

Nginx+tomcat+redis实现负载均衡以及session同步

1.环境

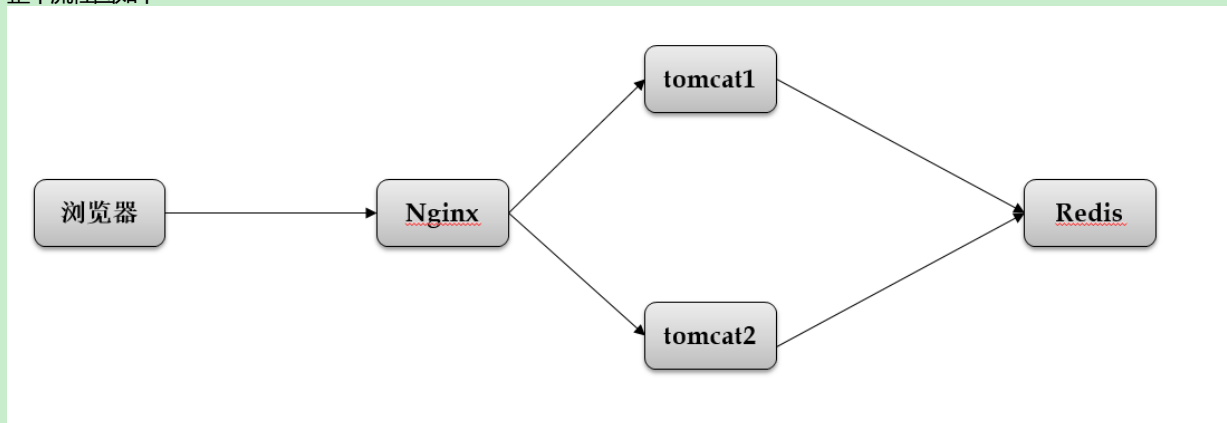
一台Nginx服务器，二台tomcat服务器，一台redis服务器。

Nginx服务器用于接受来自用户的连接请求，然后将请求报文按照一定的规则，分发到后台应用服务器tomcat上。

Tomcat服务器，真正处理用户请求的地方。

Redis服务器，用于存储session信息的地方。

整个流程图如下



1.浏览器访问Nginx服务器

2.Nginx服务器接收到请求之后将请求分发给后台的Tomcat服务器

3.tomcat服务器如果要操作session，则访问redis服务器

2.配置

1.修改tomcat1和tomcat2的监听端口，使得他们的监听端口不冲突。

2.拷贝相同的项目到tomcat1和tomcat2的webapps目录下，在tomcat1下修改项目里面的test.jsp文件为**这是在tomcat1**，tomcat2的test.jsp文件中修改为**这是在tomcat2**。

3.配置Nginx服务器。主要修改的是如下几个地方

```
upstream local tomcat {  
    #针对不同ip的用户请求分配固定的tomcat响应其请求。  
    #ip hash;  
    #配置tomcat服务器的ip: 端口, 处理请求权重  
    server localhost:8080 weight=5;  
    server localhost:8082 weight=5;  
}  
  
server {  
    listen      80;  
    server_name localhost;  
  
    #charset koi8-r;  
  
    #access_log logs/host.access.log main;  
  
    location / {  
        #root    html;  
        #index  index.html index.htm;  
        proxy_connect_timeout      1;  
        proxy_read_timeout         1;  
        proxy_send_timeout         1;  
        proxy_pass http://local_tomcat;
```

这样Nginx就可以代理2个Tomcat服务器了。

3.测试

在地址栏分别访问二个tomcat，响应结果如下

<http://localhost:8080/TestNginx/jsp/test.jsp>

← → ↻ localhost:8080/TestNginx/jsp/test.jsp

这是在tomcat1

<http://localhost:8082/TestNginx/jsp/test.jsp>

← → ↻ localhost:8082/TestNginx/jsp/test.jsp

这是在tomcat2

从而知道二个tomcat都是运行正常的。

这个时候我们访问Nginx服务器。

<http://localhost/TestNginx/jsp/test.jsp>

多次访问相同的地址之后，我们发现响应的结果是在tomcat1和tomcat2之间来回切换，结果如下

← → ↻ localhost/TestNginx/jsp/test.jsp

这是在tomcat1

← → ↻ localhost/TestNginx/jsp/test.jsp

这是在tomcat2

从而证明Nginx实现了二台tomcat的负载均衡

这个时候，我们再来做一件有趣的事情，我们shutdown tomcat2，我们再来访问Nginx服务器来看看是个什么效果。

我们发现页面始终显示的是tomcat1

← → ✕ localhost/TestNginx/jsp/test.jsp

这是在tomcat1

从而证明Nginx还可以有效预防单点故障。

4.session问题

这个时候我们再来做一件有趣的事情，我们修改tomcat里面的项目test.jsp页面，输出session信息。

```
<body>
  这是在tomcat2<br/>
  session信息:<%=session.getId()%>
</body>
```

我们再来访问Nginx服务器试试。

← → ↻ localhost/TestNginx/jsp/test.jsp

这是在tomcat1
session信息:CA46917951C8533D9B882BC72943909E

← → ↻ localhost/TestNginx/jsp/test.jsp

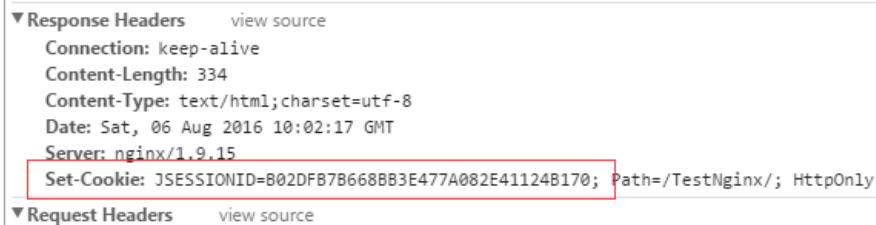
这是在tomcat2
session信息:0AFAE94158FD80DD40627122B52FFBC0

擦，session信息不一样，搞毛线啊！不行要解决这个问题。

我们现在来思考这样一个场景，就是我们现在登录一个网站，nginx给我们分配的是tomcat1服务器，然后我们的登录信息是存储在tomcat1服务器上面，登录成功之后，我们点击页面上一个链接，然后nginx服务器把我们的请求分配给tomcat2.这个时候问题就来了，tomcat2知道我们的登录信息吗？显然是不知道，这个时候问题就出现了，我们的操作就失效了。擦，那么有没有什么方法可以解决这种问题呢！方法是有的，就是使用redis服务器。

redis就是一个大大的Map，在内存中维护的。

session：session代表的是一次用户的会话。那么服务器是如何跟踪用户的会话呢！



▼ Response Headers view source

Connection: keep-alive
Content-Length: 334
Content-Type: text/html; charset=utf-8
Date: Sat, 06 Aug 2016 10:02:17 GMT
Server: nginx/1.9.15
Set-Cookie: JSESSIONID=B02DFB7B668B83E477A082E41124B170; Path=/TestNginx/; HttpOnly

我们可以看见，是服务器向我们的客户端回写了cookie。key是JSESSIONID
然后我们在请求的时候，我们的请求头里面包含了，服务器写给我们的JSESSIONID



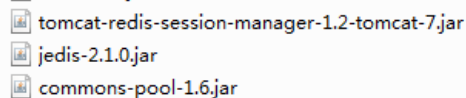
▼ Request Headers view source

Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Encoding: gzip, deflate, sdch
Accept-Language: zh-CN,zh;q=0.8
Cache-Control: max-age=0
Connection: keep-alive
Cookie: JSESSIONID=B02DFB7B668B83E477A082E41124B170
Host: localhost
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/46.0.2490.86 Safari/537.36

这样，服务器就知道，哦这个是刚才登录那个。就可以跟踪用户的会话了。

5.使用redis来共享session

1.在tomcat的lib文件夹里面添加三个jar包分别是



tomcat-redis-session-manager-1.2-tomcat-7.jar
jedis-2.1.0.jar
commons-pool-1.6.jar

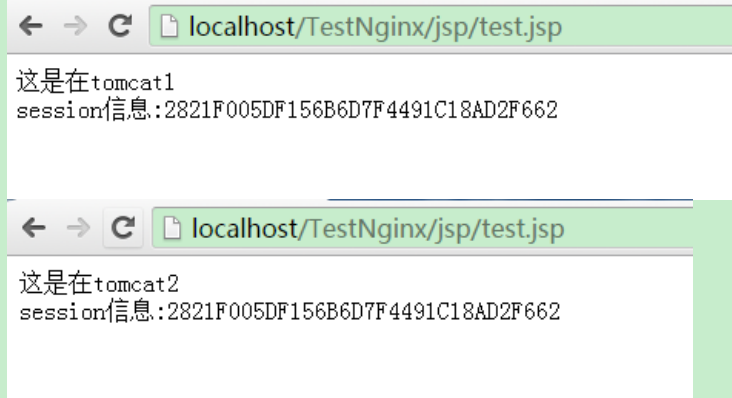
主要是用于和Redis通信相关jar包

2.修改tomcat下面的context.xml配置文件。在<Context>节点下增加如下内容

```
<Valve className="com.radiadesign.catalina.session.RedisSessionHandlerValve" />  
<Manager className="com.radiadesign.catalina.session.RedisSessionManager" host="127.0.0.1" port="6379" database="0" maxInactiveInterval="60"/>
```

主要是配置session管理器为redis，然后是redis服务器的地址，端口等信息。

这个时候我们重启tomcat服务器，启动redis服务器，再来访问Nginx服务来试试。（好紧张。。）



localhost/TestNginx/jsp/test.jsp

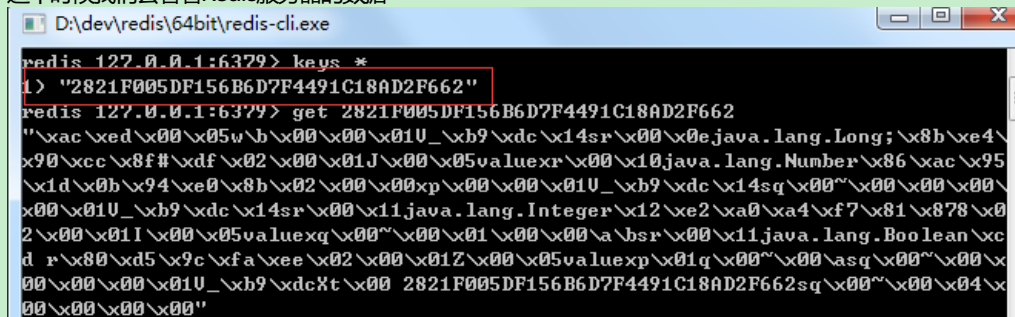
这是在tomcat1
session信息:2821F005DF156B6D7F4491C18AD2F662

localhost/TestNginx/jsp/test.jsp

这是在tomcat2
session信息:2821F005DF156B6D7F4491C18AD2F662

哈哈 二台tomcat服务器返回的session信息是一致的，yea。达到了session共享。完美解决。

这个时候我们去看看Redis服务器的数据



D:\dev\redis\64bit\redis-cli.exe

```
redis 127.0.0.1:6379> keys *  
1> "2821F005DF156B6D7F4491C18AD2F662"  
redis 127.0.0.1:6379> get 2821F005DF156B6D7F4491C18AD2F662  
"\xac\xed\x00\x05w\b\x00\x00\x01U_\xb9\xdc\x14sr\x00\x0ejava.lang.Long;\x8b\xe4\x90\xce\x8f#\xdf\x02\x00\x01J\x00\x05valuexr\x00\x10java.lang.Number\x86\xac\x95\x1d\x0b\x94\xe0\x8b\x02\x00\x00xp\x00\x00\x01U_\xb9\xdc\x14sq\x00~\x00\x00\x00\x00\x01U_\xb9\xdc\x14sr\x00\x11java.lang.Integer\x12\xe2\xa0\xa4\xf7\x81\x878\x02\x00\x01I_\x00\x05valuexq\x00~\x00\x01\x00\x00a\bsr\x00\x11java.lang.Boolean\xcd r\x80\xd5\x9c\xfa\xee\x02\x00\x01Z\x00\x05valuexp\x01q\x00~\x00\asq\x00~\x00\x00\x00\x01U_\xb9\xdc\x14st\x00 2821F005DF156B6D7F4491C18AD2F662sq\x00~\x00\x04\x00\x00\x00\x00"
```

我们看见 redis里面有个key就是我们的sessionid，至此负载均衡以及session共享完美解决

