-XX:+CMSDumpAtPromotionFailure` — Dump useful information about the state of the CMS old generation upon a promotion failure
- `-XX:+CMSPrintChunksInDump` — (with optin above) Add more detailed information about the free chunks
- `-XX:+CMSPrintObjectsInDump` — (with optin above) Add more detailed information about the allocated objects

Misc CMS options

- `-XX:+CMSClassUnloadingEnabled` — If not enabled, CMS will not clean permanent space. You may need to enable it for containers such as JEE or OSGi.
- `-XX:+CMSIncrementalMode` — Enable incremental CMS mode. Incremental mode was meant for severs with small number of CPU, but may be used on multicore servers to benefit from more conservative initiation strategy.
- `-XX:+CMSOldPLABMin=16` `-XX:+CMSOldPLABMax=1024` — Min and max size of CMS gen PLAB caches per worker per block size

⏱ - Options for "deterministic" CMS, they disable some heuristics and require careful validation