java -Xms1024m -Xmx2056m -XX:PermSize=512m -XX:MaxPermSize=1024m -XX:MaxNewSize=1024m -jar

=========================================================

java -server -Xmx4g -Xms4g -XX:+UseG1GC -XX:MaxGCPauseMillis=50 -XX:+PrintGCDetails -XX:+PrintGCDateStamps -XX:+HeapDumpOnOutOfMemoryError -XX:HeapDumpPath=D:\cruve-logs\ -jar

============================================================

10.你们线上应用的 JVM 参数有哪些。

-server

-Xms6000M

-Xmx6000M

-Xmn500M

-XX:PermSize=500M

-XX:MaxPermSize=500M

-XX:SurvivorRatio=65536

-XX:MaxTenuringThreshold=0

-Xnoclassgc

-XX:+DisableExplicitGC

-XX:+UseParNewGC

-XX:+UseConcMarkSweepGC

-XX:+UseCMSCompactAtFullCollection

-XX:CMSFullGCsBeforeCompaction=0

-XX:+CMSClassUnloadingEnabled

-XX:-CMSParallelRemarkEnabled

-XX:CMSInitiatingOccupancyFraction=90

-XX:SoftRefLRUPolicyMSPerMB=0

-XX:+PrintClassHistogram

-XX:+PrintGCDetails

-XX:+PrintGCTimeStamps

-XX:+PrintHeapAtGC

-Xloggc:log/gc.log

11.g1 和 cms 区别,吞吐量优先和响应优先的垃圾收集器选择。

Cms是以获取最短回收停顿时间为目标的收集器。基于标记-清除算法实现。比较占用cpu资源，切易造成碎片。

G1是面向服务端的垃圾收集器，是jdk9默认的收集器，基于标记-整理算法实现。可利用多核、多cpu，保留分代，实现可预测停顿，可控。

参考链接

请解释如下 jvm 参数的含义：

-server -Xms512m -Xmx512m -Xss1024K

-XX:PermSize=256m -XX:MaxPermSize=512m -XX:MaxTenuringThreshold=20

XX:CMSInitiatingOccupancyFraction=80 -XX:+UseCMSInitiatingOccupancyOnly。

Server模式启动

最小堆内存512m

最大512m

每个线程栈空间1m

永久代256m

最大永久代512m

最大转为老年代检查次数20

Cms回收开启时机：内存占用80%

命令JVM不基于运行时收集的数据来启动CMS垃圾收集周期

12.32位系统jvm堆内存如下哪一个设置是最大且有效的( B )

A. -Xmx1000m

B. -Xmx1500m

C. -Xmx8G

D. 无限

JVM最大内存: 首先JVM内存限制于实际的最大物理内存，假设物理内存无限大的话，JVM内存的最大值跟操作系统有很大的关系。简单的说就32位处理器虽然可控内存空间有4GB,但是具体的操作系统会给一个限制，这个限制一般是2GB-3GB（一般来说Windows系统下为1.5G-2G，Linux系统下为2G-3G），而64bit以上的处理器就不会有限制了

————————————————

版权声明：本文为CSDN博主「是华仔呀」的原创文章，遵循CC 4.0 BY-SA版权协议，转载请附上原文出处链接及本声明。

原文链接：https://blog.csdn.net/weixin\_43495390/article/details/86533482

**总结导致java.lang.OutOfMemoryError异常的常见原因有以下几种：**

内存中加载的数据量过于庞大，如一次从数据库取出过多数据；

集合类中有对对象的引用，使用完后未清空，使得JVM不能回收；

代码中存在死循环或循环产生过多重复的对象实体-----（本例中）；

使用的第三方软件中的BUG；

启动参数内存值设定的过小；

至此问题总算解决了，通常情况下，Java开发人员并不需要去关心JVM是如何运行的。即使不理解JVM的工作原理，也不会给开发人员带来过多困惑。Java程序员更容易忽视基础技术。

“蚓无爪牙之利，筋骨之强，上食埃土，下饮黄泉，用心一也。蟹六跪而二螯，非蛇蟮之穴无可寄托者，用心躁也”。对于技术人员来说，如果长期忽略自身技术的根基而去一昧地追求高层框架技术，这无疑是舍本求末的做法。

JVM的出现，为程序员屏蔽了操作系统与硬件的细节，使得程序员从诸如内存管理这样的繁琐任务中解放出来。但这不并等同于允许Java程序员放弃对基础的重视。

事实上，任何平台的程序员都应当了解平台的基本特性、实现机制以及接口，这是提高自身修养的必经之路。对于Java程序员来说，我们还是需要多多了解JVM。

了解JVM的基本实现机制，不仅对于解决实际应用中诸如GC等虚拟机问题时有直接帮助，还有利于我们更好地理解语言本身。