**[tomcat 性能优化](http://www.cnblogs.com/ggjucheng/archive/2013/04/16/3024731.html)**

tomcat默认参数是为开发环境制定，而非适合生产环境，尤其是内存和线程的配置，默认都很低，容易成为性能瓶颈。

**tomcat内存优化**

linux修改TOMCAT\_HOME/bin/catalina.sh，在前面加入

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

windows修改TOMCAT\_HOME/bin/catalina.bat，在前面加入

set JAVA\_OPTS=-XX:PermSize=64M -XX:MaxPermSize=128m -Xms512m -Xmx1024m

最大堆内存是1024m，对于现在的硬件还是偏低，实施时，还是按照机器具体硬件配置优化。

**tomcat 线程优化**

<Connector port="80" protocol="HTTP/1.1" maxThreads="600" minSpareThreads="100" maxSpareThreads="500" acceptCount="700"

connectionTimeout="20000" redirectPort="8443" />

maxThreads="600"       ///最大线程数  
minSpareThreads="100"///初始化时创建的线程数  
maxSpareThreads="500"///一旦创建的线程超过这个值，Tomcat就会关闭不再需要的socket线程。  
acceptCount="700"//指定当所有可以使用的处理请求的线程数都被使用时，可以放到处理队列中的请求数，超过这个数的请求将不予处理

这里是http connector的优化，如果使用apache和tomcat做集群的负载均衡，并且使用ajp协议做apache和tomcat的协议转发，那么还需要优化ajp connector。

<Connector port="8009" protocol="AJP/1.3" maxThreads="600" minSpareThreads="100" maxSpareThreads="500" acceptCount="700"

connectionTimeout="20000" redirectPort="8443" />

由于tomcat有多个connector，所以tomcat线程的配置，又支持多个connector共享一个线程池。

首先。打开/conf/server.xml，增加

<Executor name="tomcatThreadPool" namePrefix="catalina-exec-" maxThreads="500" minSpareThreads="20" maxIdleTime="60000" />

最大线程500（一般服务器足以），最小空闲线程数20，线程最大空闲时间60秒。

然后，修改<Connector ...>节点，增加executor属性，executor设置为线程池的名字：

<Connector executor="tomcatThreadPool" port="80" protocol="HTTP/1.1" connectionTimeout="60000" keepAliveTimeout="15000" maxKeepAliveRequests="1" redirectPort="443" />

可以多个connector公用1个线程池，所以ajp connector也同样可以设置使用tomcatThreadPool线程池。

**禁用DNS查询**

当web应用程序向要记录客户端的信息时，它也会记录客户端的IP地址或者通过域名服务器查找机器名 转换为IP地址。

DNS查询需要占用网络，并且包括可能从很多很远的服务器或者不起作用的服务器上去获取对应的IP的过程，这样会消耗一定的时间。

修改server.xml文件中的Connector元素，修改属性enableLookups参数值: enableLookups="false"

如果为true，则可以通过调用request.getRemoteHost()进行DNS查询来得到远程客户端的实际主机名，若为false则不进行DNS查询，而是返回其ip地址

**设置session过期时间**

conf\web.xml中通过参数指定：

<session-config>

<session-timeout>180</session-timeout>

</session-config>

单位为分钟。

**Apr插件提高Tomcat性能**

  Tomcat可以使用APR来提供超强的可伸缩性和性能，更好地集成本地服务器技术.

  APR(Apache Portable Runtime)是一个高可移植库，它是Apache HTTP Server 2.x的核心。APR有很多用途，包括访问高级IO功能(例如sendfile,epoll和OpenSSL)，OS级别功能(随机数生成，系统状态等等)，本地进程管理(共享内存，NT管道和UNIX sockets)。这些功能可以使Tomcat作为一个通常的前台WEB服务器，能更好地和其它本地web技术集成，总体上让Java更有效率作为一个高性能web服务器平台而不是简单作为后台容器。

  在产品环境中，特别是直接使用Tomcat做WEB服务器的时候，应该使用Tomcat Native来提高其性能

  要测APR给tomcat带来的好处最好的方法是在慢速网络上（模拟Internet），将Tomcat线程数开到300以上的水平，然后模拟一大堆并发请求。  
  如果不配APR，基本上300个线程狠快就会用满，以后的请求就只好等待。但是配上APR之后，并发的线程数量明显下降，从原来的300可能会马上下降到只有几十，新的请求会毫无阻塞的进来。  
  在局域网环境测，就算是400个并发，也是一瞬间就处理/传输完毕，但是在真实的Internet环境下，页面处理时间只占0.1%都不到，绝大部分时间都用来页面传输。如果不用APR，一个线程同一时间只能处理一个用户，势必会造成阻塞。所以生产环境下用apr是非常必要的。

[复制代码](javascript:void(0);)

(1)安装APR tomcat-native

apr-1.3.8.tar.gz 安装在/usr/local/apr

#tar zxvf apr-1.3.8.tar.gz

#cd apr-1.3.8

#./configure;make;make install

apr-util-1.3.9.tar.gz 安装在/usr/local/apr/lib

#tar zxvf apr-util-1.3.9.tar.gz

#cd apr-util-1.3.9

#./configure --with-apr=/usr/local/apr ----with-java-home=JDK;make;make install

#cd apache-tomcat-6.0.20/bin

#tar zxvf tomcat-native.tar.gz

#cd tomcat-native/jni/native

#./configure --with-apr=/usr/local/apr;make;make install

(2)设置 Tomcat 整合 APR

修改 tomcat 的启动 shell （startup.sh），在该文件中加入启动参数：

CATALINA\_OPTS="$CATALINA\_OPTS -Djava.library.path=/usr/local/apr/lib" 。

(3)判断安装成功:

如果看到下面的启动日志，表示成功。

2007-4-26 15:34:32 org.apache.coyote.http11.Http11AprProtocol init

[复制代码](javascript:void(0);)

参考 <http://blog.sina.com.cn/s/blog_417b97470100glmi.html>

<http://datadig.blog.163.com/blog/static/171229928201082075946726/>