高薪必备JVM面试题解惑

主讲：Cat 老师

北京动力节点教育科技有限公司  
2021 • 北京

动力节点•版权所有•禁止传播

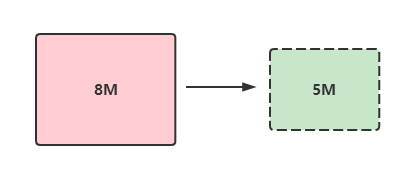
## 什么是内存泄漏？什么是内存溢出？

**内存溢出：**OutOfMemory

它是指程序在申请内存时，没有足够的内存空间供其使用，抛出OutOfMemory错误；

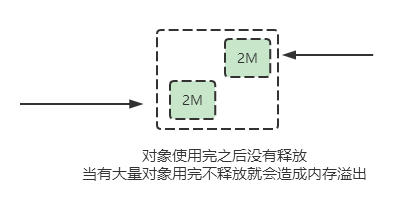
比如申请了一个8MB空间，但是当前内存可用空间只有5MB，那么就是内存溢出；

即：OutOfMemoryError，是指没有空闲内存，垃圾收集器回收后也不能提供更多的内存空间；



**内存泄露：**Memory Leak

它是指程序运行后，没有释放所占用的内存空间，一次内存泄漏可能不会有很大的影响，但长时间的内存泄漏，堆积到一定程度就会产生内存溢出；



（1）单例对象，生命周期和应用程序一样长，如果单例对象持有对外部对象的引用的话，那么这个外部对象是不能被回收的，则会产生内存泄露；

（2）一些资源未闭也会导致内存泄漏，比如数据库连接，网络连接socket和IO流的连接都必须在finally中close，否则不能被回收的；

## 你们线上环境的JVM都设置多大？

线上：4核8G机器；

JVM：栈、堆、元空间；

1、栈 1m，xss512k，一个线程是1m，一个线上项目Tomcat可能有300个线程，300m；

2、堆：大概把机器的一半内存给堆，4G（新生代、老年代）；

CMS：1/3 、2/3 G1: 6:4

3、元空间： 一般512M肯定够了；

此时JVM参数如下：

-Xms4096M -Xmx4096M -Xss1M -XX:MetaspaceSize=512M -XX:MaxMetaspaceSize=512M -XX:+UseG1GC

## 线上Java服务器内存飙升怎么回事？

jmap -histo pid

jmap -heap pid

jmap -dump:format=b,file=heap.hprof pid

## 线上Java项目CPU飙到100%怎么排查？

top

top -H -p pid

printf ‘%x’ tid 8ef

jstack pid

## JVM堆溢出后，其他线程是否可以继续工作？

## 线上Java项目OOM了，怎么回事？

-Xms256M

-Xmx256M

-XX:+PrintGCDetails

-XX:+PrintGCDateStamps

-Xloggc:d:/gc.log

-XX:+HeapDumpOnOutOfMemoryError

-XX:HeapDumpPath=d:/heap.hprof

## 某一天晚上，线上Java服务器宕机了，怎么回事？

## 亿级流量的订单系统JVM优化

1. 系统预估

每个用户访问20次/天，500万日活用户，流量 = 500万 \* 20 = 10000万 = 1亿；

购买率15%，每人1单，每天订单量 = 500万 \* 15% \* 1 = 75万订单/天；

二八原则，下单集中在一天4小时内，洪峰下单量 = 75万 / 4小时 = 18.75万单/小时

= 18.75万单/60分/60秒 = 52单/秒；

52k\*20\*10=52\*200=10400 = 10MB/秒，每秒52单基本上JVM没有压力；

1. 流量洪峰场景



普通4核8G服务器，一台机器抗300-400并发下单请求比较合理；

583000 / 300 = 1943台机器

300KB \* 20 \* 10 = 60MB的内存开销，一秒后60MB对象就成为垃圾；

1. 内存分配

4核8G的机器，JVM给4G，剩下几个G会留给操作系统；

堆3G（新生代1.5G，老年代1.5G）

栈1MB，JVM里大概会有300-500个线程，大概300-500MB;

元空间/永久代512MB；

“-Xms3072M -Xmx3072M -Xmn1536M -Xss1M

-XX:MetaspaceSize=512M -XX:MaxMetaspaceSize=512M”

-XX:+PrintGCDetails

-XX:+PrintGCDateStamps

-Xloggc:d:/gc.log

-XX:+HeapDumpOnOutOfMemoryError

-XX:HeapDumpPath=d:/heap.hprof

1. 内存占用动态推算

每秒处理300个订单，占据新生代60MB的内存空间，新生代总共有1.5G的内存空间；

1.5G \* 1024MB / 60MB = 25秒 新生代Eden占满，触发Minor GC；

一般情况下一次可以回收掉90%的新生代对象，存活对象 = 1.5G \* 1024MB \* 10% = 150MB；

如果“-XX:SurvivorRatio”参数默认值为8，那么：

新生代Eden=1.2GB、S0 = 150MB、S1 = 150MB

1. 如何调优？

（1）、1次Minor GC后，可能Survivor不足或者触发动态年龄判断，对象进入老年代，明显是Survivor空间不足；

新生代调整为2G，老年代为1G，此时Eden：1.6G，每个Survivor：200MB；

解决可能的Survivor不足或者触发动态年龄判断，降低新生代对象进入老年代的概率；

此时JVM参数：

“-Xms3072M -Xmx3072M -Xmn2048M -Xss1M -XX:MetaspaceSize=512M -XX:MaxMetaspaceSize=512M”

（2）、一般系统里的@Service、@Controller之类的注解需要长期存活，这些对象一般也不会很多，可能几十兆，应该让它们尽快进入老年代；

此时JVM参数：

“-Xms3072M -Xmx3072M -Xmn2048M -Xss1M -XX:MetaspaceSize=256M -XX:MaxMetaspaceSize=256M -XX:MaxTenuringThreshold=5”

（3）、 一般情况下，大对象可能需要长期存活和使用，让它直接进入老年代；（根据项目实际情况来确定）

此时JVM参数如下：

“-Xms3072M -Xmx3072M -Xmn2048M -Xss1M -XX:MetaspaceSize=256M -XX:MaxMetaspaceSize=256M -XX:MaxTenuringThreshold=5 -XX:PretenureSizeThreshold=1M”

（4）指定合适的垃圾回收器；

此时JVM参数 ：

“-Xms3072M -Xmx3072M -Xmn2048M -Xss1M -XX:MetaspaceSize=256M -XX:MaxMetaspaceSize=256M -XX:MaxTenuringThreshold=5 -XX:PretenureSizeThreshold=1M -XX:+UseParNewGC -XX:+UseConcMarkSweepGC”

没有使用G1垃圾收集器；

1. 大概每隔几分钟Minor GC之后有大概200MB左右对象进入老年代，推算可能差不多1小时后，才会有接近1GB的对象进入老年代，触发Full GC，然后高峰期一过，可能需要几个小时才会一次Full GC；

-Xms3072M -Xmx3072M -Xmn2048M -Xss1M -XX:-UseCompressedClassPointers -XX:MetaspaceSize=512M -XX:MaxMetaspaceSize=512M -XX:SurvivorRatio=8 -XX:MaxTenuringThreshold=5 -XX:PretenureSizeThreshold=1M -XX:+UseParNewGC -XX:+UseConcMarkSweepGC -XX:CMSInitiatingOccupancyFraction=75 -XX:+UseCMSCompactAtFullCollection

-XX:CMSFullGCsBeforeCompaction=0

1. 优化思路
2. 尽可能让对象在新生代里分配和回收，避免对象频繁进入老年代导致老年代频繁垃圾回收；
3. 给系统充足的内存空间，避免新生代频繁的垃圾回收；
4. 指定合适的垃圾收集器；