**分析解决java.lang.OutOfMemoryError： unable to create new native thread**

**一：起因**

楼主所在公司线上环境出现java.lang.OutOfMemoryError： unable to create new native thread异常，导致系统不可用。在以前的工作中环境中，也遇到过几次这种问题，虽说解决比较简单，但是楼主想策底分析排查这一类问题的造成原因与因素。

Java运行过程中的OutOfMemoryError有三种(三姐妹)，分别是：java.lang.OutOfMemoryError： unable to create new native thread；

java.lang.OutOfMemoryError: Java heap space

java.lang.OutOfMemoryError: PermGen space

本文重点讲解unable to create new native thread这一种。

**二：如何解决与分析问题**

楼主按照惯例解决如上问题---重启服务。原本正常就能OK了，不过这次发现刚开始的时候是OK的，但是过了10多分钟就不OK了(此时的线上系统并发量较大，在200多)，还是有OOM: unable to create new native thread爆出来，导致部分服务不能正常，这是查看系统内存发现所剩无几。到此发现问题所在了，赶紧清理操作系统内存，让空闲内存保证在2G以上，后续系统使用正常，再无问题。

问题解决了，现在就需要分析分析究竟是啥原因导致的这些问题了。在java语言里， 当你创建一个线程的时候，虚拟机会在JVM内存创建一个Thread对象同时创建一个操作系统线程，而这个系统线程的内存用的不是JVMMemory，而是系统中剩下的内存。

Java线程的实现是基于底层系统的线程机制来实现的, 默认一个线程会占用1024K的操作系统内存空间。

百度中关于OOM: unable to create new native thread,很多都在说一个公式: (MaxProcessMemory - JVMMemory - ReservedOsMemory) / (ThreadStackSize) = Number of threads，楼主并没有完全按照这个公式(主要是不太赞同)来验证和复现问题。楼主基于JVM创建一个线程就需要一个创建一个对应的操作系统内存的理论在验证分析问题，默认创建一个线程需要1024K的空间，所以只要操作系统所剩空间不多，就可以复现和验证这些问题。

**三：如何复现问题**

3.1：复现问题思路与代码

楼主有一阿里云ECS，正常使用还剩1.4G空闲内存，楼主写了一个简单的Web应用，规划启动是占用Heap1.1G，Web应用启动好之后调用一个可以生产不能被GC回收内存占用700M，在调用一个接口用于不断的创建线程，看最终会创建多少个线程。

3.2：项目运行复现过程与结果

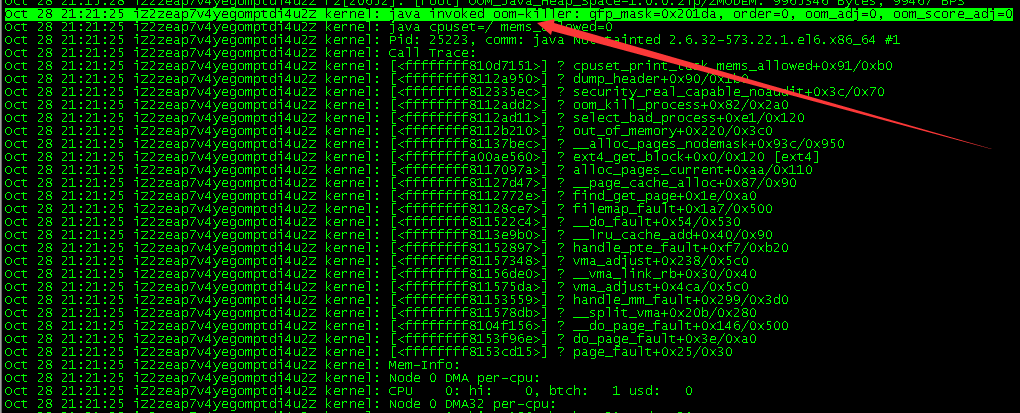
能出现问题工程代码路径: <https://github.com/yun19830206/Yun_PracticeAndDemo/tree/master/OOM_Java_Heap_Space>

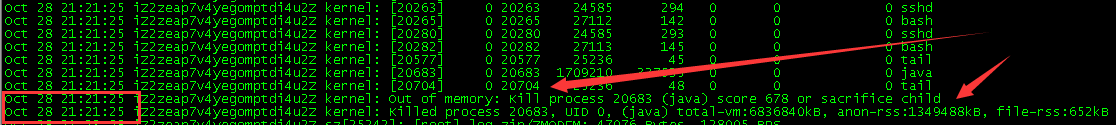
在阿里云的ECS中启动程序，执行如下接口：

http://host:9000/createOneCanotGCBigObject  
 -->第一次调用可以，服务没有异常，没办法在Java VisualVM当中强制回收掉  
http://host:9000/normalRequest  
 -->调用正常接口，可以正常返回。(此时JVM还剩300M左右的空闲量)  
http://host:9000/debugCreateMaxThread  
 -->创建一万个无法结束的线程的接口，也就是查看本应用在现有空闲内存下，看还能创建多少线程(本次创建1.1W个之后应用进程直接没有了)

执行完成第二个接口后，操作系统大概剩余500M的内存，执行完第三个接口后发现，Web应用程序大概创建了1.1W个线程之后，程序直接终止了，和楼主想的大不一样，照理说应该创建几百个线程就能出现unable to create new native thread异常，单实际上并没有。这点楼主花了很长时间去找原因，最终找到原因是楼主以Root账号启动的Web应用，楼主最后在操作系统日志中分析到了日志，是操作系统与Java的保护机制，当发现某个应用占用太多资源，直接将其kill了，查看的日志文件在/var/log/messages中，发现的确与我们分析的一样。

截图截图





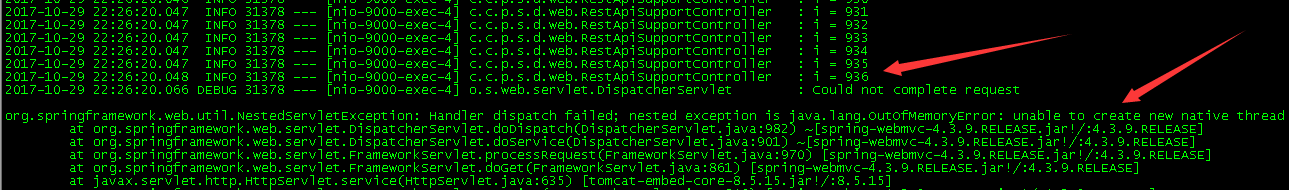
至于为什么创建的线程数量远远大于我们所想象的，可能是操作系统启动内存交换机制，将一些运行较旧的进程的内存swap到的缓存中，以便开辟更多的内存当当前进程使用。

3.3：复现结果说明

楼主改用tomcat账号运行上面的程序：

http://host:9000/createOneCanotGCBigObject  
 -->第一次调用可以，服务没有异常，没办法在Java VisualVM当中强制回收掉  
http://host:9000/normalRequest  
 -->调用正常接口，可以正常返回。(此时JVM还剩300M左右的空闲量)  
http://host:9000/debugCreateMaxThread  
 -->创建一万个无法结束的线程的接口，也就是查看本应用在现有空闲内存下，看还能创建多少线程(本次创建900个线程出现了unable to create new native thread异常，问题复现)

结果与楼主想的大致一直，应用程序日志截图如下：

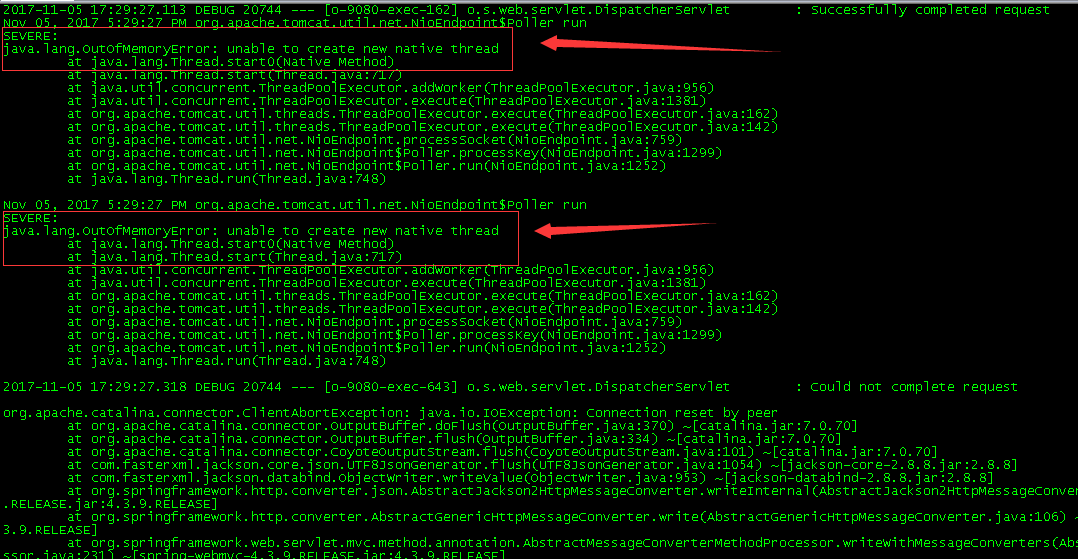


3.4：实际情况复现结果说明(tomcat7使用JDK7启动)

在实际项目中，是不可能有上面的代码来复现问题的，一般都是一个接口耗时比较长一些，然后操作系统的可用内存也比较少了，而此时并发度较高，导致高发过来的线程没有及时执行完成，最终导致OutOfMemoryError: unable to create new native thread。楼主也复现了如下问题，执行接口顺序为：

http://host:9000/createOneCanotGCBigObject  
 -->第一次调用可以，服务没有异常，没办法在Java VisualVM当中强制回收掉  
http://host:9000/normalRequest  
 -->调用正常接口，可以正常返回。(此时JVM还剩300M左右的空闲量)  
http://host:9000/usedFiftySecondApi  
 -->创建一万个无法结束的线程的接口，也就是查看本应用在现有空闲内存下，看还能创建多少线程(本次创建900个线程出现了unable to create new native thread异常，问题复现)

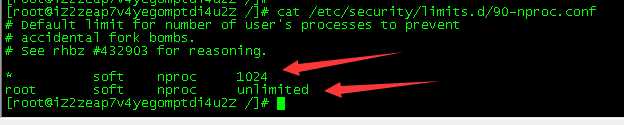
然后自己写一个并发请求的多线程程序，不断调用这个接口(<http://host:9000/usedFiftySecondApi，这个接口返回结果需要50>秒时间)，模拟接口响应时间过慢的情况，模拟高并发请求的情况，在700个线程以内,就会出现OOM了，效果截图如下：



**四：知识拓展---如何部署项目，如何更好写代码**

一个Java进程能创建的线程总数，受多个因素影响：1）操作系统管控的进程可创建线程数；2）还有多少空闲内存可用

cat /etc/security/limits.d/90-nproc.conf



4.1：tomcat配置文件servier.xml中 maxThreads="3000"一般而言也是达不到的，最佳工作线程500，一般不超过1024(系统级别)，准实时1024并发，已经相当大了。

4.2：基于如上的分析，一定要确保一个接口的代码消耗时间尽量的小，如果时间过长，直接导致tomcat能够处理的并发数减少，也更容易发生OutOfMemoryError： unable to create new native thread类型的问题。

**五：其他OOM的分析与解决---堆内存溢出OutOfMemoryError:Java heap space**

还是基于上面的应用，调用如下接口就能复现问题。

http://127.0.0.1:9000/createOneCanotGCBigObject  
 -->第一次调用可以，服务没有异常，没办法在Java VisualVM当中强制回收掉  
http://127.0.0.1:9000/normalRequest  
 -->调用正常接口，可以正常返回。(此时JVM还剩300M左右的空闲量)  
http://127.0.0.1:9000/createOneCanotGCBigObject  
 -->第二次调用无法回收创建大对象的接口，后台直接报.OutOfMemoryError: Java heap space  
http://127.0.0.1:9000/normalRequest  
 -->虽然已经报了OOM,但是由于JDK8的保护机制，其他正常接口，可以正常返回的,如果是低版本JDK应用就不能使用了。(此时JVM还剩300M左右的空闲量)

一般而言，Java Web应用的对内存设置多大合适，需要结合应用在实际运行不同周期内存的最小内存、最大内存占用情况、强制GC回收后内存占用情况，设置合适的对内存大小。

**六：其他OOM的分析与解决---静态区内存溢出OutOfMemoryError：PermGen space**

还是基于上面的应用，把启动Web应用参数的MaxPermSize参数设置小一些就可以复现了。楼主没有刻意的去复现这个文件，此问题比较简单与比较容易解决--直接增加MaxPermSize参数值既可以。 但是一般而言，Java Web应用的静态区MaxPermSize设置多大合适呢，需要结合应用在实际运行不同周期内存的最小内存、最大内存占用情况、强制GC回收后内存占用情况，一般设置上面几种情况平均值的2倍就可以了。

JDK8当中已经没有永久代(又名PermGen)这样的内存区域了，在JDK8当中对应的内存区域叫Metaspace(元空间)，此区域设置不正确也有可能溢出的，溢出特征如下(解决方案和JDK7当中一样的)：

