目　录

Tomcat性能优化

目的

我们通过优化tomcat 来提高网站的并发

影响性能的资源

服务器提供CPU,硬盘,内存,网络;这些对服务器性能有着重要性影响

linux 内核优化

通过优化/etc/sysctl.conf文件参数 可以提高服务器响应性能,其主要是针对TCP的设置优化

SYN Flood防御

SYN Flood攻击大量消耗服务器的**CPU**、内存资源，并占满**SYN**等待队列。相应的，我们修改内核参数即可有效缓解。主要参数如下：

```
net.ipv4.tcp_syncookies = 1
net.ipv4.tcp_max_syn_backlog = 8192
net.ipv4.tcp_synack_retries = 2
```

```

#----------------------------------
#wanglei add to up ipv4   生效：#sysctl -p
# ----------------------------------

#表示开启重用。允许将TIME-WAIT sockets重新用于新的TCP连接，默认为0，表示关闭；
net.ipv4.tcp_tw_reuse = 1
#表示开启TCP连接中TIME-WAIT sockets的快速回收，默认为0，表示关闭；
net.ipv4.tcp_tw_recycle = 1
#修改系统默认的 TIMEOUT 时间。
net.ipv4.tcp_fin_timeout = 30

net.ipv4.tcp_keepalive_time = 1200
net.ipv4.ip_local_port_range = 10000 65000
net.ipv4.tcp_max_syn_backlog = 8192

net.core.netdev_max_backlog = 32768
net.core.somaxconn = 32768
net.core.wmem_default = 8388608
net.core.rmem_default = 8388608
net.core.rmem_max = 873200
net.core.wmem_max = 873200

net.ipv4.tcp_syn_retries = 2
net.ipv4.tcp_wmem = 8192
net.ipv4.tcp_rmem = 32768
net.ipv4.tcp_max_orphans = 3276800
```

tomcat服务器优化

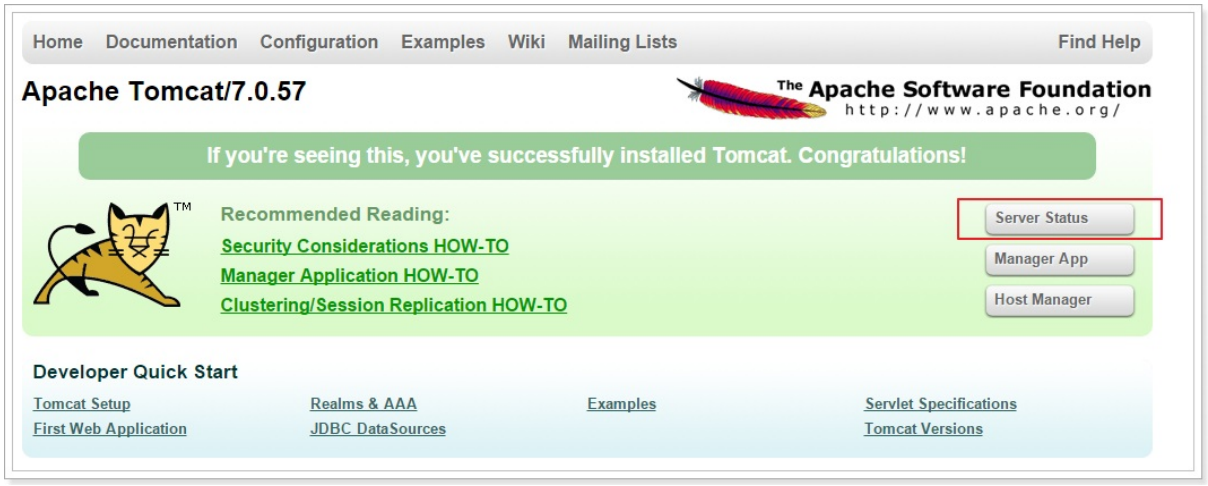
配置账户

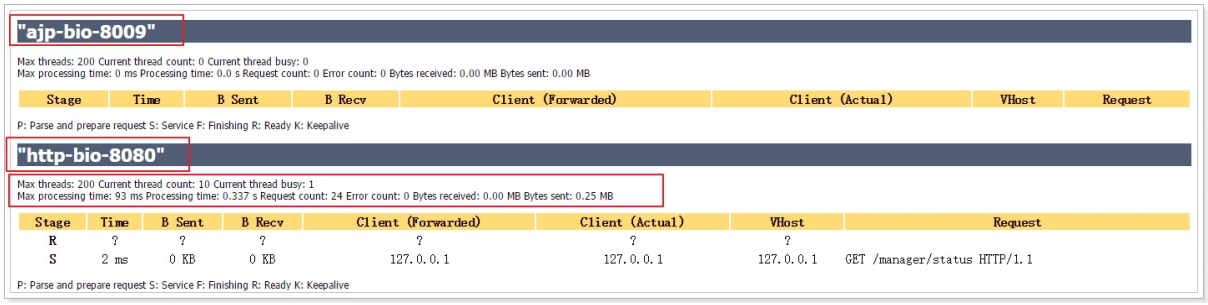
在conf/ tomcat-users.xml下添加用户；

```
<role rolename="manager"/>
<role rolename="manager-gui"/>
<role rolename="admin"/>
```

```
4.  <role rolename="admin-gui"/>
5.  <user username="tomcat" password="tomcat" roles="admin-
    gui,admin,manager-gui,manager"/>
```

启动tomcat，登录查看信息：
http://127.0.0.1:8080/





运行模式

tomcat的运行模式有3种：

1、 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.

2、 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。

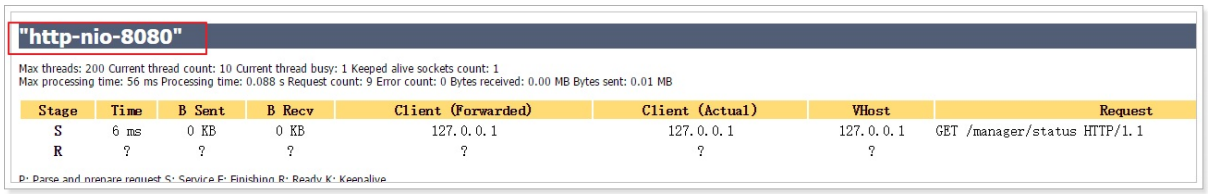
3、 apr

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.

NIO配置

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

```
1.  <Connector port="80"
protocol="org.apache.coyote.http11.Http11NioProtocol"
2.     connectionTimeout="20000"
3.     URIEncoding="UTF-8"
4.     useBodyEncodingForURI="true"
5.     enableLookups="false"
6.     redirectPort="8443" />
```



线程池(Executor)

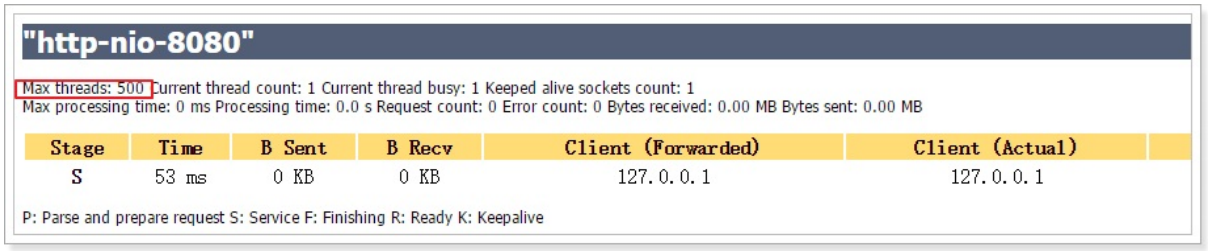
在tomcat中每一个用户请求都是一个线程，所以可以使用线程池提高性能。

```
51        so you may not define subcomponents such as "Valves" at this level.
52        Documentation at /docs/config/service.html
53    -->
54  <Service name="Catalina">
55
56    <!--The connectors can use a shared executor, you can define one or more named thread pools-->
57    <!--
58    <Executor name="tomcatThreadPool" namePrefix="catalina-exec-"
59        maxThreads="150" minSpareThreads="4"/>
60    -->
61
62
63    <!-- A "Connector" represents an endpoint by which requests are received
64         and responses are returned. Documentation at :
65         Java HTTP Connector: /docs/config/http.html (blocking & non-blocking)
66         Java AJP  Connector: /docs/config/ajp.html
67         APR (HTTP/AJP) Connector: /docs/apr.html
68         Define a non-SSL HTTP/1.1 Connector on port 8080
69    -->
70    <Connector port="8080" protocol="HTTP/1.1"
71               connectionTimeout="20000"
72               URIEncoding="UTF-8"
73               disableUploadTimeout="true"
74               acceptCount="2000"
75               maxThreads="1000"
76               redirectPort="8443"
77                useURIValidationHack="false"
78                compression="on"
79                compressionMinSize="2048"
80                compressableMimeType="text/html,text/xml,text/javascript,text/css,text/plain"
81               URLEncoding="UTF-8"  useBodyEncodingForURI="true"  />
82    <!-- A "Connector" using the shared thread pool-->
```

打开注释

```
<Service name="Catalina">

  <!--The connectors can use a shared executor, you can define one or more named thread pools-->
  <!---->
  <Executor name="tomcatThreadPool" namePrefix="catalina-exec-"
      maxThreads="500" minSpareThreads="4"/>


  <!-- A "Connector" represents an endpoint by which requests are received
       and responses are returned. Documentation at :
       Java HTTP Connector: /docs/config/http.html (blocking & non-blocking)
       Java AJP  Connector: /docs/config/ajp.html
       APR (HTTP/AJP) Connector: /docs/apr.html
       Define a non-SSL HTTP/1.1 Connector on port 8080
  -->
  <Connector executor="tomcatThreadPool" port="8080" protocol="org.apache.coyote.http11.Http11NioProtocol"
             connectionTimeout="20000"
             redirectPort="8443" />
```

指定线程池



线程池配置参数:

参数	值
threadPriority （优先级）	(int) The thread priority for threads in the executor, the default is 5 (the value of **theThread.NORM_PRIORITY** constant)
daemon（守护进程）	(boolean) Whether the threads should be daemon threads or not, the default is **true**
namePrefix（名称前缀）	(String) The name prefix for each thread created by the executor. The thread name for an individual thread will be **namePrefix+threadNumber**
maxThreads（最大线程数）	(int) The max number of active threads in this pool, default is **200**
minSpareThreads（最小活跃线程数）	(int) The minimum number of threads always kept alive, default is **25**
maxIdleTime（空闲线程等待时间）	(int) The number of milliseconds before an idle thread shutsdown, unless the number of active threads are less or equal to minSpareThreads. Default value is 60000(1 minute)
maxQueueSize（最大的等待队里数，超过则请求拒绝）	(int) The maximum number of runnable tasks that can queue up awaiting execution before we reject them. Default value is Integer.MAX_VALUE
prestartminSpareThreads（是否在启动时就生成minSpareThreads个线程）	(boolean) Whether minSpareThreads should be started when starting the Executor or not, the default is false
threadRenewalDelay（重建线程的时间间隔）	(long) If a ThreadLocalLeakPreventionListener is configured, it will notify this executor about stopped contexts. After a context is stopped, threads in the pool are renewed. To avoid renewing all threads at the same time, this option sets a delay between renewal of any 2 threads. The value is in ms, default value is 1000 ms. If value is negative, threads are not renewed.。重建线程池内的线程时，为了避免线程同时重建，每隔threadRenewalDelay（单位： ms ）重建一个线程。默认值为1000 ，设置为负则不重建

最佳实践

```
<Connector port="8080"
               namePrefix="catalina-exec-"
               executor="tomcatThreadPool"
               protocol="HTTP/1.1"
              connectionTimeout="20000"
              URIEncoding="UTF-8"
              disableUploadTimeout="true"
              acceptCount="2000"
              maxThreads="800"
              minSpareThreads="100"
              maxQueueSize="100"
              redirectPort="8443"
              prestartminSpareThreads="true"
              useURIValidationHack="false"
              compression="on"
              compressionMinSize="2048"
compressableMimeType="text/html,text/xml,text/javascript,text/css,text/
              URLEncoding="UTF-8"  useBodyEncodingForURI="true"  />
```

连接器（Connector）

Connector是Tomcat接收请求的入口，每个Connector有自己专属的监听端口

Connector有两种：HTTP Connector和AJP Connector

通用属性

参数	值
allowTrace	如果需要服务器能够处理用户的HAED/TRACE请求，这个值应该设置为true，默认值是false
asyncTimeout	默认超不时候以毫秒为单位的异步恳求。若是没有指定，该属性被设置为10000（10秒）。
enableLookups	若是你想request.getRemoteHost（）的调用 履行，以便返回的长途客户端的实际主机名的DNS查询，则设置为true。设置为false时跳过DNS查找，并返回字符串情势的IP地址（从而提高性能）。默认景象下，禁用DNS查找。
maxHeaderCount	容器允许的请求头字段的最大数目。请求中包含比指定的限制更多的头字段将被拒绝。值小于0表示没有限制。如果没有指定，默认设置为100。

maxParameterCount	将被容器自动解析的最大数量的参数和值对（GET加上POST）。参数值对超出此限制将被忽略。值小于0表示没有限制。如果没有指定，默认为10000。请注意， FailedRequestFilter 过滤器可以用来拒绝达到了极限值的请求。
maxPostSize	将被容器以FORM URL参数形式处理的最大长度（以字节为单位）的POST。通过设置此属性的值小于或等于0可以禁用该限制。如果没有指定，该属性被设置为2097152（2兆字节）。
maxSavePostSize	将被容器在FORM或CLIENT-CERT认证中保存/缓冲的POST的最大尺寸（以字节为单位）。对于这两种类型的身份验证，在用户身份验证之 前，POST将被保存/缓冲。对于POST CLIENT-CERT认证，处理该请求的SSL握手和缓冲清空期间，POST将被缓存。对于Form认证，POST将被保存，同时用户将被重定向到登陆 表单。POST将被一直保留直到用户成功认证或者认证请求关联的会话超时。将此属性设置为-1可以禁用此限制。将此属性设置为0，POST数据在身份验证 过程中将不被保存。如果没有指定，该属性设置为4096（4千字节）。
parseBodyMethods	以逗号分隔的HTTP方法列表，通过方法列表，等同于POST方法，request 正文将被解析成请求参数。这在RESTful应用程序要支持以POST式的语义解析PUT请求中是非常有用的。需要注意的是设置其他值（不是POST）会导致Tomcat的行为违反servlet规范的目的。在这里为了符合HTTP规范明确禁止HTTP方法TRACE。默认值是POST
port	TCP端口号，连接器利用该端口号将创建一个服务器套接字，并等待传入的连接。你的操作系统将只允许一个服务器应用程序在一个特定的IP地址侦听特定的端口号。如果使用特殊值0（零），则Tomcat将为连接器随机选择一个空闲的端口。这是通常只用在嵌入式和测试应用程序。
protocol	设置协议来处理传入流量。默认值是 HTTP/1.1，将使用自动切换机制来选择阻塞的基于Java的连接器或APR /native 为基础的连接器。如果PATH（Windows）或LD_LIBRARY_PATH（在大多数Unix系统）的环境变量包含在Tomcat的本地库里，APR /native 连接器将被使用。如果在本地库中无法找到，阻断基于Java的连接器将被使用。需要注意的是使用HTTPS比Java连接器与APR /native 连接器有不同的设置。一个明确的协议，而不是依靠上述自动切换机构，可用以下值： org.apache.coyote.http11.Http11Protocol -阻塞式的Java连接器 org.apache.coyote.http11.Http11NioProtocol -不阻塞Java连接器 org.apache.coyote.http11.Http11AprProtocol的 -的APR / native 连接器 也可以使用的用户自定义的实现。看一看在我们的连接器比较图。Java连接器，HTTP和HTTPS，配置是相同的。 APR连接器和APR特定的SSL设置的更多信息，请访问APR文档
proxyName	如果这个连接正在使用的代理服务器配置，配置该属性指定的服务器的名称，可以调用request.getServerName（）返回。有关更多信息，请参见代理支持。
proxyPort	如果这个连接正在使用的代理服务器配置，配置该属性指定服务器端口，可以调用request.getServerPort（）返回。有关更多信息，请参见代理支持。
redirectPort	如果该连接器支持非SSL请求，并且接收到的请求为满足安全约束需要SSL传输， Catalina 将自动将请求重定向到指定的端口号。

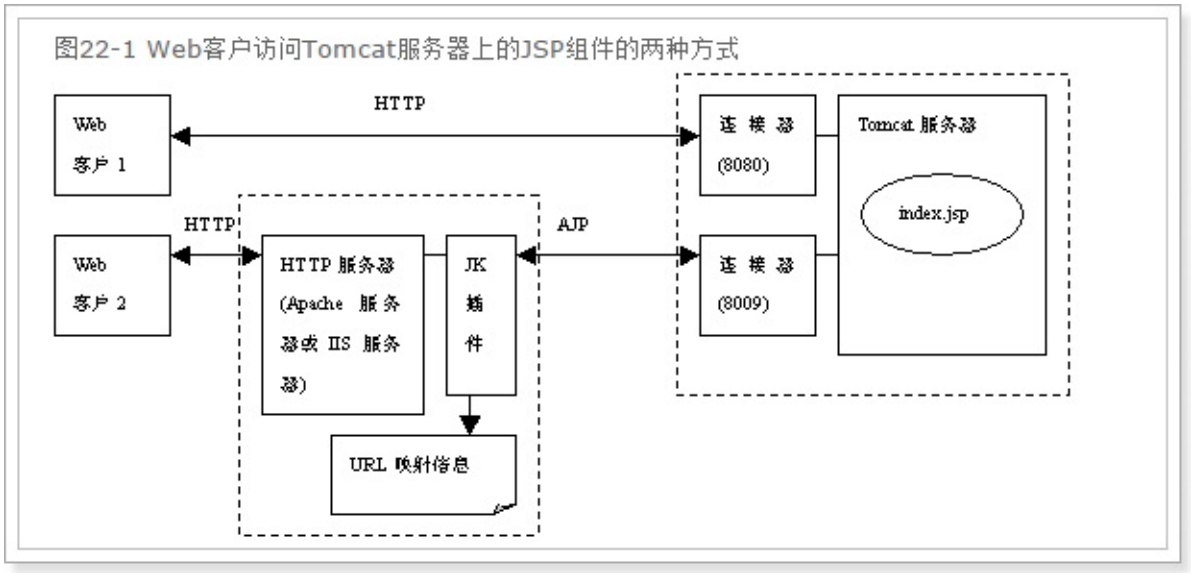
scheme	将该属性设置为你想调用request.getScheme（）返回的协议的名称。例如，对于SSL连接器，你会将此属性设置为“HTTPS”。默认值是“ HTTP ”。
secure	如果你想调用request.isSecure（）收到此连接器的请求返回true，请该该属性设置为true。您希望SSL连接器或非SSL连接器接收数据通过一个SSL加速器，像加密卡，SSL设备，甚至一个web服务器。默认值是假的。
URIEncoding	这将指定使用的字符编码，来解码URI字符。如果没有指定，ISO-8859-1将被使用。
useBodyEncodingForURI	这指定是否应该用于URI查询参数，而不是使用URIEncoding contentType中指定的编码。此设置兼容性Tomcat 4.1.x版（该版在contentType中指定编码，或者使用request.setCharacterEncoding的方法显式设置（参数为URL传来的值）。默认值false。
useIPVHosts	将该属性设置为true会导致Tomcat使用收到请求的IP地址，来确定将请求发送到哪个主机。默认值是假的。
xpoweredBy	将此属性设置为true会导致Tomcat支持使用Servlet规范的通知，（在规范中推荐使用头字段）。默认值是假的。

最佳实践

```
-->
<Connector executor="tomcatThreadPool" port="8080" protocol="org.apache.coyote.http11.Http11NioProtocol"
           connectionTimeout="20000"
           redirectPort="8443"
           enableLookups="false"
           maxPostSize="10485760"
           URIEncoding="UTF-8"
           acceptCount="100"
           acceptorThreadCount="2"
           disableUploadTimeout="true"
           maxConnections="10000"
           SSLEnabled="false"/>
<!-- A "Connector" using the shared thread pool-->
<!--
```

禁用**AJP**连接器

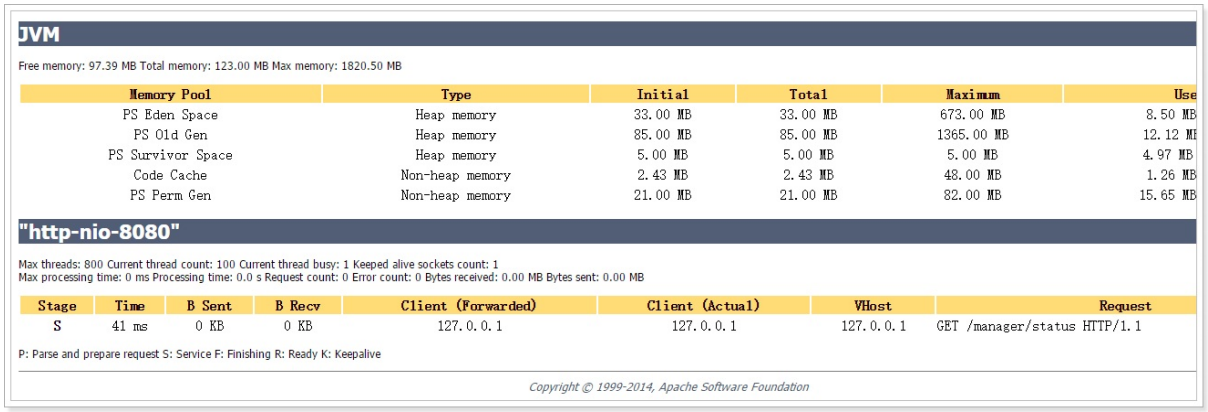
AJP（Apache JServer Protocol）
AJPv13协议是面向包的。WEB服务器和Servlet容器通过TCP连接来交互；为了节省SOCKET创建的昂贵代价，WEB服务器会尝试维护一个永久TCP连接到servlet容器，并且在多个请求和响应周期过程会重用连接。



我们一般是使用Nginx+tomcat的架构，所以用不着AJP协议，所以把AJP连接器禁用。

```
            clientAuth="false" sslProtocol="TLS" />
    -->

    <!-- Define an AJP 1.3 Connector on port 8009 -->
    <!--<Connector port="8009" protocol="AJP/1.3" redirectPort="8443" />-->


    <!-- An Engine represents the entry point (within Catalina) that processes
         every request.  The Engine implementation for Tomcat stand alone
         analyzes the HTTP headers included with the request, and passes them
         on to the appropriate Host (virtual host).
```

在管理界面中看不到ajp了：



JVM

Free memory: 97.39 MB Total memory: 123.00 MB Max memory: 1820.50 MB

Memory Pool	Type	Initial	Total	Maximum	Use
PS Eden Space	Heap memory	33.00 MB	33.00 MB	673.00 MB	8.50 MB
PS Old Gen	Heap memory	85.00 MB	85.00 MB	1365.00 MB	12.12 MB
PS Survivor Space	Heap memory	5.00 MB	5.00 MB	5.00 MB	4.97 MB
Code Cache	Non-heap memory	2.43 MB	2.43 MB	48.00 MB	1.26 MB
PS Perm Gen	Non-heap memory	21.00 MB	21.00 MB	82.00 MB	15.65 MB

"http-nio-8080"

Max threads: 800 Current thread count: 100 Current thread busy: 1 Keeped alive sockets count: 1
Max processing time: 0 ms Processing time: 0.0 s Request count: 0 Error count: 0 Bytes received: 0.00 MB Bytes sent: 0.00 MB

Stage	Time	B Sent	B Recv	Client (Forwarded)	Client (Actual)	VHost	Request
S	41 ms	0 KB	0 KB	127.0.0.1	127.0.0.1	127.0.0.1	GET /manager/status HTTP/1.1

P: Parse and prepare request S: Service F: Finishing R: Ready K: Keepalive

JVM参数的优化

适当调整Tomcat的运行JVM参数可以提升整体性能。

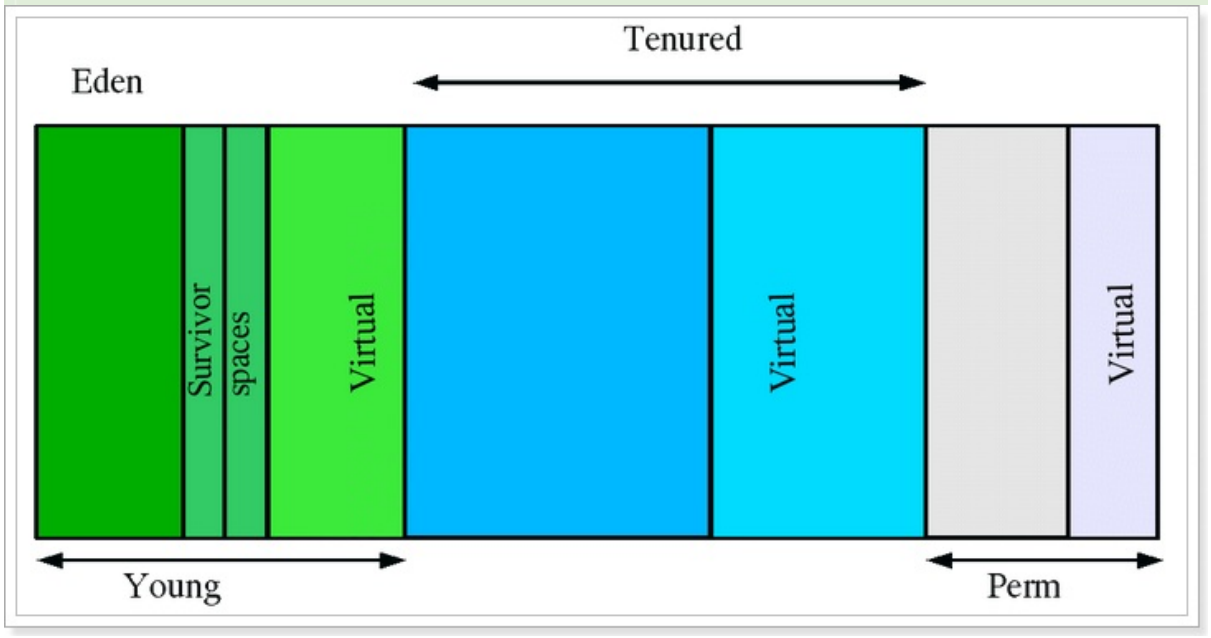
Java栈

Java栈是与每一个线程关联的，JVM在创建每一个线程的时候，会分配一定的栈空间给线程。它主要用来存储线程执行过程中的局部变量，方法的返回值，以及方法调用上下文。栈空间随着线程的终止而释放。

Java堆

Java中堆是由所有的线程共享的一块内存区域，堆用来保存各种JAVA对象，比如数组，线程对象等。

```
-Xmn2g: 设置年轻代大小为2G。整个JVM内存大小=年轻代大小 + 年老代大小 + 持久代大小

-----------------------------------------------------------------------
|   Eden        |   S0  |   S1  |   Old 老年代   |       Per(持久)    |
|   新对象       |       |       |               |                   |
-----------------------------------------------------------------------

-Xms2560m               初始化堆
-Xmx2560m               堆准许
-Xmn1200m               设置年轻代大小 要保证够用 够http发请求 : 一般游戏配置Xmn:Xmx=1:3

-XX:PermSize=256m       持久代
-Xss256k                每个线程堆栈大小

-XX:NewRatio=2          (Eden+S0+S1):Old=1:2
-XX:SurvivorRatio=3     (S0+S1):Eden=2:3      S区=2/5=Xmn1200m*2/5=480   Eden=Xmn1200m*4/5=960


Cache-Control:max-age=300,must-revalidate
```



Young 年轻区（代）

Young区被划分为三部分，Eden区和两个大小严格相同的Survivor区，其中，Survivor区间中，某一时刻只有其中一个是被使用的，另外一个留做垃圾收集时复制对象用，在Eden区间变满的时候，GC就会将存活的对象移到空闲的Survivor区间中，根据JVM的策略，在经过几次垃圾收集后，任然存活于Survivor的对象将被移动到Tenured区间。

◆ Tenured 年老区

Tenured区主要保存生命周期长的对象，一般是一些老的对象，当一些对象在Young复制转移一定的次数以后，对象就会被转移到Tenured区，一般如果系统中用了application级别的缓存，缓存中的对象往往会被转移到这一区间。

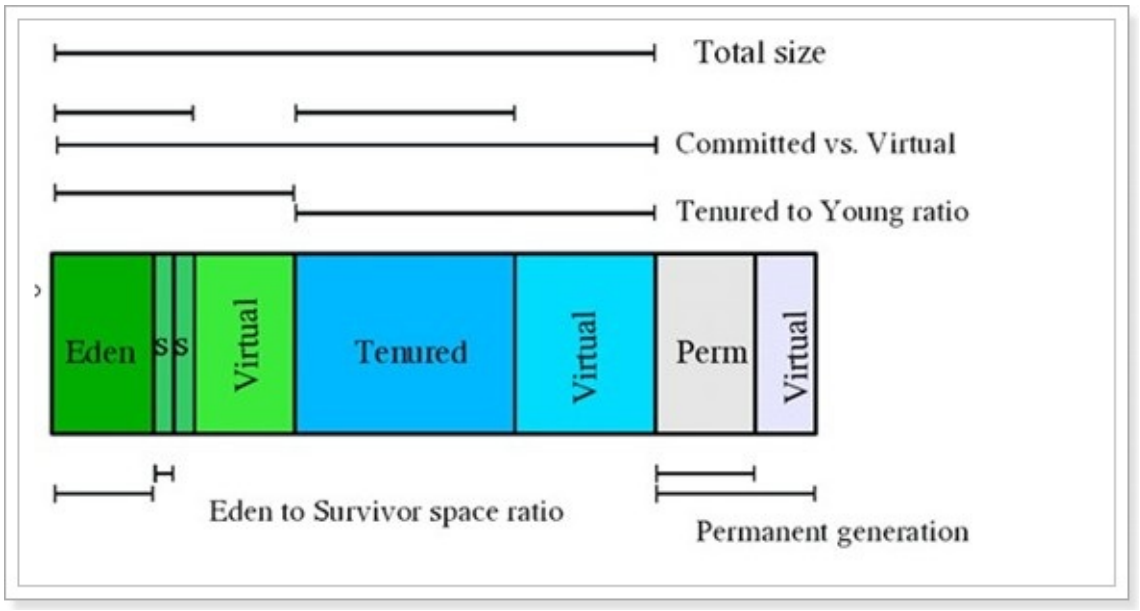
◆ Perm 永久区

Perm代主要保存class,method,filed对象，这部份的空间一般不会溢出，除非一次性加载了很多的类，不过在涉及到热部署的应用服务器的时候，有时候会遇到java.lang.OutOfMemoryError : PermGen space 的错误，造成这个错误的很大原因就有可能是每次都重新部署，但是重新部署后，类的class没有被卸载掉，这样就造成了大量的class对象保存在了perm中，这种情况下，一般重新启动应用服务器可以解决问题。

Virtual区：
最大内存和初始内存的差值，就是Virtual区。

设置区大小

JVM提供了相应的参数来对内存大小进行配置。正如上面描述，JVM中堆被分为了3个大的区间，同时JVM也提供了一些选项对Young,Tenured的大小进行控制。



Total Heap

-Xms ：指定了JVM初始启动以后初始化内存

-Xmx：指定JVM堆得最大内存，在JVM启动以后，会分配-Xmx参数指定大小的内存给JVM，但是不一定全部使用，JVM会根据-Xms参数来调节真正用于JVM的内存

-Xmx -Xms之差就是三个Virtual空间的大小

◆ Young Generation

-XX:NewRatio=8意味着tenured 和 young的比值8：1，这样eden+2*survivor=1/9

堆内存

-XX:SurvivorRatio=32意味着eden和一个survivor的比值是32：1，这样一个Survivor就占Young区的1/34.

-Xmn 参数设置了年轻代的大小

◆ Perm Generation

-XX:PermSize=16M -XX:MaxPermSize=64M

Thread Stack

-XX:Xss=128K

常用参数

修改文件：bin/catalina.sh

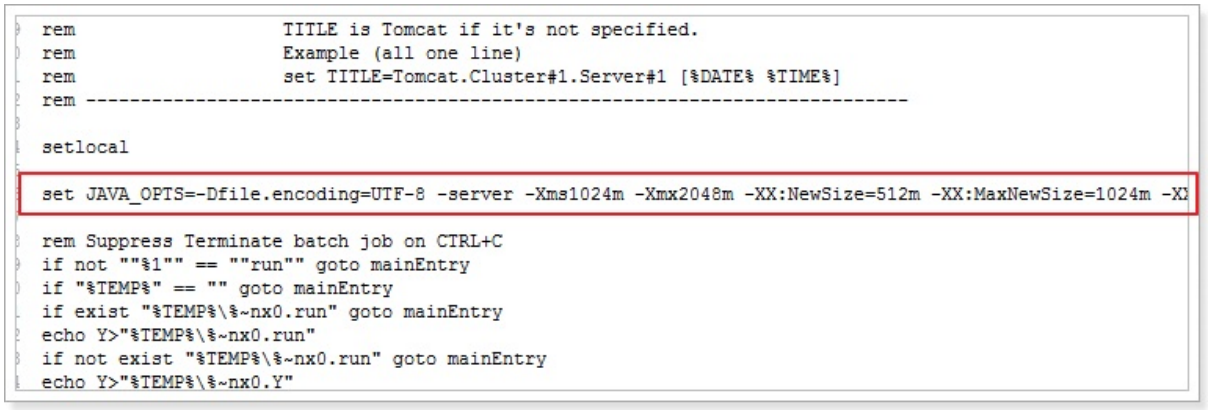
```
JAVA_OPTS="-Dfile.encoding=UTF-8 -server -Xms1024m -Xmx1024m -
XX:NewSize=512m -XX:MaxNewSize=512m -XX:PermSize=256m -XX:MaxPermSize=256m
-XX:NewRatio=2 -XX:MaxTenuringThreshold=50 -XX:+DisableExplicitGC"
```

参数说明：
1、 file.encoding 默认文件编码
2、 -Xmx1024m 设置JVM最大可用内存为1024MB
3、 -Xms1024m 设置JVM最小内存为1024m。此值可以设置与-Xmx相同，以避免每次垃圾回收完成后JVM重新分配内存。
4、 -XX:NewSize 设置年轻代大小
5、 XX:MaxNewSize 设置最大的年轻代大小
6、 -XX:PermSize 设置永久代大小
7、 -XX:MaxPermSize 设置最大永久代大小
8、 -XX:NewRatio=4:设置年轻代（包括Eden和两个Survivor区）与终身代的比值（除去永久代）。设置为4，则年轻代与终身代所占比值为1：4，年轻代占整个堆栈的1/5
9、 -XX:MaxTenuringThreshold=0：设置垃圾最大年龄，默认为：15。如果设置为0的话，则年轻代对象不经过Survivor区，直接进入年老代。对于年老代比较多的应用，可以提高效率。如果将此值设置为一个较大值，则年轻代对象会在Survivor区进行多次复制，这样可以增加对象再年轻代的存活时间，增加在年轻代即被回收的概论。
10、 -XX:+DisableExplicitGC这个将会忽略手动调用GC的代码使得 System.gc()的调用就会变成一个空调用，完全不会触发任何GC

在tomcat中设置JVM参数

修改bin/catalina.bat文件设置参数（第一行）

```
set JAVA_OPTS=-Dfile.encoding=UTF-8 -server -Xms1024m -Xmx2048m -
XX:NewSize=512m -XX:MaxNewSize=1024m -XX:PermSize=256m -
XX:MaxPermSize=256m -XX:MaxTenuringThreshold=10 -XX:NewRatio=2 -
XX:+DisableExplicitGC
```



linux

修改bin/catalina.sh文件参数（第一行）

```
JAVA_OPTS=”-Dfile.encoding=UTF-8 -server -Xms1024m -Xmx2048m -
XX:NewSize=512m -XX:MaxNewSize=1024m -XX:PermSize=256m -
XX:MaxPermSize=256m -XX:MaxTenuringThreshold=10 -XX:NewRatio=2 -
XX:+DisableExplicitGC”
```