Tomcat性能优化

1：修改启动内存参数，并修改时区

Set JAVA\_OPTS=-XX:PermSize=64M -XX:MaxPermSize=128m -Xms512m -Xmx1024m;-Duser.timezone=GMT+08;

**-Xms128m JVM初始分配的堆内存**

**-Xmx512m JVM最大允许分配的堆内存，按需分配**

**-XX:PermSize=64M JVM初始分配的非堆内存**

**-XX:MaxPermSize=128M JVM最大允许分配的非堆内存，按需分配**

linux下，在catalina.sh最前面增加：

JAVA\_OPTS="-XX:PermSize=64M -XX:MaxPermSize=128m -Xms512m -Xmx1024m -Duser.timezone=Asia/Shanghai"

1. **堆(Heap)和非堆(Non-heap)内存**

**堆是运行时数据区域，所有实例和数组内存从此分配，**

堆和非堆。简单来说堆就是Java代码可及的内存，是留给开发人员使用的；非堆就是JVM留给自己用的，

所以方法区、JVM内部处理或优化所需的内存(如JIT编译后的代码缓存)、每个类结构(如运行时常数池、字段和方法数据)以及方法和构造方法的代码都在非堆内存中。

1. **堆内存分配**

JVM初始分配的堆内存由-Xms指定，默认是物理内存的1/64；JVM最大分配的堆内存由-Xmx指定，默认是物理内存的1/4。默认空余堆内存小于40%时，JVM就会增大堆直到-Xmx的最大限制；  
 空余堆内存大于70%时，JVM会减少堆直到-Xms的最小限制。因此服务器一般设置-Xms、-Xmx 相等以避免在每次GC 后调整堆的大小。  
 说明：如果-Xmx 不指定或者指定偏小，应用可能会导致java.lang.OutOfMemory错误，此错误来自JVM，不是Throwable的，无法用try...catch捕捉。

1. **非堆内存分配**

 JVM使用-XX:PermSize设置非堆内存初始值，默认是物理内存的1/64；由XX:MaxPermSize设置最大非堆内存的大小，默认是物理内存的1/4。

1. **JVM内存限制(最大值)**

**二.Tomcat并发优化**

**调整Connector并发处理能力**

**maxThread 客户请求最大线程数**

**minSpareThreads tomcat初始化时创建socket线程数**

**maxSpareThreads tomcat 连接器最大空闲socket线程数**

**enableLookups 是否反查域名**

**redirectPort 在需要基于安全通道的场合，把客户请求转发到基于ssl的redirectPort端口**

**acceptAccount 监听端口队列最大数，满了之后客户请求会被拒绝**

**connectionTimeout 连接超时**

**minProcessors 服务器创建时的最小处理线程数**

**maxProcessors 服务器同时处理线程数**

**URIEncoding URL统一编码**

**其中和最大连接数相关的参数为maxProcessors 和 acceptCount 。如果要加大并发连接数，应同时加大这两个参数。**

●compression="on"   打开压缩功能

●compressionMinSize="2048"启用压缩的输出内容大小，默认为2KB  
●noCompressionUserAgents="gozilla,traviata" 对于以下的浏览器，不启用压缩  
●compressableMimeType="text/html,text/xml,text/javascript,text/css,text/plain"　哪些资源类型需要压缩

  Tomcat 的压缩是在客户端请求服务器对应资源后，从服务器端将资源文件压缩，再输出到客户端，由客户端的浏览器负责解压缩并浏览。相对于普通的浏览过程 HTML、CSS、Javascript和Text，它可以节省40% 左右的流量。更为重要的是，它可以对动态生成的，包括CGI、PHP、JSP、ASP、Servlet,SHTML等输出的网页也能进行压缩，压缩效率也很高。但是， 压缩会增加 Tomcat 的负担，因此最好采用Nginx + Tomcat 或者 Apache + Tomcat 方式，将压缩的任务交由 Nginx/Apache 去做。

# Tomcat缓存优化

1. tomcat的maxThreads、acceptCount（最大线程数、最大排队数）
2. maxThreads：tomcat起动的最大线程数，即同时处理的任务个数，默认值为200
3. acceptCount：当tomcat起动的线程数达到最大时，接受排队的请求个数，默认值为100

maxThreads如何配置

第一种极端情况，如果我们的操作是纯粹的计算，那么系统响应时间的主要限制就是cpu的运算能力，此时maxThreads应该尽量设的小，降低同一时间内争抢cpu的线程个数，可以提高计算效率，提高系统的整体处理能力。

第二种极端情况，如果我们的操作纯粹是IO或者数据库，那么响应时间的主要限制就变为等待外部资源，此时maxThreads应该尽量设的大，这样才能提高同时处理请求的个数，从而提高系统整体的处理能力。此情况下因为tomcat同时处理的请求量会比较大，所以需要关注一下tomcat的虚拟机内存设置和linux的open file限制

1. **常见的Java内存溢出有以下三种：**

**Java.lang.OutOfMemoryError java heap space JVM Heap 堆溢出**

**堆初始值：-Xms为1/64 最大空间：-Xmx不超过物理内存**

**Java.lang.OutOfMemoryError PermGen space**

**永久保留区溢出**

**在JVM中如果98％的时间是用于GC，且可用的Heap size 不足2％的时候将抛出此异常信息。**

**java.lang.OutOfMemoryError: PermGen space ---- PermGen space溢出。**

**APP载入class过多，会造成PermGen space 溢出**

**解决方法： 手动设置MaxPermSize大小**

**java.lang.StackOverflowError ---- 栈溢出**

**调用构造函数的 “层”太多了，以致于把栈区溢出了**

**解决方法：修改程序。**