简述
内存相关
收集器相关
辅助信息

简述

通过虚拟机参数可以对虚拟机进行设置（如设置堆大小，栈深度等），可以对虚拟机进行跟踪（跟踪垃圾回收信息，类加载情况）。
通过对虚拟机参数的设置来解决诊断问题与性能优化

内存相关

选项	参数详解
-Xms	初始堆大小
-Xmx	最大堆大小
-Xmn	年轻代大小(1.4or lator)整个JVM内存大小=年轻代大小 + 年老代大小 + 持久代大小。持久代一般固定大 年轻代后，将会减小年老代大小。此值对系统性能影响较大，Sun官方推荐配置为整个堆的3/8
-XX:newSize	表示新生代初始内存的大小，应该小于 -Xms的值
-XX:NewRatio	设置年轻代和年老代的比值。如:为3，表示年轻代与年老代比值为1：3，年轻代占整个年轻代年老代和
-XX:MaxNewSize	年轻代最大值(for 1.3/1.4)
-XX:PermSize	设置持久代(perm gen)初始值
-XX:MaxPermSize	设置持久代最大值
-Xss	每个线程的堆栈大小
-XX:ThreadStackSize	--
-XX:SurvivorRatio	Eden区与Survivor区的大小比值， 设置为8,则两个Survivor区与一个Eden区的比值为2:8,一个Survivor
-XX:LargePageSizeInBytes	内存页的大小不可设置过大， 会影响Perm的大小，基本没用过
-XX:+UseFastAccessorMethods	原始类型的快速优化 1.7以后不建议使用，1.6之前默认打开的
-XX:+UseFastEmptyMethods	优化空方法，1.7以后不建议使用，1.6之前默认打开的
-XX:+DisableExplicitGC	关闭System.gc()
-XX:MaxTenuringThreshold	设置垃圾最大年龄。如果设置为0的话，则年轻代对象不经过Survivor区，直接进入年老代。对于年老 提高效率。如果将此值设置为一个较大值，则年轻代对象会在Survivor区进行多次复制，这样可以增加 间，增加在年轻代即被回收的概率
-XX:+AggressiveOpts	加快编译
-XX:+UseBiasedLocking	锁机制的性能改善， 有偏见的锁是使得锁更偏爱上次使用到它线程。在非竞争锁的场景下，即只有一 以实现近乎无锁的开销。
-Xnoclassgc	禁用类垃圾回收
-XX:SoftRefLRUPolicyMSPerMB	每兆堆空闲空间中SoftReference的存活时间
-XX:PretenureSizeThreshold	对象超过多大是直接在旧生代分配，单位字节 新生代采用Parallel Scavenge GC时无效另一种直接在旧 数组对象,且数组中无外部引用对象.
-XX:+CollectGen0First	FullGC时是否先YGC

收集器相关

选项	参数详解
-XX:+UseSerialGC	设置串行收集器
-XX:+UseParallelGC	选择垃圾收集器为并行收集器。此配置仅对年轻代有效。可以同时并行多个垃圾收集线程，1 止。
-XX:+UseParNewGC	设置年轻代收集器ParNew
-XX:ParallelGCThreads	Parallel并行收集器的线程数
-XX:+UseParallelOldGC	设置老年代的并行收集器是ParallelOld
-XX:+UseG1GC	使用G1收集器
-XX:MaxGCPauseMillis	每次年轻代垃圾回收的最长时间(最大暂停时间)
-XX:+UseAdaptiveSizePolicy	设置此选项后,并行收集器会自动选择年轻代区大小和相应的Survivor区比例,以达到目标系统 者收集频率等,此值建议使用并行收集器时,一直打开.
-XX:GCTimeRatio	设置垃圾回收时间占程序运行时间的，百分比公式为1/(1+n)
-XX:+ScavengeBeforeFullGC	Full GC前调用YGC
-XX:+UseConcMarkSweepGC	使用CMS内存收集
-XX:+AggressiveHeap	试图是使用大量的物理内存长时间大内存使用的优化，能检查计算资源（内存， 处理器数量 大量的CPU / 内存， （在1.4.1在4CPU的机器上已经显示有提升）
-XX:CMSFullGCsBeforeCompaction	由于并发收集器不对内存空间进行压缩,整理,所以运行一段时间以后会产生"碎片",使得运行效 多少次GC以后对内存空间进行压缩,整理
-XX:+CMSParallelRemarkEnabled	降低CMS标记停顿
-XX+UseCMSCompactAtFullCollection	在FULL GC的时候， 对年老代的压缩，CMS是不会移动内存的， 因此， 这个非常容易产生碎 用， 因此， 内存的压缩这个时候就会被启用。 增加这个参数是个好习惯。可能会影响性能,
-XX:+UseCMSInitiatingOccupancyOnly	使用手动定义初始化定义开始CMS收集，禁止hostspot自行触发CMS GC
-XX:CMSInitiatingOccupancyFraction=70	使用cms作为垃圾回收使用70%后开始CMS收集
-XX:CMSInitiatingPermOccupancyFraction	设置Perm Gen使用到达多少比率时触发
-XX:+CMSIncrementalMode	设置为增量模式
-XX:CMSTriggerRatio	CMSInitiatingOccupancyFraction = (100 - MinHeapFreeRatio) + (CMSTriggerRatio * MinH 罚cms收集的比例
-XX:MinHeapFreeRatio	java堆中空闲量占的最小比例
-XX:+CMSClassUnloadingEnabled	如果你启用了CMSClassUnloadingEnabled ，垃圾回收会清理持久代，移除不再使用的class UseConcMarkSweepGC 也启用的情况下才有用。参数如下：

辅助信息

选项	参数详解
-XX:+PrintGC	输出形式:[GC 118250K->113543K(130112K), 0.0094143 secs]Full GC 121376K->10414K(1301
-XX:+PrintGCDetails	--
-XX:+PrintGCTimeStamps	--
-XX:+PrintGC:PrintGCTimeStamps	--
-XX:+PrintGCApplicationStoppedTime	打印垃圾回收期间程序暂停的时间.可与上面混合使用
-XX:+PrintGCApplicationConcurrentTime	打印每次垃圾回收前,程序未中断的执行时间.可与上面混合使用
-XX:+PrintHeapAtGC	打印GC前后的详细堆栈信息
-Xloggc:filename	把相关日志信息记录到文件以便分析.与上面几个配合使用
-XX:+PrintClassHistogram	遇到Ctrl-Break后打印类实例的柱状信息，与jmap -histo功能相同
-XX:+PrintTenuringDistribution	查看每次minor GC后新的存活周期的阈值
-XX:PrintHeapAtGC	打印GC前后的详细堆栈信息