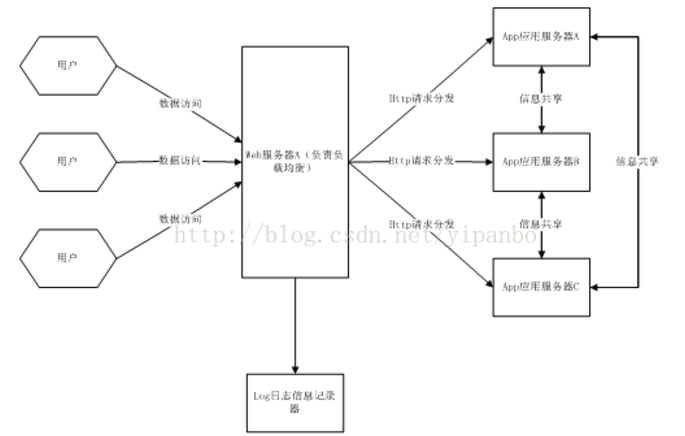
多应用服务器的简单流程图大致如下：



为了实现这个原理我们就需要解决两个问题：

1：如何实现多应用服务器间的session共享：（一台服务器崩溃，另外一台服务器可以继续支持）

2：如何分发请求到各个应用服务器实现压力分解：（这里的解决方案是用apache做 web服务器）

1. **准备条件：(以windows32位为主)**

|  |  |
| --- | --- |
| **材料** | **本次搭建所用具体版本** |
| **Apache HTTP Server** | httpd-2.2.25-win32-x86-no\_ssl.msi |
| **Tomcat  N个** | apache-tomcat-7.0.73-windows-x8611.zip |
| **Mod\_jk** | tomcat-connectors-1.2.37-windows-i386-httpd-2.2.x.zip |
| **用于部署的项目** | Student\_Course\_003.war |

|  |
| --- |
|  |
|  |
|  |
|  |

下载地址

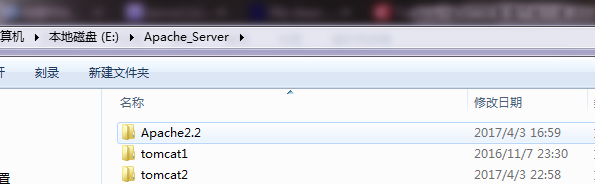
|  |  |
| --- | --- |
| **材料** | **下载地址** |
| **Apache2.2** | <http://httpd.apache.org/download.cgi> |
| **Tomcat** | [http://tomcat.apache.org/download](http://tomcat.apache.org/download-60.cgi) |
| **Mod\_jk-1.2.x** | <http://tomcat.apache.org/download-connectors.cgi> |

**2.   安装apache 和 tomcat**

将apache安装到硬盘上， 然后将2个tomcat解压。（这里的安装和解压不限定具体目录，安装和解压到某个目录下即可）。

这里我将apache2.2安装到E:\Apache\_Server。

2个tomcat也解压到同一个目录下。



接着检查apache是否能正常启动。



**3.   对apache 配置负载均衡**

**（1）取出下载的tomcat-connectors-1.2.37-windows-i386-httpd-2.2.x下的 mod\_jk.so。**

**将其放置到Apache2.2\modules目录下。**

**（2）打开Apache2.2\conf\ httpd.conf文件。在其末尾添加上一行：**

#Tomcat conf

include E:\Apache\_Server\Apache2.2\conf\mod\_jk.conf

即调用conf目录下的mod\_jk.conf这个配置文件。

当然，apache conf目录下原来是没有这个文件的。我们需要自己新建这个文件。

**（3）在conf目录下新建Mod\_jk.conf内容如下：**

|  |
| --- |
| #加载mod\_jk Module  LoadModule jk\_module modules/mod\_jk.so  #指定 workers.properties文件路径  JkWorkersFile conf/workers.properties  #指定那些请求交给tomcat处理,"controller"为在workers.propertise里指定的负载分配控制器  JkMount /\* controller |

第2行中的mod\_jk.so，就是我们之前放到modules目录下的那个.so文件。这里的名字就根据放进去的实际名字来写。版本不同，名字也就不一样。当然，mod\_jk.so这个文件允许自己重命名。重命名后，只要在这里相应的配置上重命名后的名字，不会影响这个插件的功能。

第4行中conf/workers.properties文件也是不存在的，也需要我们自己建立。这个文件的作用是对用于均衡负载的 负载器进行具体的登记。（比如我们用的2个tomcat）.换句话说，workers.properties就是对几个worker的登记表。我们的2个tomcat就作为2个worker被登记在这个文件中。而具体要如何进行登记，下面再具体说。

第6行的作用就是配置apache对哪些请求进行转发。这里我按照网络上的教程，配置了对所有action，jsp， html页面的请求的转发。

若配置成“JkMount /jsp controller”这样，那么，当有jsp请求到达apache时，apache就会根据当时的负载情况，从worker中挑出负载低的那个worker，将这个jsp请求转发给他。（这里需要注意，当部署到tomcat上的项目有更多类型的请求的时候，需要在这里配置更多的请求规则）。而controller会在workers.peroperties中被配置。总的来说，转发规则的格式如下：

JkMount + 过滤请求类型 + 处理器

**（4）接着，我们在conf目录下新建workers.properties内容如下：**

|  |
| --- |
| **#server 列表**  **worker.list = controller,tomcat1,tomcat2**  **#========tomcat1========**  **#ajp13 端口号，在tomcat下server.xml配置,默认8009**  **worker.tomcat1.port=18009**  **#tomcat的主机地址，如不为本机，请填写ip地址**  **worker.tomcat1.host=localhost**  **#定向包协议**  **worker.tomcat1.type=ajp13**  **#server的加权比重，值越高，分得的请求越多**  **worker.tomcat1.lbfactor = 1**  **worker.tomcat1.redirect = tomcat2**  **#========tomcat2========**  **#ajp13 端口号，在tomcat下server.xml配置,默认8009**  **worker.tomcat2.port=28009**  **#tomcat的主机地址，如不为本机，请填写ip地址**  **worker.tomcat2.host=localhost**  **#定向包协议**  **worker.tomcat2.type=ajp13**  **#server的加权比重，值越高，分得的请求越多**  **worker.tomcat2.lbfactor = 1**  **worker.tomcat2.redirect = tomcat1**  **#对某一个tomcat的活动性进行禁用负载均衡启动的时候，其实负载都到tomcat1上面去。Tomcat2默认不参与负载。**  **#当tomcat1当掉的时候，通过worker.tomcat1.redirect = tomcat2 ,tomcat2就会自己启动起来，接替tomcat1继续工作。**  **#当然，这样就和负载均衡没有关系了。**  **#worker.tomcat2.activation = disabled**  **#========controller,负载均衡控制器========**  **worker.controller.type=lb**  **#指定分担请求的tomcat**  **worker.controller.balanced\_workers=tomcat1,tomcat2**  **worker.controller.sticky\_session=1** |

第一行，worker.list，列出了worker的列表。Tomcat1, tomcat2就是我们为2个tomcat的命名。而controller相当于包工头。他负责对2个（或者更多）的worker进行负载分配。在上面提到的Mod\_jk.conf文件中配置的对各式各样请求的处理器就是controller.在这里再由controller具体将任务分配给某一个tomcat.

下面几行就是分别对worker的配置。Port, host等属性都需要根据tomcat的实际配置来填。

实际上，这里对tomcat还有一个属性可以进行配置，即worker.tomcat2.activation = disabled 。就是对某一个tomcat的活动性进行禁用。这句配置的意思是，负载均衡启动的时候，其实负载都到tomcat1上面去。Tomcat2默认不参与负载。当tomcat1当掉的时候，通过worker.tomcat1.redirect = tomcat2 ,tomcat2就会自己启动起来，接替tomcat1继续工作。当然，这样就和负载均衡没有关系了。

**（5）到此，对apache的配置基本告一段落。**

如果集群中有更多的tomcat的话，请在worker.properties中进行配置。（注：当有6个tomcat进行集群的时候，为了提高性能， worker.properties中的 redirect和activation就可以起到作用。我们可以对tomcat两两配对，两两互相设置redirect，两个其中一个设置activation. 形成3对，这样，就可以处理意外宕机的情况，保证6台集群可以继续运行下去。）

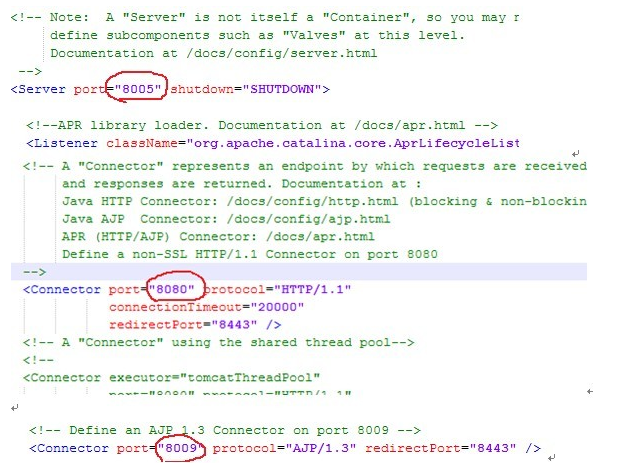
**4.   Tomcat集群配置**

将2个tomcat分别解压后，将文件夹重新命名为tomcat1， tomcat2

打开tomcat1 \conf\server.xml ， tomcat2 \conf\server.xml文件。

**(1)  修改端口：**

2个tomcat, 为了避免端口冲突，其中一个端口使用默认的，而另一个得修改（当然可以两个都同时更改， 而且还要与3.4步骤中的端口一一对应）。必须修改的端口如下：



**(2)  修改集群设置：**

在<Engine>或<Host>元素下添加以下内容均可。

按照apache tomcat官方文档上面的说法，对于tomcat6要做集群的话，只需要将<Engine>元素下的

<!--

  <Cluster className="org.apache.catalina.ha.tcp.SimpleTcpCluster"/>

-->

的注释符号去掉，启用这句配制就可以正常使用集群了。

不过由于我搭建的测试环境2个tomcat是在同一台机子上面，因此只启用这个<Cluster>的话，这个元素下的默认Receiver port就会冲突。因此必须补全这些配置。

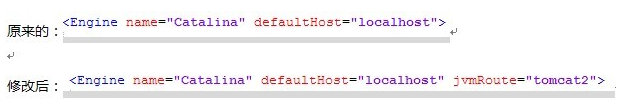
另外，按照官方文档里说明的<Cluster>的默认配置，<Membership>元素下的address属性默认值为228.0.0.4 .

需要而外注意的是，2个tomcat里添加到上面这段<Cluster>配置，其<Receiver>下的Port元素必需配置成不同的。如其中一个是4001，一个是4002.（tomcat默认可以检测到4000～4100之间的端口）。

**具体配置如下：**

|  |
| --- |
| **<Cluster className="org.apache.catalina.ha.tcp.SimpleTcpCluster">**  **<Channel className="org.apache.catalina.tribes.group.GroupChannel">**  **<Membership className="org.apache.catalina.tribes.membership.McastService"**  **bind="127.0.0.1"**  **address="228.0.0.4"**  **port="45564"**  **frequency="500"**  **dropTime="3000"/>**  **<Receiver className="org.apache.catalina.tribes.transport.nio.NioReceiver"**  **address="127.0.0.1"**  **port="4001"**  **autoBind="100"**  **selectorTimeout="5000"**  **maxThreads="6"/>**    **<Sender className="org.apache.catalina.tribes.transport.ReplicationTransmitter">**  **<Transport className="org.apache.catalina.tribes.transport.nio.PooledParallelSender"/>**  **</Sender>**    **<Interceptor className="org.apache.catalina.tribes.group.interceptors.TcpFailureDetector"/>**  **<Interceptor className="org.apache.catalina.tribes.group.interceptors.MessageDispatch15Interceptor"/>**  **</Channel>**  **<Valve className="org.apache.catalina.ha.tcp.ReplicationValve" filter=""/>**  **<Valve className="org.apache.catalina.ha.session.JvmRouteBinderValve"/>**    **<Deployer className="org.apache.catalina.ha.deploy.FarmWarDeployer"**  **tempDir="/tmp/war-temp/"**  **deployDir="/tmp/war-deploy/"**  **watchDir="/tmp/war-listen/"**  **watchEnabled="false"/>**    **<ClusterListener className="org.apache.catalina.ha.session.JvmRouteSessionIDBinderListener"/>**  **<ClusterListener className="org.apache.catalina.ha.session.ClusterSessionListener"/>**  **</Cluster>** |

**（3）修改<Engine>的属性。**



2个tomcat, jvmRoute分别配置成tomcat1和tomcat2,即和apache/conf里worker.properites配置文件中配置的worker名称对应。

既是tomcat1配置的 jvmRoute=tomcat2； tomcat2的配置正好相反。

**5.   应用 集群配置：**

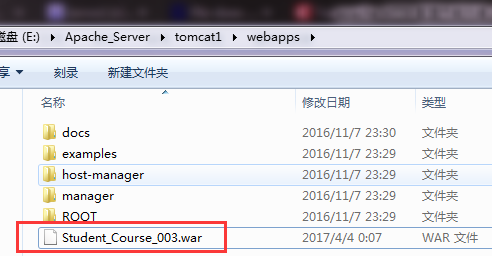
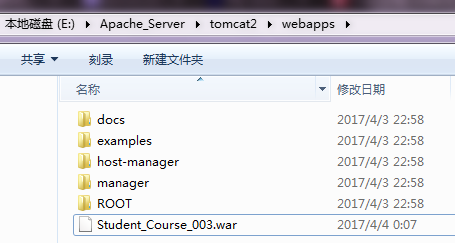
**（1）    修改应用的web.xml,添加  <Context distributable="true" />**

****

所有需要集群的web项目，其web.xml中都必须添加<Context distributable=*"true"* /> 这个定义

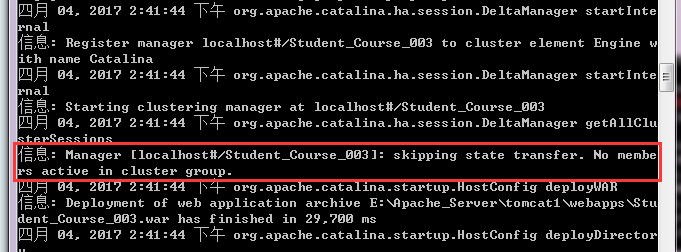
**（2）    将应用分别部署到2个tomcat中：**

直接把项目放到tomcat/webapp目录下：

**（3）    分别启动两个tomcat**

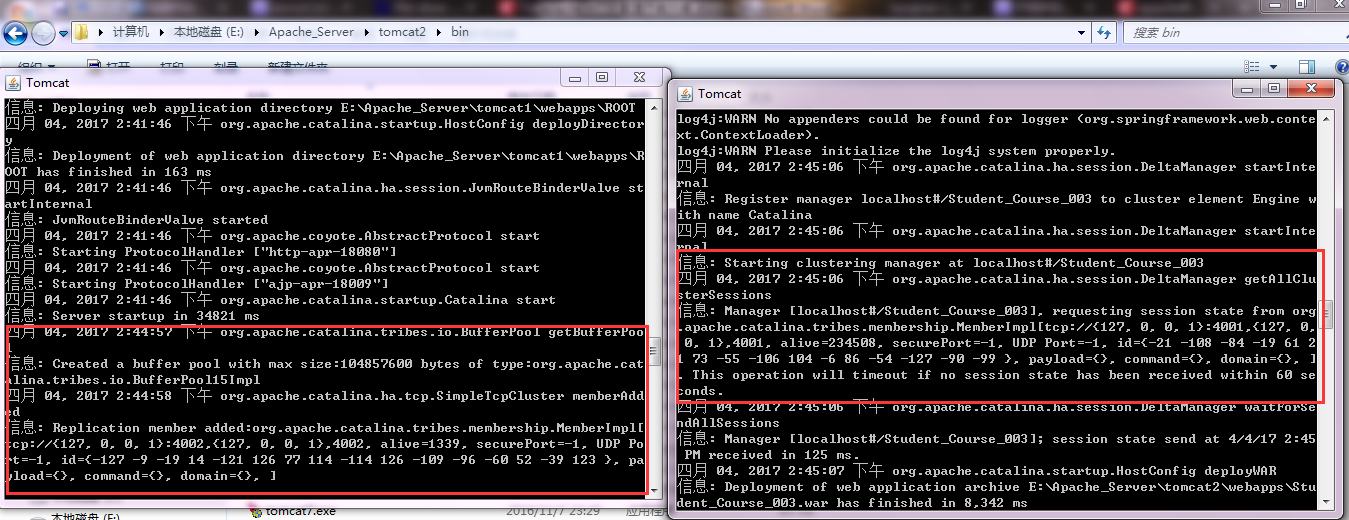
启动第一个tomcat1,可以看到如下启动日志：



说明当前集群只启动了1个tomcat， 所以没有集群的说法。

启动tomcat2:

启动过程和tomcat1一样，值得注意的是，在tomcat2启动的时候，tomcat1会打印出集群中加入member的提示，左边卫tomcat1，右边为tomcat2：



（4）    **启动apache.**

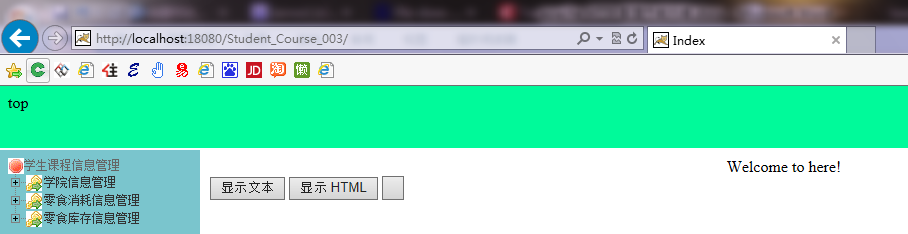
通过IP访问apache. 比如我的是  <http://127.0.0.1> (这里80端口写不写是一样的)。

若一开始配置的Mod\_jk.conf文件中只配置了JSP的话，你会发现可能应用都登陆不了，或者登陆了页面都无法正常打开。文件末尾的请求转发。我们只转发了jsp页面的请求，就是说apache只会把jsp页面的请求转发到2个tomcat上去。因此我们需要修改过滤的规则。我没有找到资料具体的各种规则应该怎么写，只能写一个大的过滤：

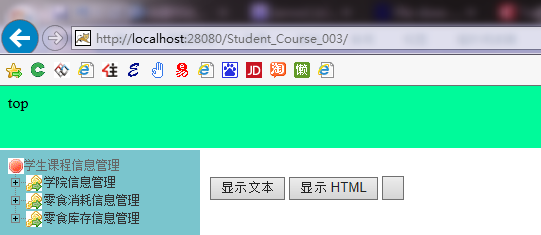
JkMount /\* controller

将mod\_jk.conf末尾改成上面这样。重启apache，你就可以看到应用登陆界面了：

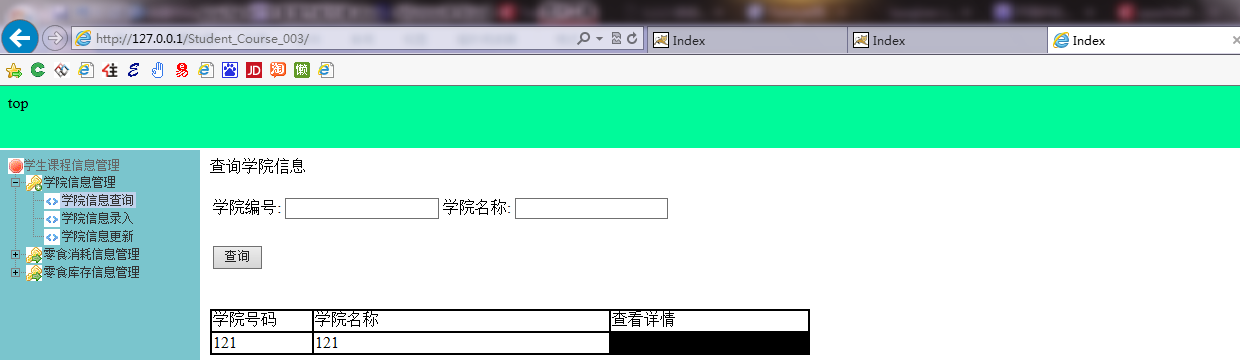
Tomcat1:



Tomcat2:

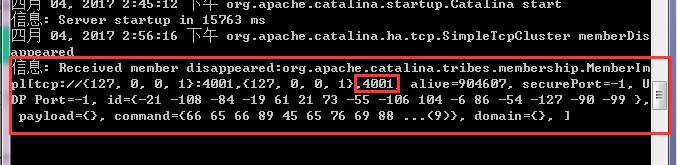


Apache server 80端口:



重复多开几个session，可以看到日志在两个tomcat的后台里打印

直接把tomcat1停止掉。可以在tomcat2的控制台中看到关于集群成员关闭的信息：



验证方法如下：

|  |
| --- |
| 1. 重复多开几个session访问应用，查看日志打印 |
| 1. 关闭其中一台 ，查看另外一台日志 |
| 1. 倘若在tomcat1打印日志， 说明客户访问的服务器是tomcat1， 此时把tomcat1关闭，   再做一些CRUD的操作，看应用是否正常运行，同时tomcat2的日志是否打印。 |

以上3点均可验证出集群是否配置成功.

-----至此， 负载均衡配置完成------

apache外网不能访问分析与解决方法

apache安装好以后，在本机可以用：http://localhost 或者http://127.0.0.1进行访问，但是，在外网（相对本机来说的，**局域网**也算外网）不能访问。

这种情况可以分为两个问题，第一个是在局域网内，第二国是在互联网；

有三种情况可能造成这种现象，

1、配置文件没有设置外网访问；

2、防火墙，计算机本身的防火墙禁止了外网通过80端口（默认的，也可以自定义别的）访问；

3、80端口没有对外网开放，或者局域网内，或者互联网（外网防火墙或者其他运帷软件）；

如果你配置的服务器是个远程主机，并且有可能是个虚拟主机，这时一定要搞清楚，是否对外开放了80端口（默认的，可以自定义），或者有防火墙限制了访问，如果有，可能你通过ip链接站点是会出现意想不到的页面，这是因为管理员通过管理软件将你的链接给转移了，在到大你想要的站点之前就给转移了，我以前见过一个就是这样的，明明我的站点是apache，远程访问时却显示nginx服务器页面不存在，我通过对方给的远程链接ip分析（给的ip带有一个特殊的端口号），可能在服务器的外围还有一个管理软件，限制了远程访问，果不其然，通知对方后开放限制后，apache能正常访问。

上面三种问题的解决方法：

1、修改配置文件，apache配置文件httpd.conf的默认配置中有个：Allow from127.0.0.1，将其修改为：Allow from all 即可；

2、本身防火墙造成的，此时你可以先试试关闭防火墙，看是否能访问，确定问题是否在这里，如果是，就需要配置一下防火墙，如下：

控制面板打开防火墙界面－>点击左上角的“允许程序或功能通过Windows防火墙”

看看Apache HTTPServer是否选择了公网，将其选择，尝试外网链接，如果还不行，继续执行下面操作－>

点击右下角“允许运行另一程序”－>然后点击“浏览”－>找到httpd.exe程序“C:\wamp\bin\apache\apache2.2.22\bin\httpd.exe”添加即可；

3、如何知道80端口已经开放，而被外围网络管理软件阻止了呢？

答：在本机和外网分别执行端口扫描（http://tool.chinaz.com/port/），内网开着80,而外网显示80未开放，说明又管理软件阻止了80端口信息传输；这个问题只能由管理员来设置了。