**Tomcat优化**

1. **内核优化**

**fs.file-max = 655350**　　# 系统文件描述符总量

**net.ipv4.ip\_local\_port\_range = 1024 65535**　　# 打开端口范围

**net.ipv4.tcp\_max\_tw\_buckets = 2000**　　# 设置tcp连接时TIME\_WAIT个数

**net.ipv4.tcp\_tw\_recycle = 1**　　# 开启快速tcp TIME\_WAIT快速回收

**net.ipv4.tcp\_tw\_reuse = 1**　　# 开启TIME\_WAIT重用

**net.ipv4.tcp\_syncookies = 1**　　# 开启SYN cookies 当出现syn等待溢出，启用cookies来处理，可防范少量的syn攻击

**net.ipv4.tcp\_syn\_retries = 2**　　# 对于一个新建的tcp连接，内核要发送几个SYN连接请求才决定放弃

**net.ipv4.tcp\_synack\_retries = 2**　　# 这里是三次握手的第二次连接，服务器端发送syn+ack响应 这里决定内核发送次数

**net.ipv4.tcp\_keepalive\_time = 1200**　　# tcp的长连接，这里注意：tcp的长连接与HTTP的长连接不同

**net.ipv4.tcp\_fin\_timeout = 15**　　 # 设置保持在FIN\_WAIT\_2状态的时间

**net.ipv4.tcp\_max\_syn\_backlog = 20000**　　# tcp半连接最大限制数

**net.core.somaxconn = 65535**　# 定义一个监听最大的队列数

**net.core.netdev\_max\_backlog = 65535**　　# 当网络接口比内核处理数据包速度快时，允许送到队列数据包的最大数目

1. **Tomcat内存优化，启动时告诉JVM使用一大块内存（调优内存是最直接的方式）**

在该注释后添加如下代码：

# OS specific support. $var \_must\_ be set to either true or false.

export JAVA\_OPTS="-server -Xms512M -Xmx1400M -Xss512k -XX:+AggressiveOpts -XX:+UseBiasedLocking -XX:+DisableExplicitGC -XX:MaxTenuringThreshold=15 -XX:+UseConcMarkSweepGC -XX:+UseParNewGC -XX:+CMSParallelRemarkEnabled -XX:LargePageSizeInBytes=128m -XX:+UseFastAccessorMethods -XX:+UseCMSInitiatingOccupancyOnly -Djava.awt.headless=true"

JAVA\_OPTS="-server -XX:PermSize=512M -XX:MaxPermSize=1024m -Xms2048m -Xmx2048m"

**export JAVA\_OPTS=”-server -Xms1024m -Xmx1024m”**

-server 启用jdk 的 server 版；

-Xms<size> JVM初始化堆的大小

-Xmx<size> JVM堆的最大值 实际参数大小根据服务器配置或者项目具体设置.（建议内存的一半）

1. **Tomcat线程优化（server.xml配置文件）**

<Connector port="8080" protocol="HTTP/1.1"

maxThreads="1000"

minProcessors="100"

maxProcessors="1000"

minSpareThreads="100"

maxSpareThreads="1000"

enableLookups="false"

URIEncoding="utf-8"

acceptCount="1000"

connectionTimeout="30000"

disableUploadTimeout="ture"

redirectPort="8443" />

minProcessors：最小空闲连接线程数，用于提高系统处理性能，默认值为 10

maxProcessors：最大连接线程数，即：并发处理的最大请求数，默认值为 75

acceptCount：允许的最大连接数，应大于等于 maxProcessors ，默认值为 100

enableLookups：是否反查域名，取值为： true 或 false 。为了提高处理能力，应设置为 false

connectionTimeout：网络连接超时，单位：毫秒。设置为 0 表示永不超时，这样设置有隐患的。通常可设置为 30000 毫秒。

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。

web server允许的最大连接数还受制于操作系统的内核参数设置，通常 Windows 是 2000 个左右， Linux 是 1000 个左右。

maxThreads 客户请求最大线程数

minSpareThreads Tomcat初始化时创建的 socket 线程数

maxSpareThreads Tomcat连接器的最大空闲 socket 线程数

enableLookups 若设为true, 则支持域名解析，可把 ip 地址解析为主机名

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（不能小于maxSpareThreads ）

connectionTimeout 连接超时

minProcessors 服务器创建时的最小处理线程数

maxProcessors 服务器同时最大处理线程数

URIEncoding URL统一编码

tomcat对内存及并发的优化就这些。可根据具体的资源进行调整。重启tomcat，观察日志有无报错。

1. **Tomcat IO优化**

1:同步阻塞IO（JAVA BIO） 同步并阻塞，服务器实现模式为一个连接一个线程(one connection one thread 想想都觉得恐怖,线程可是非常宝贵的资源)，当然可以通过线程池机制改善.

2:JAVA NIO:又分为同步非阻塞IO,异步阻塞IO 与BIO最大的区别one request one thread.可以复用同一个线程处理多个connection(多路复用).

3:,异步非阻塞IO(Java NIO2又叫AIO) 主要与NIO的区别主要是操作系统的底层区别.可以做个比喻:比作快递，NIO就是网购后要自己到官网查下快递是否已经到了(可能是多次)，然后自己去取快递；AIO就是快递员送货上门了(不用关注快递进度)。

BIO方式适用于连接数目比较小且固定的架构，这种方式对服务器资源要求比较高，并发局限于应用中，JDK1.4以前的唯一选择，但程序直观简单易理解.

NIO方式适用于连接数目多且连接比较短（轻操作）的架构，比如聊天服务器，并发局限于应用中，编程比较复杂，JDK1.4开始支持.

AIO方式使用于连接数目多且连接比较长（重操作）的架构，比如相册服务器，充分调用OS参与并发操作，编程比较复杂，JDK7开始支持

在server.xml中

<Connector port="80" protocol="org.apache.coyote.http11.Http11NioProtocol"

connectionTimeout="20000"

URIEncoding="UTF-8"

useBodyEncodingForURI="true"

enableLookups="false"

redirectPort="8443" />

实现对Tomcat的IO切换.

1. **安全加固**

5.1隐藏Tomcat版本信息

5.2禁用Tomcat管理界面

 生产环境一般不适用Tomcat默认的管理界面，这些页面存放在Tomcat 的webapps安装目录下，把该目录下的所有文件删除即可：rm -rf  /usr/local/tomcat8/webapps/\* 另外删除相关的配置文件 host-manager.xml 和 manager.xml，在Tomcat 安装目录 conf/Catalina/localhost目录下。

     注释或删除tomcat\_user.xml 中的所有用户权限

5.3、应用程序安全：

默认 Tomcat 是开启了对war包的热部署的。为了防止被植入木马等恶意程序，因此我们要关闭自动部署。

修改实例：

<Host name="localhost" appBase="" unpackWARs="false" autoDeploy="false">

5.4、更改关闭Tomcat的指令

  server.xml中定义了可以直接关闭 Tomcat 实例的管理端口。我们通过 telnet 连接上该端口之后，输入 SHUTDOWN （此为默认关闭指令）即可关闭 Tomcat 实例（注意，此时虽然实例关闭了，但是进程还是存在的）。由于默认关闭 Tomcat 的端口和指令都很简单。默认端口为8005，指令为SHUTDOWN 。因此我们需要将关闭指令修改复杂一点。

       当然，在新版的 Tomcat 中该端口仅监听在127.0.0.1上，因此大家也不必担心。除非黑客登陆到tomcat本机去执行关闭操作。

       修改实例：

           <Server port="8005" shutdown="9SDKJ29jksjf23sjf0LSDF92JKS9DKkjsd">

        或者禁用8005端口

            <Server port="-1" shutdown="SHUTDOWN">