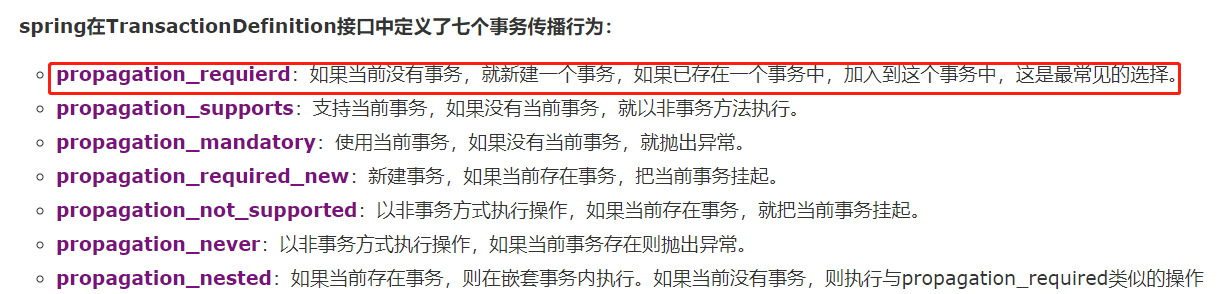
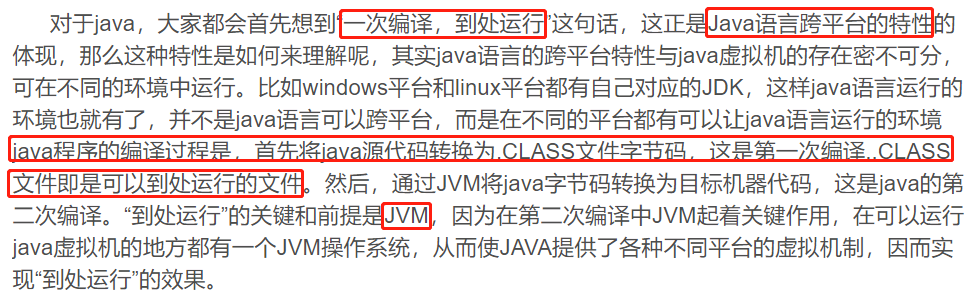
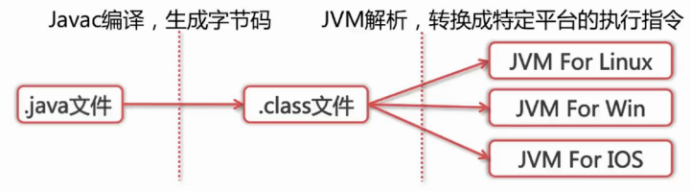


项目启动时，先启动监听器，再启动过滤器







Java源码首先被编译成字节码，再由不同平台的JVM进行解析，JAVA语言在不同的平台上运行时不需要进行重新编译，Java虚拟机在执行字节码的时候，把字节码转换成具体平台上的机器指令。

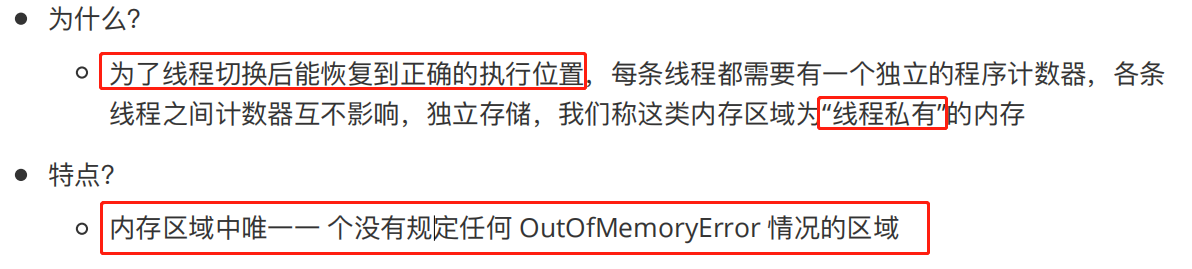
平台无关性，只要编译为class文件，不同平台的jvm都能运行这个文件

⾃从内存管理机制之下，不再需要为没⼀个new操作去写配对的内存分配和回收等代码， 不容易出现内存泄漏和内存溢出等问题

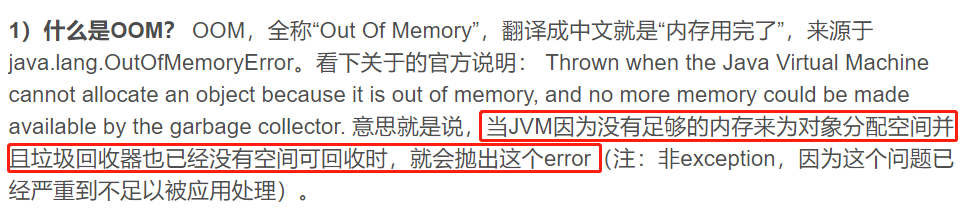
为了线程切换后能恢复到正确的执⾏位置，每条线程都需要有⼀个独⽴的程序计数器，各条线程之间计数器互不影响，独⽴存储，我们称这类内存区域为“线程私有”的内存

程序计数器让cpu并发执行的情况下知道这个任务上一次执行到啥位置

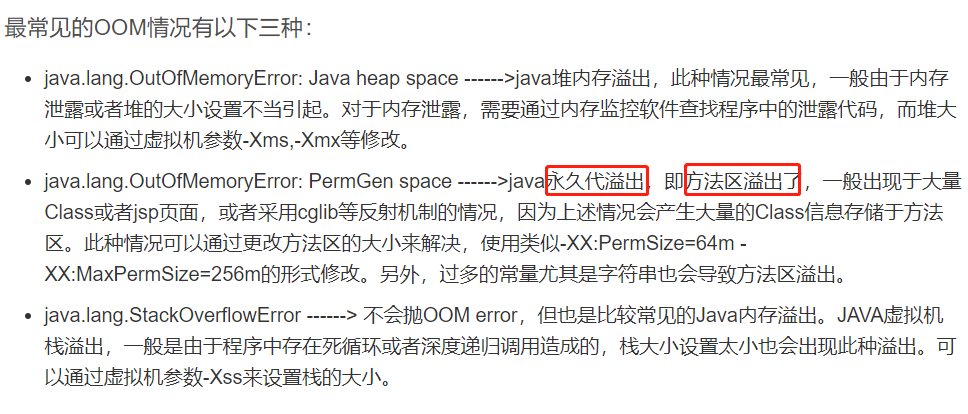
字节码解释器⼯作时就是通过改变这个计数器的值来选取下⼀条需要执⾏的字节码指令，分⽀、循环、跳转、异常处理、线程恢复等基础功能都需要依赖这个计数器来完成



程序计数器是jvm自己操作的，人为操作才会OOM

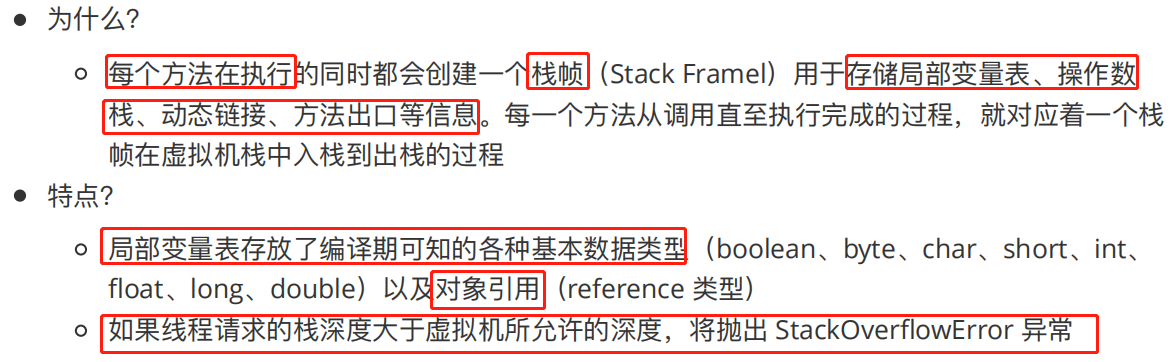


加大堆内存以防OOM



<https://blog.csdn.net/qq_42447950/article/details/81435080>

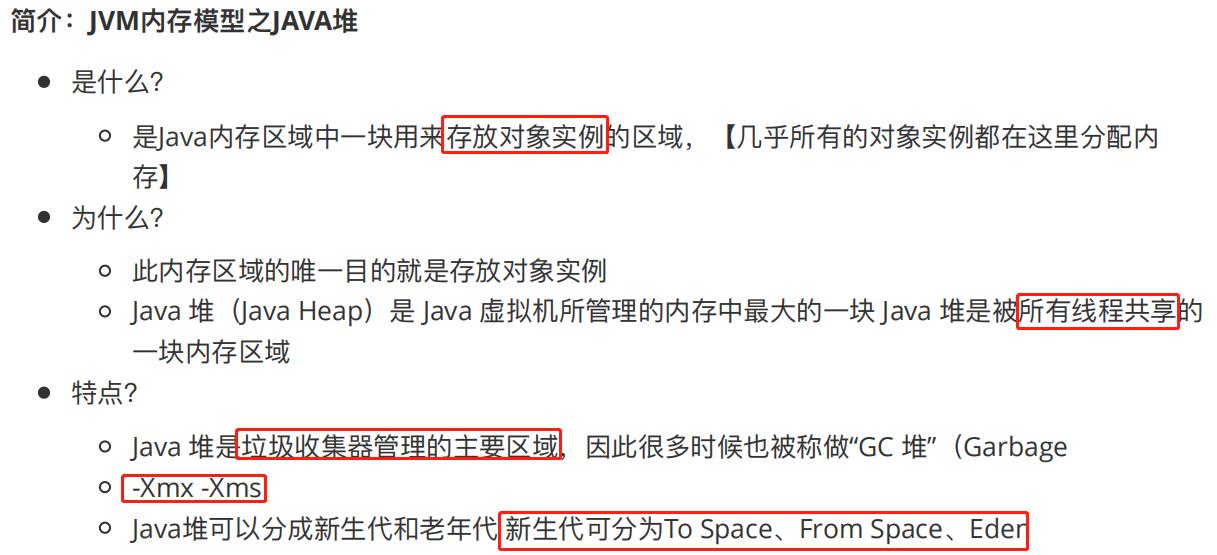
OOM先jconsole查看堆内存是否爆了，然后看类实例数目排序，然后dump下堆内存

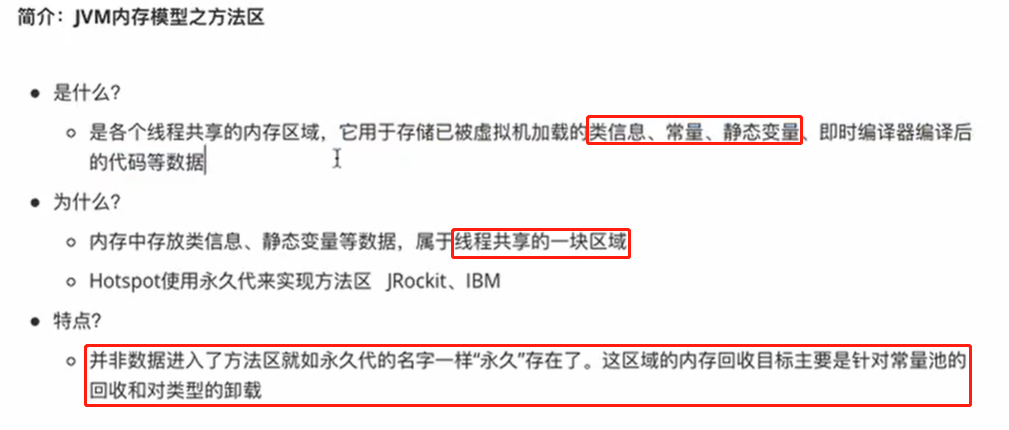


栈引用类型就保存地址（虚拟机栈）

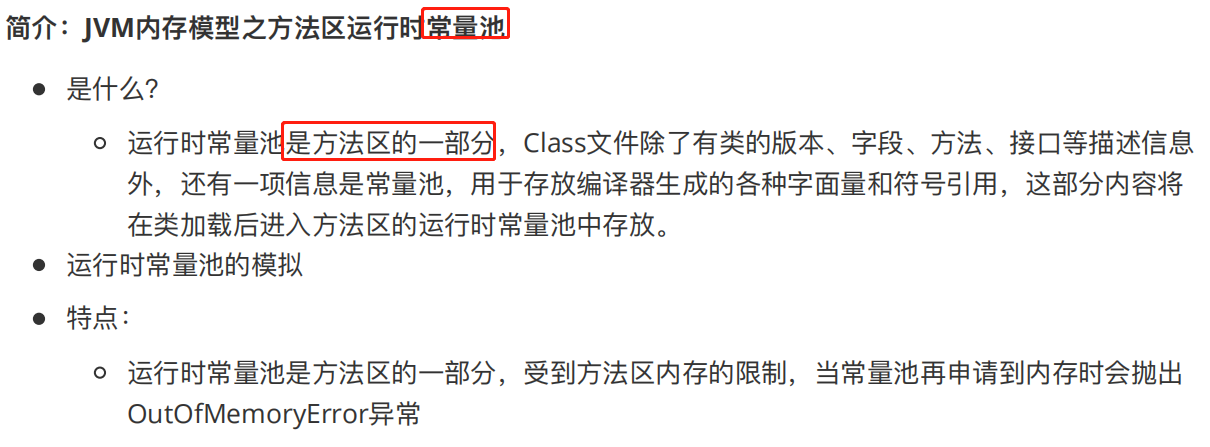
CAS是CPU的一条原子指令

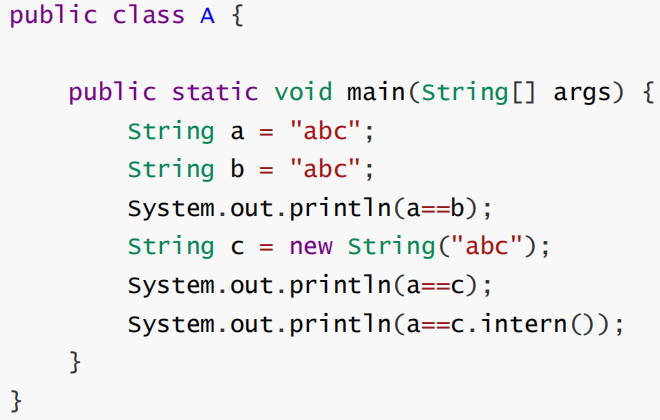
本地⽅法栈，native修饰的方法，其他语言编写的（非java）



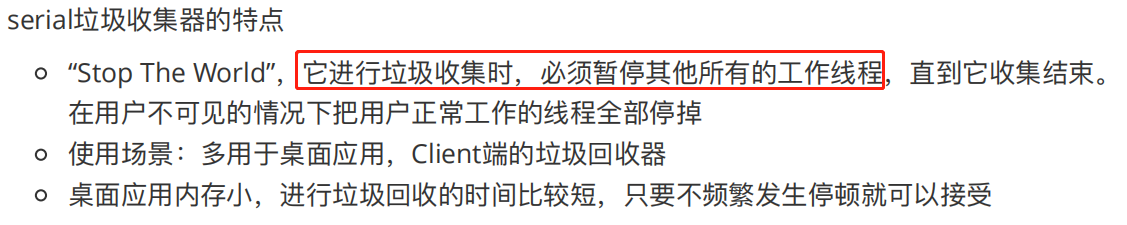


方法区存放类信息，常量，静态变量

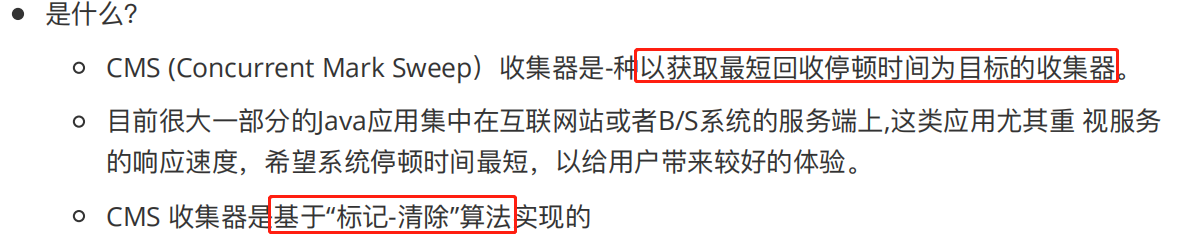


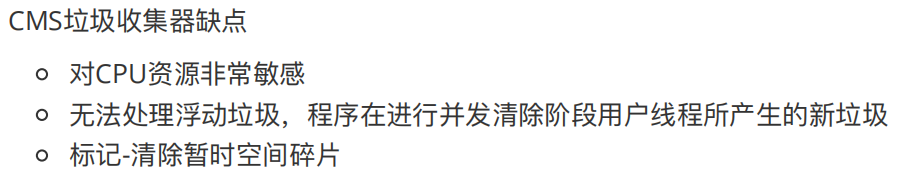


intern将堆的内存移到方法区里



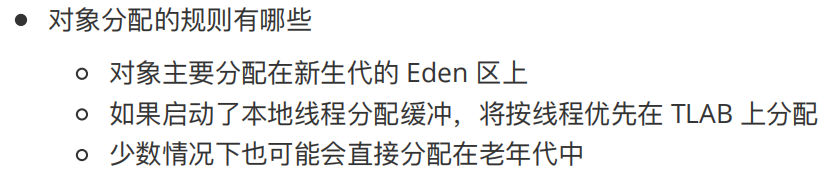
垃圾回收只有一个线程进行回收



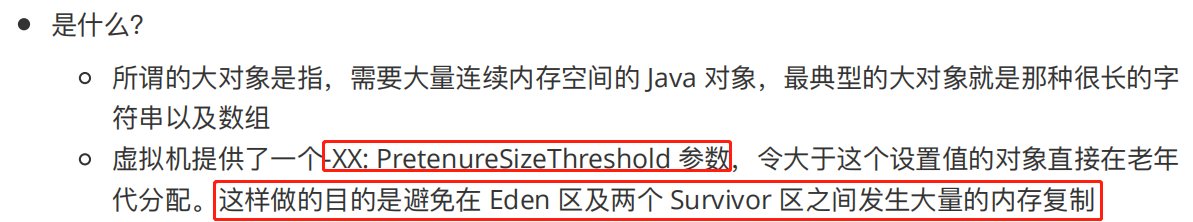


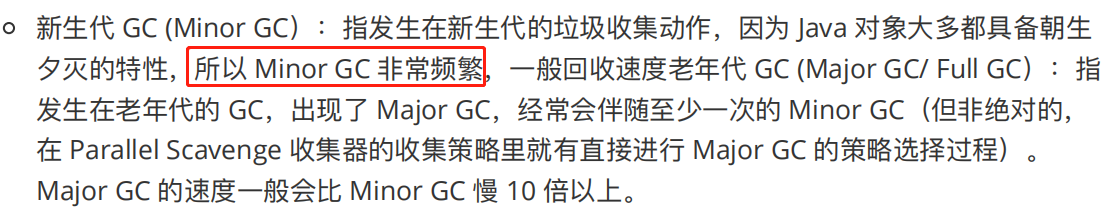
清楚时只处理标记的对象，新的垃圾只能等到下一次，标记清除后空间不连续

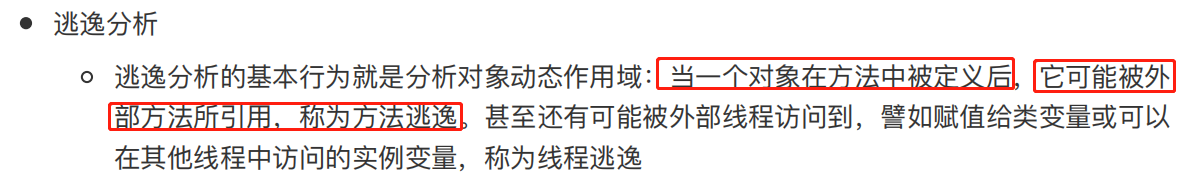
G1是⼀款⾯向服务端应⽤的垃圾收集器，性能高

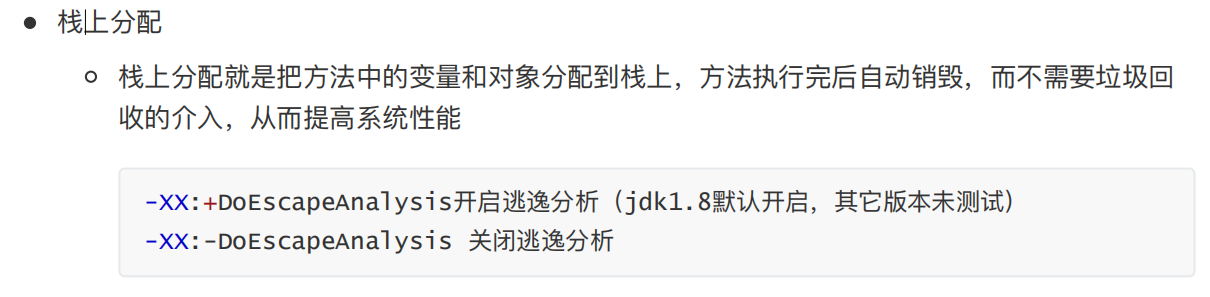


大对象可能会直接分配到老年代中





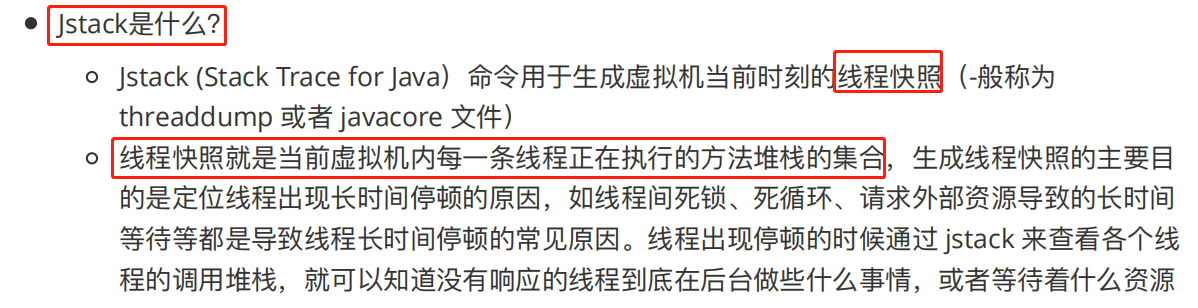




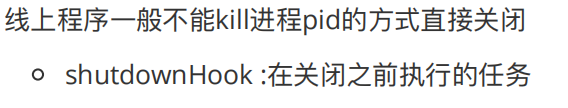


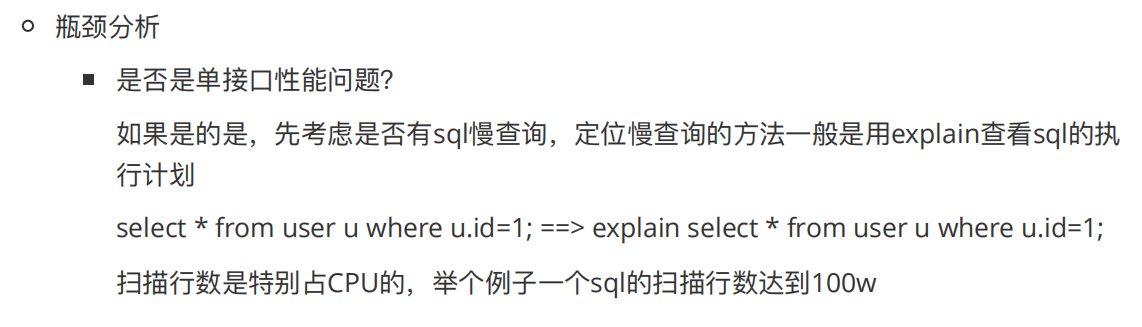
开启逃逸分析就进行栈上分配，不是在堆分配的，不需要GC。栈是方法结束就销毁了



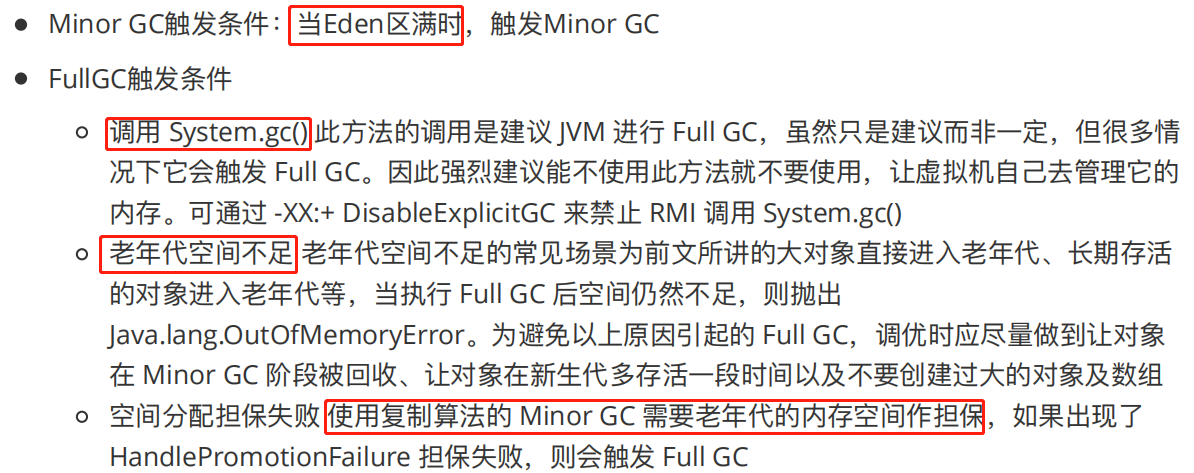


定位线程长时间停顿的问题

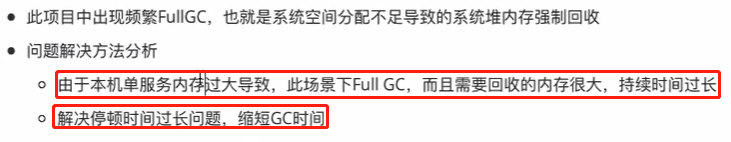




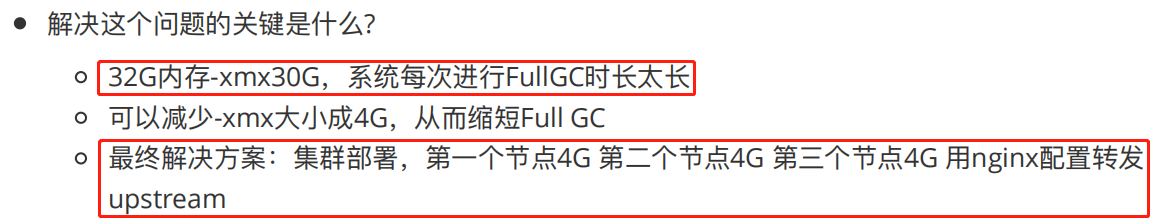
多个接口整体都有问题就要观测磁盘网卡cpu内存的使用情况了

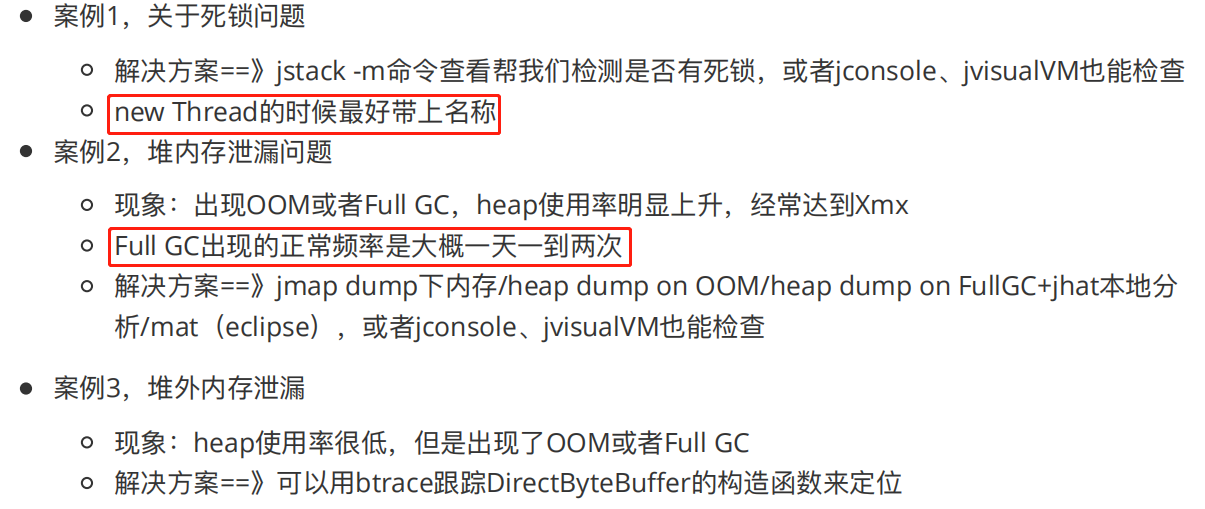


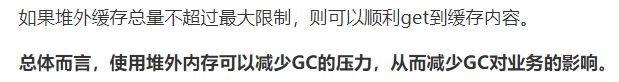
FullGC时间比较长，扫描的是老年代扫描的数据量也多



内存不是约到越好，32G内存，FULL GC的时间就比较长，用户要等待的时间就越长







<https://www.cnblogs.com/reload-sun/p/12216931.html>