export JAVA\_OPTIONS="-server

**-Xmx2048M** （最大堆大小）

**-Xms2048M** （初始堆大小）

**-Xmn768M** （年轻代大小）

**-Xss256k** （每个线程堆大小）

**-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:+DisableExplicitGC** （将System.gc()变为空调用，不会触发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:+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:+CMSParallelRemarkEnabled** （开启CMS收集器重新标记阶段的并行处理）

**-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:-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）

**-XX:SurvivorRatio=8** （年轻代Survivor区的比例）

**-XX:MaxTenuringThreshold=7** （年轻代晋升至老年代需要经历的GC次数）

**-XX:GCTimeRatio=19** （假设该值为n，那么系统将花费不超过1/(1+n)的时间用于垃圾收集，用于Parallel scavenge垃圾收集器）

**-XX:+UseParNewGC** （使用ParNew垃圾收集器）

**-XX:SoftRefLRUPolicyMSPerMB=0** （SoftReference对象的存活时间：空闲内存\*SoftRefLRUPolicyMSPerMB；此参数过小可能导致频繁发生fullGC）

**-Djava.awt.headless=true** （在系统缺少鼠标键盘等设备的情况下使用该模式）

**-Djava.net.preferIPv4Stack=true** （只支持ipv4）

**-Dapplication.codeset=UTF-8** （字符集设置）

**-Dfile.encoding=UTF-8** （文件字符集设置）

**-Djava.util.Arrays.useLegacyMergeSort=true** （解决排序问题，<https://www.jianshu.com/p/4a687ddf4f62>）"

######################

-XX:TraceClassLoading

-XX:TraceClassUnloading

追踪类加载和类卸载的情况，调试时使用