export JAVA\_OPTIONS="-server

**-Xmx2048M** （最大堆大小）

**-Xms2048M** （初始堆大小）

**-Xmn768M** （年轻代大小）

**-XX:MaxMetaspaceSize=256M** （metaSpace/永久代大小）

**-XX:MetaspaceSize=256M** （metaSpace/永久代大小）

**-XX:+UseConcMarkSweepGC** （使用CMS垃圾回收器）

**-XX:+UseCMSInitiatingOccupancyOnly** （仅使用设定的阈值，不加此参数，阈值会自适应调整）

**-XX:CMSInitiatingOccupancyFraction=70** （设定CMS在老年代占用率达到指定阈值时开始full GC）

**-XX:+ExplicitGCInvokesConcurrentAndUnloadsClasses** （用来改变System.gc()的默认行为，只能配合CMS使用。System.gc()还是会触发GC，只不过不是触发一个 完全stop-the-world的full GC，而是一次并发GC周期。）

**-XX:+CMSClassUnloadingEnabled** （表示在使用CMS垃圾回收机制的时候是否启用类卸载功能，垃圾回收会清理持久代，移除不再使用的classes。这个参数只有在 UseConcMarkSweepGC 也启用的情况下才有用）

**-XX:+ParallelRefProcEnabled** （如果应用有很多的Reference或finalizable objects，那么可以使用此参数来减少耗时）

**-XX:+CMSScavengeBeforeRemark** （在CMS重新标记阶段之前的清除（YGC）尝试。CMS并发标记阶段与用户线程并发进行，此阶段会产生已经被标记了的对象又发生变化的情况，若打开此开关，可在一定程度上降低CMS重新标记阶段对上述“又发生变化”对象的扫描时间，“清除尝试”也会消耗一些时间。开启此开关并不会保证在标记阶段前一定会进行清除操作）

**-XX:-HeapDumpOnOutOfMemoryError** （当JVM发生OOM时，自动生成DUMP文件）

**-XX:+UseFastAccessorMethods** （原始类型get,set方法的优化）

-Xss256k

-XX:SurvivorRatio=8

-XX:MaxTenuringThreshold=7

-XX:GCTimeRatio=19

-XX:+UseParNewGC

-XX:+DisableExplicitGC

-XX:+CMSParallelRemarkEnabled

-XX:SoftRefLRUPolicyMSPerMB=0

-Djava.awt.headless=true

-Djava.net.preferIPv4Stack=true

-Dapplication.codeset=UTF-8

-Dfile.encoding=UTF-8

-Djava.util.Arrays.useLegacyMergeSort=true "