**优化内存**

/bin/catalina.sh添加JAVA\_OPTS参数

jdk1.7

JAVA\_OPTS="

-Djava.awt.headless=true

-Dfile.encoding=UTF-8

-server

-Xms512m

-Xmx1024m

-XX:NewSize=512m

-XX:MaxNewSize=1024M

-XX:PermSize=1024m

-XX:MaxPermSize=1024m

-XX:+DisableExplicitGC"

jdk1.8

1.8版本中已经没有PermSize、MaxPermSize

JAVA\_OPTS="

-Djava.awt.headless=true

-Dfile.encoding=UTF-8

-server

-Xms512m

-Xmx1024m

-XX:NewSize=512m

-XX:MaxNewSize=1024M

-XX:+DisableExplicitGC"

参数说明：  
-Djava.awt.headless：没有设备、键盘或鼠标的模式。有关介绍：[What is Headless mode in Java](https://link.jianshu.com?t=https:/blog.idrsolutions.com/2013/08/what-is-headless-mode-in-java/" \t "_blank)  
-Dfile.encoding： 设置字符集  
-server：jvm的server工作模式，对应的有client工作模式。使用“java -version”可以查看当前工作模式  
-Xms512m：初始Heap大小，使用的最小内存  
-Xmx1024m：Java heap最大值，使用的最大内存  
-XX:NewSize=512m：表示新生代初始内存的大小，应该小于 -Xms的值  
-XX:MaxNewSize=1024M：表示新生代可被分配的内存的最大上限，应该小于 -Xmx的值  
-XX:PermSize=1024m：设定内存的永久保存区域(注：jdk1.8 was removed)  
-XX:MaxPermSize=1024m：设定最大内存的永久保存区域(注：jdk1.8 was removed)  
-XX:+DisableExplicitGC：自动将System.gc()调用转换成一个空操作，即应用中调用System.gc()会变成一个空操作

**并发优化**

1.Tomcat连接相关参数

　　在Tomcat 配置文件 server.xml 中的

　　<Connector port="9027"

　　protocol="HTTP/1.1"

　　maxHttpHeaderSize="8192"

　　minProcessors="100"

　　maxProcessors="1000"

　　acceptCount="1000"

　　redirectPort="8443"

　　disableUploadTimeout="true"/>

2.调整连接器connector的并发处理能力

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

3.Tomcat中的配置示例

　　<Connector port="9027"

　　protocol="HTTP/1.1"

　　maxHttpHeaderSize="8192"

　　maxThreads="1000"

　　minSpareThreads="100"

　　maxSpareThreads="1000"

　　minProcessors="100"

　　maxProcessors="1000"

　　enableLookups="false"

　　URIEncoding="utf-8"

　　acceptCount="1000"

　　redirectPort="8443"

　　disableUploadTimeout="true"/>

**缓存优化**

1. 参数说明

compression 打开压缩功能   
　　compressionMinSize 启用压缩的输出内容大小，这里面默认为2KB   
　　compressableMimeType 压缩类型   
　　connectionTimeout 定义建立客户连接超时的时间. 如果为 -1, 表示不限制建立客户连接的时间

2.Tomcat中的配置示例

<Connector port="9027"

　　protocol="HTTP/1.1"

　　maxHttpHeaderSize="8192"

　　maxThreads="1000"

　　minSpareThreads="100"

　　maxSpareThreads="1000"

　　minProcessors="100"

　　maxProcessors="1000"

　　enableLookups="false"

　　compression="on"

　　compressionMinSize="2048"　compressableMimeType="text/html,text/xml,text/javascript,text/css,text/plain"

　　connectionTimeout="20000"

　　URIEncoding="utf-8"

　　acceptCount="1000"

　　redirectPort="8443"

disableUploadTimeout="true"/>

**参考配置**

Connector 配置增加

　　minSpareThreads="100"

　　maxSpareThreads="1000"

　　minProcessors="100"

　　maxProcessors="1000"

enableLookups="false"

**minSpareThreads** Tomcat初始化时创建的线程数。默认值4。

**maxSpareThreads** 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。默认值50。

**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

**通讯模式**

**bio：** 默认的模式,性能非常低下,没有经过任何优化处理和支持.

**nio：** nio(new I/O)，是Java SE 1.4及后续版本提供的一种新的I/O操作方式(即java.nio包及其子包)。Java nio是一个基于缓冲区、并能提供非阻塞I/O操作的Java API，因此nio也被看成是non-blocking I/O的缩写。它拥有比传统I/O操作(bio)更好的并发运行性能。

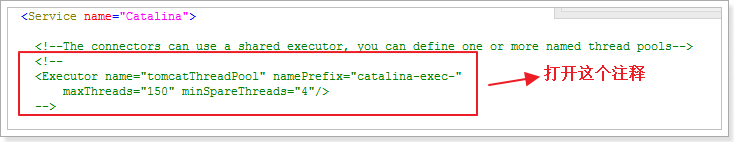
**apr：** 安装起来最困难,但是从操作系统级别来解决异步的IO问题,大幅度的提高性能.

修改server.xml里的Connector节点,修改protocol为org.apache.coyote.http11.Http11NioProtocol

**执行器优化（线程池）**

在tomcat中每一个用户请求都是一个线程，所以可以使用线程池提高性能。

开启并且使用   
配置：



在Connector中指定使用共享线程池



**Executor重要参数说明：**

name：共享线程池的名字。这是Connector为了共享线程池要引用的名字，该名字必须唯一。默认值：None；

namePrefix:在JVM上，每个运行线程都可以有一个name 字符串。这一属性为线程池中每个线程的name字符串设置了一个前缀，Tomcat将把线程号追加到这一前缀的后面。默认值：tomcat-exec-；

maxThreads：该线程池可以容纳的最大线程数。默认值：200；

maxIdleTime：在Tomcat关闭一个空闲线程之前，允许空闲线程持续的时间(以毫秒为单位)。只有当前活跃的线程数大于minSpareThread的值，才会关闭空闲线程。默认值：60000(一分钟)。

minSpareThreads：Tomcat应该始终打开的最小不活跃线程数。默认值：25。

threadPriority：线程的等级。默认是Thread.NORM\_PRIORITY

**Connector重要参数说明：**

executor：表示使用该参数值对应的线程池；

minProcessors：服务器启动时创建的处理请求的线程数；

maxProcessors：最大可以创建的处理请求的线程数；

acceptCount：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。