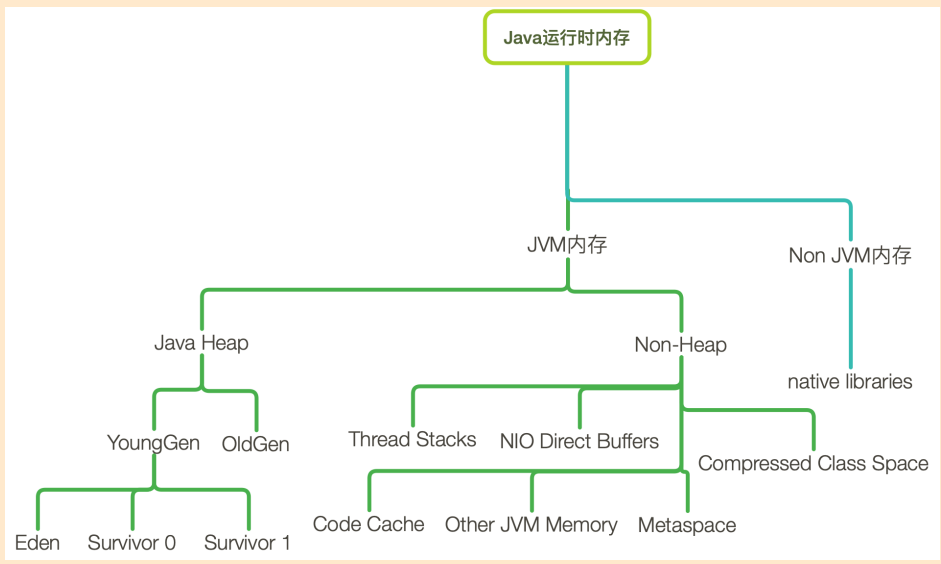
# Java 8

HotSpot JVM 调优秘籍

程序那些事 www.flydean.com




| HotSpot JVM可用GC类型 | | |
|---|---|---|
| 年轻代回收算法 | 老年代回收算法 | GC启用参数 |
| Serial(DefNew)) | Serial Mark Sweep Compact(PSOldGen) | -XX:+UseSerialGC |
| Parallel scavenge(PSYoungGen) | Serial Mark Sweep Compact(PSOldGen) | -XX:+UseParallelGC(默认) |
| Parallel scavenge(PSYoungGen)) | Parallel Mark Sweep Compact(ParOldGen) | -XX:+UseParallelOldGC(随-XX:+UseParallelGC开启) |
| Parallel(ParNew)) | Serial Mark Sweep Compactt(PSOldGen) | -XX:+UseParNewGC(随-XX:+UseConcMarkSweepGC开启) |
| Serial(DefNew)) | Concurrent Mark Sweep | -XX:-UseParNewGC -XX:+UseConcMarkSweepGC |
| Parallel(ParNew)) | Concurrent Mark Sweep | -XX:+UseParNewGC -XX:+UseConcMarkSweepGC |
| G1(Garbage First) | | -XX:+UseG1GC |

| GC日志详情 | |
|---|---|
| VM参数 | 说明 |
| -verbose:gc -XX:+PrintGCPrint | 打印基本的GC信息 |
| -XX:+PrintGCDetails | 打印详细的GC信息 |
| -XX:+PrintGCTimeStamps | 打印GC事件的timestamps |
| -XX:+PrintGCDateStamps | 以date stamps的形式打印GC事件 |
| -XX:+PrintReferenceGC | STW阶段打印reference processing |
| -XX:+PrintGCCause | 添加GC的原因 |
| -XX:+PrintAdaptiveSizePolicy | 输出自适应的分代大小 |
| -XX:+PrintPromotionFailure | 输出promotion失败的信息 |
| -XX:+PrintGCApplicationStoppedTime | 显示应用程序在安全点停止的时间 |
| -XX:+PrintGCApplicationConcurrentTime | 是应用程序在不停止的情况下工作的时间,即两个连续安全点之间的时间 |

| GC日志输出 | |
|---|---|
| VM参数 | 说明 |
| -Xloggc:filename | 将GC日志输出到文件 |
| -XX:ErrorFile=filename | GC错误日志重定向 |
| -XX:LogFile=path | JVM日志重定向 |
| -XX:+UseGCLogFileRotation | 开启滚动的GC日志 |
| -XX:GCLogFileSize=512m | GC日志文件的最大值 |
| -XX:NumberOfGCLogFiles=5 | GC日志文件的个数 |
| 其他GC日志信息 | |
| -XX:+PrintTLAB | 输出TLAB信息 |
| -XX:+PrintPLAB | 输出PLAB信息 |
| -XX:+PrintOldPLAB | 输出old space的PLAB信息 |
| -XX:+PrintGCTaskTimeStamps | 输出GC work thread task的timestamps |
| -XX:+PrintHeapAtGC | GC的heap信息 |
| -XX:+PrintHeapAtSIGBREAK | 接收到signal的时候的heap信息 |
| -XX:+PrintClassHistogramAfterFullGC | full GC后的class直方图信息 |
| -XX:+PrintClassHistogramBeforeFullGC | full GC之前的class直方图信息 |

| 内存大小调整 | |
|---|---|
| VM参数 | 说明 |
| -Xmnsize | young gen的初始化和最大值 |
| -XX:NewSize | young gen的初始化大小 |
| -XX:MaxNewSize | young gen的最大值 |
| -Xmssize -XX:InitialHeapSize=size | heap的初始值 |
| -Xmxsize -XX:MaxHeapSize | heap的最大值 |
| -Xsssize -XX:ThreadStackSize | Thread stack size |
| -XX:MaxDirectMemorySize=size | 设置NIO的最大direct-buffer size |
| -XX:ConcGCThreads=threads | concurrent GC的线程个数 |
| -XX:+DisableExplicitGC | 禁止显式调用System.gc() |
| -XX:InitialSurvivorRatio=ratio | survivor space相对Eden的比例 |
| -XX:MaxMetaspaceSize=size | 元数据区域的最大大小 |
| -XX:MetaspaceSize=size | 首次触发GC的class元数据区域大小 |
| -XX:NewRatio=ratio | young和old区域的大小比率 |
| -XX:SurvivorRatio=ratio | eden和survivor大小的比率 |
| -XX:+UseAdaptiveSizePolicy | 使用自适应的分代大小策略 |
| -XX:CompressedClassSpaceSize=1g | compressed class space大小 |
| -XX:InitialCodeCacheSize=256m | codeCache的初始化大小 |
| -XX:ReservedCodeCacheSize=512m | codeCache的最大大小 |
| -XX:MaxDirectMemorySize=2g | NIO direct buffer的最大值 |

| Thread配置 | |
|---|---|
| VM参数 | 说明 |
| -XX:TLABSize=size | TLAB的初始大小 |
| -XX:+UseTLAB | 开启TLAB |
| -XX:+ResizeTLAB | 允许JVM对TLAB进行调整 |
| -XX:MinTLABSize=64k | 最小TLAB大小 |

| G1调优参数 | |
|---|---|
| VM参数 | 说明 |
| -XX:G1HeapRegionSize=32m | heap区域的大小 |
| -XX:G1ReservePercent=10 | 设置预留空闲内存百分比，以降低内存溢出的风险 |
| -XX:G1MixedGCCountTarget=8 | 一个混合收集周期中包含多少次混合收集 |
| -XX:G1HeapWastePercent=10 | 设置浪费的堆内存百分比，当可回收百分比小于浪费百分比时，JVM就不会启动混合垃圾收 |
| -XX:MaxGCPauseMillis=500 | GC的最大暂停时间 |

| VM通用参数 | |
|---|---|
| VM参数 | 说明 |
| -XX:ObjectAlignmentInBytes=alignment | Java对象的对其字节大小 |
| -XX:-UseBiasedLocking | 禁止使用偏向锁 |
| -XX:-UseCompressedOops | 停止使用对象指针压缩 |

| 通用GC参数 | |
|---|---|
| VM参数 | 说明 |
| -Xnoclassgc | 禁用classes的GC |
| -XX:ActiveProcessorCount=x | 重写VM使用的CPU核数 |
| -XX:+AggressiveHeap | 开启java heap优化 |
| -XX:+AlwaysPreTouch | 在main方法执行之前将所有的page都加载到heap中 |
| -XX:InitiatingHeapOccupancyPercent=percent | 触发GC的heap使用比例 |
| -XX:MaxGCPauseMillis=time | 最大的GC暂停时间 |
| -XX:MaxHeapFreeRatio=percent | GC之后最大的Heap释放比例 |
| -XX:MinHeapFreeRatio=percent | GC之后最小的Heap释放比例 |
| -XX:TargetSurvivorRatio=percent | young GC之后Survivor的目标比例 |
| -XX:+ScavengeBeforeFullGC | 在fullGC之前运行youngGC |
| -XX:StringDeduplicationAgeThreshold=threshold | 字符串去重的最小Age数 |
| -XX:+UseStringDeduplication | 开启字符串去重 |

| GC并发线程控制 | |
|---|---|
| VM参数 | 说明 |
| -XX:ParallelGCThreads=threads | 设置STW的垃圾收集线程数 |
| -XX:+ParallelRefProcEnabled | 开启并发reference processing |
| -XX:+UseGCOverheadLimit | OutOfMemoryError之前JVM在GC上使用的时间比例 |
| -XX:ConcGCThreads = n | 设置并行标记线程的数量 |
| Young space tenuring | |
| -XX:InitialTenuringThreshold=8 | 设置保有年龄阀值,就是一个对象多少age之后会被升级到old space |
| -XX:MaxTenuringThreshold=15 | 最大的保有年龄阈值 |
| -XX:PretenureSizeThreshold=2m | 超出该阈值，对象将会被直接分配到old space |
| -XX:+AlwaysTenure | 将young space中的survivor对象直接提升到old space |
| -XX:+NeverTenure | 除非survivor space不能容纳该对象了，否则不会提升到old space |

| CMS初始化参数 | |
|---|---|
| VM参数 | 说明 |
| -XX:+UseCMSInitiatingOccupancyOnly | 只是用设定的回收阈值,如果不指定,JVM仅在第一次使用设定值,后续则自动调整. |
| -XX:CMSInitiatingOccupancyFraction=70 | 设定CMS在对内存占用率达到70%的时候开始GC |
| -XX:CMSBootstrapOccupancy=50 | 用于触发CMS收集bootstrap统计信息的比率 |
| -XX:CMSTriggerRatio=70 | 触发CMS的最小的Heap free比率 |
| -XX:CMSTriggerInterval=60000 | 定期触发CMS收集 |
| CMS Stop the world调优 | |
| -XX:CMSWaitDuration=30000 | CMS被触发后，等待下一次年轻代执行标记的时间 |
| -XX:+CMSScavengeBeforeRemark | 在CMS GC前启动一次ygc，目的在于减少old gen对ygc gen的引用 |
| -XX:+CMSScheduleRemarkEdenSizeThreshold | Eden使用比率小于该值，则不会触发remark |
| -XX:CMSScheduleRemarkEdenPenetration=20 | 触发remark的Eden使用比率 |

| CMS并发参数 | |
|---|---|
| VM参数 | 说明 |
| -XX:+CMSParallelInitialMarkEnabled | 是否开启parallel initial mark |
| -XX:+CMSParallelRemarkEnabled | 是否开启parallel rmark |
| -XX:+CMSParallelSurvivorRemarkEnabled | 是否开启Survivor的parallel remark |
| -XX:+CMSConcurrentMTEnabled | 并发阶段是否使用多线程 |
| CMS调试参数 | |
| -XX:PrintCMSStatistics=1 | 输出CMS的统计信息 |
| -XX:+PrintCMSInitiationStatistics | 输出CMS初始化信息 |
| -XX:+CMSDumpAtPromotionFailure | promotion失败后，dump old generation的状态 |
| -XX:+CMSPrintChunksInDump | 打印free chunks的信息 |
| -XX:+CMSPrintObjectsInDump | 打印allocated objects的信息 |