

Nginx+Tomcat搭建高性能负载均衡集群

Nginx+Tomcat搭建高性能负载均衡集群

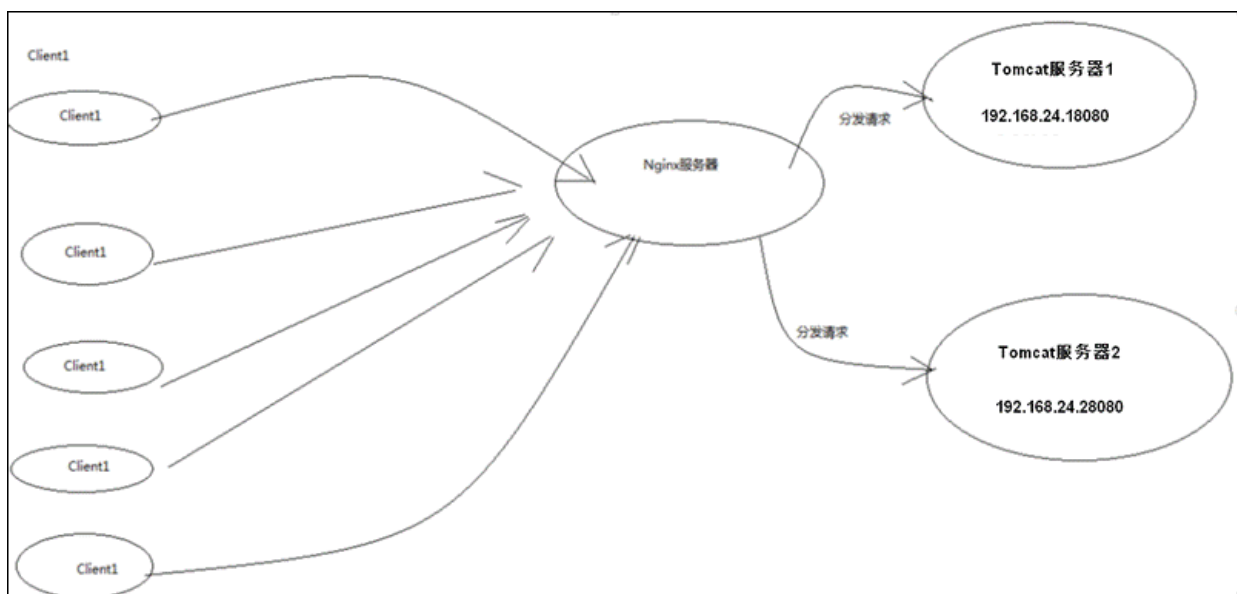
一、工具

nginx-1.8.0

apache-tomcat-6.0.33

二、目标

实现高性能负载均衡的Tomcat集群：



三、步骤

1、首先下载Nginx，要下载稳定版：

E:\servers\nginx-1.8.0				
名称	修改日期	类型	大小	
conf	2015/8/14 10:42	文件夹		
contrib	2015/4/21 17:13	文件夹		
docs	2015/4/21 17:13	文件夹		
html	2015/4/21 17:13	文件夹		
logs	2015/8/18 21:57	文件夹		
temp	2015/8/14 10:43	文件夹		
nginx.exe	2015/4/21 16:44	应用程序	2,691 KB	
zhipeieng---nginx 常用命令.txt	2015/8/14 11:31	文本文档	3 KB	

2、然后解压两个Tomcat，分别命名为apache-tomcat-6.0.33-1和apache-tomcat-6.0.33-2：

E:\servers			
名称	修改日期	类型	大小
apache-tomcat-6.0.33	2015/8/17 9:13	文件夹	
apache-tomcat-6.0.33-1	2015/8/14 10:11	文件夹	
apache-tomcat-6.0.33-2	2015/8/14 10:11	文件夹	
nginx-1.8.0	2015/8/14 11:29	文件夹	
nginx-1.8.0.zip	2015/8/14 9:45	WinRAR ZIP 压缩...	1,252 KB
tomcat1.bat	2015/8/14 10:30	快捷方式	2 KB
tomcat2.bat	2015/8/14 10:29	快捷方式	2 KB

3、然后修改这两个Tomcat的启动端口，分别为18080和28080，下面以修改第一台Tomcat为例，打开Tomcat的conf目录下的server.xml：

E:\servers\apache-tomcat-6.0.33-1\conf			
名称	修改日期	类型	大小
Catalina	2015/8/14 10:12	文件夹	
catalina.policy	2011/8/16 20:25	POLICY 文件	11 KB
catalina.properties	2011/8/16 20:25	PROPERTIES 文件	4 KB
context.xml	2011/8/16 20:25	XML 文件	2 KB
logging.properties	2011/8/16 20:25	PROPERTIES 文件	4 KB
server.xml	2015/8/14 10:19	XML 文件	7 KB
tomcat-users.xml	2011/8/16 20:25	XML 文件	2 KB
web.xml	2011/8/16 20:25	XML 文件	51 KB

共需修改3处端口：

<pre><Connector port="18080" protocol="HTTP/1.1" connectionTimeout="20000" redirectPort="8443" /></pre>
<pre><Connector port="18080" protocol="HTTP/1.1" connectionTimeout="20000" redirectPort="8443" /></pre>
<pre><!-- Define an AJP 1.3 Connector on port 8009 --> <Connector port="18009" protocol="AJP/1.3" redirectPort="8443" /></pre>

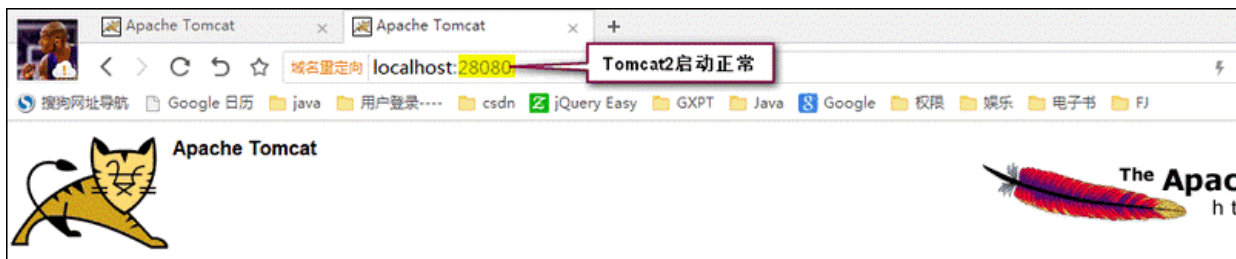
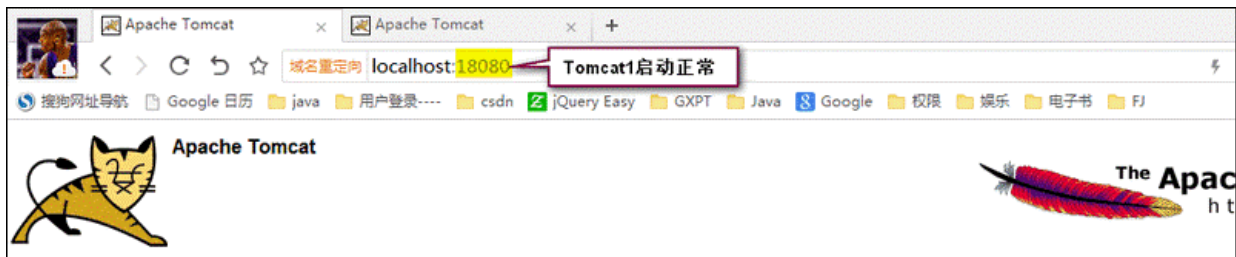
当然第二台Tomcat也一样，如下图：

```
<Server port="28005" shutdown="SHUTDOWN">
```

```
<Connector port="28080" protocol="HTTP/1.1"  
connectionTimeout="20000"  
redirectPort="8443" />
```

```
<Connector port="28009" protocol="AJP/1.3" redirectPort="8443" />
```

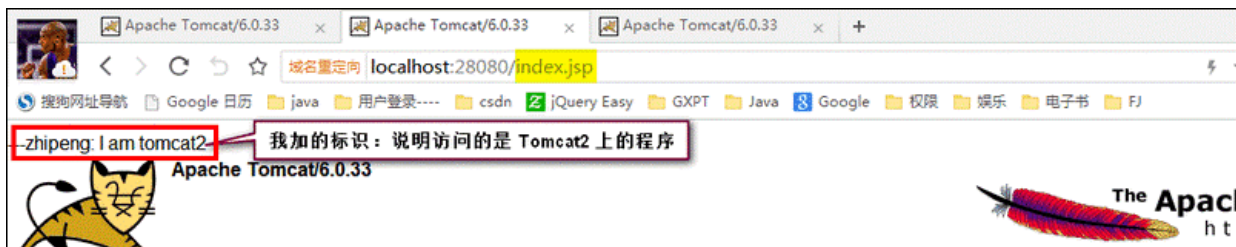
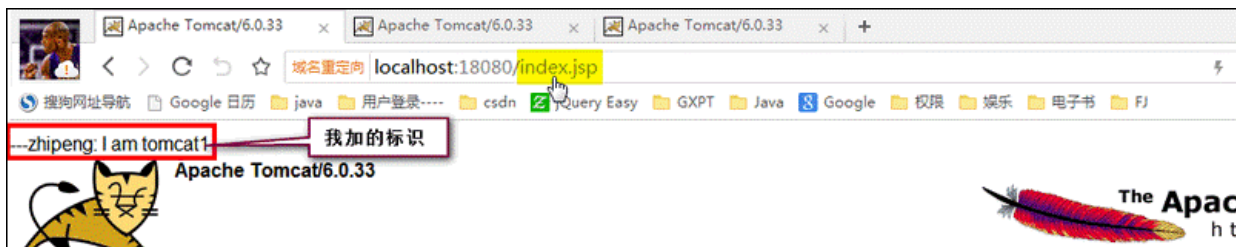
4、然后启动两个Tomcat，并访问，看是否正常：



5、然后修改上面两个Tomcat的默认页面（为了区分下面到底访问的是那一台Tomcat，随便改一下即可）：

E:\servers\apache-tomcat-6.0.33-1\webapps\ROOT				
名称	修改日期	类型	大小	
WEB-INF	2015/8/14 10:11	文件夹		
asf-logo-wide.gif	2011/8/16 20:25	GIF 文件	6 KB	
build.xml	2011/8/16 20:25	XML 文件	4 KB	
favicon.ico	2011/8/16 20:25	图标	22 KB	
index.html	2011/8/16 20:25	360 Chrome HT...	8 KB	
index.jsp	2015/8/14 11:25	JSP 文件	9 KB	
RELEASE-NOTES.txt	2011/8/16 20:25	文本文档	9 KB	
tomcat.gif	2011/8/16 20:25	GIF 文件	2 KB	

改完以后，进行访问，如下图：



6、OK，现在我们可以开始配置Nginx来实现负载均衡了，其实非常的简单，只需要配置好Nginx的配置文件即可：

E:\servers\nginx-1.8.0\conf				
名称	修改日期	类型	大小	
fastcgi.conf	2015/4/21 17:13	CONF 文件	2 KB	
fastcgi_params	2015/4/21 17:13	文件	1 KB	
koi-utf	2015/4/21 17:13	文件	3 KB	
koi-win	2015/4/21 17:13	文件	3 KB	
mime.types	2015/4/21 17:13	TYPES 文件	4 KB	
nginx.conf	2015/8/19 7:41	CONF 文件	4 KB	
scgi_params	2015/4/21 17:13	文件	1 KB	
uwsgi_params	2015/4/21 17:13	文件	1 KB	
win-utf	2015/4/21 17:13	文件	4 KB	

配置如下（这里只进行了简单的配置，实际生产环境可以进行更详细完善配置）：

[html] [view plain copy](#)

```

1. worker_processes 1;#工作进程的个数，一般与计算机的cpu核数一致
2.
3. events {
4.     worker_connections 1024;#单个进程最大连接数（最大连接数=连接数*进程数）
5. }
6.
7. http {
8.     include mime.types; #文件扩展名与文件类型映射表
9.     default_type application/octet-stream;#默认文件类型
10.
11.     sendfile on;#开启高效文件传输模式，sendfile指令指定nginx是否调用sendfile函数来输出文件，对于普通应用设为 on，如果用来进行下载等应用磁盘IO重负载应用，可设置为off，以平衡磁盘与网络I/O处理速度，降低系统的负载。注意：如果图片显示不正常把这个改成off。
12.
13.     keepalive_timeout 65; #长连接超时时间，单位是秒
14.
15.     gzip on;#启用Gzip压缩
16.
17.     #服务器的集群
18.     upstream netitcast.com { #服务器集群名字
19.         server 127.0.0.1:18080 weight=1;#服务器配置 weight是权重的意思，权重越大，分配的概率越大。

```

```

20.     server    127.0.0.1:28080  weight=2;
21.     }
22.
23.     #当前的Nginx的配置
24.     server {
25.         listen      80;#监听80端口，可以改成其他端口
26.         server_name localhost;##### 当前服务的域名
27.
28.         location / {
29.             proxy_pass http://netitcast.com;
30.             proxy_redirect default;
31.         }
32.
33.
34.         error_page   500 502 503 504 /50x.html;
35.         location = /50x.html {
36.             root    html;
37.         }
38.     }
39. }

```

核心配置如下：

The diagram illustrates the Nginx configuration for an upstream cluster and a server. It includes the following annotations:

- 服务器的集群** (Server Cluster): Points to the `upstream` block.
- 集群的服务器列表，IIS，最终请求会被转发到这里执行** (Cluster server list, IIS, final request will be forwarded here for execution): Points to the `server` lines within the `upstream` block.
- 当前的Nginx的配置** (Current Nginx configuration): Points to the `server` block.
- 如果请求为localhost80，则交给名称为netitcast.com的Nginx集群来处理** (If the request is for localhost80, it will be handled by the Nginx cluster named netitcast.com): Points to the `server_name` and `listen` lines.
- 要与3中配置的Nginx的名称一致** (Must be consistent with the Nginx name configured in 3): Points to the `proxy_pass` line.

```

#服务器的集群
upstream netitcast.com { #服务器集群名字
    #server 172.16.21.13:8081 weight=1;#服务器配置 weight是权重的意思，权重越大，分配的概率越大。
    server 127.0.0.1:18080 weight=1;
    server 127.0.0.1:28080 weight=2;
}

#当前的Nginx的配置
server {
    listen 80;#监听80端口，可以改成其他端口
    server_name localhost;##### 当前服务的域名

    #charset koi8-r;

    #access_log logs/host.access.log main;

    #location / {
    #    root html;
    #    index index.html index.htm;
    #}

    location / {
        proxy_pass http://netitcast.com;
        proxy_redirect default;
    }
}

```

到此配置完成，下面开始演示负载均衡。

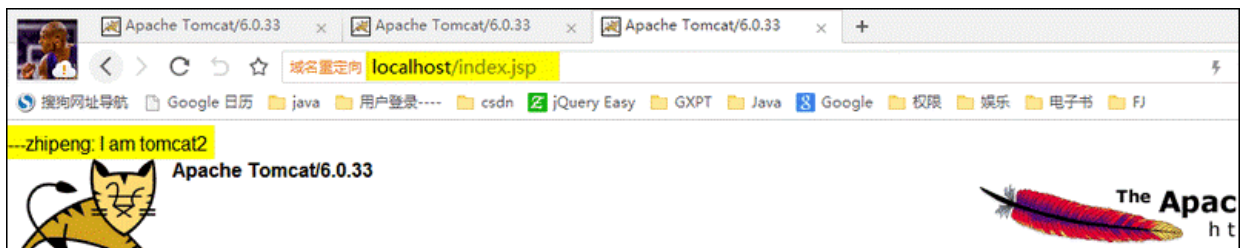
7、首先，我们启动Nginx：（直接输入nginx）

The screenshot shows the following steps in a Windows command prompt:

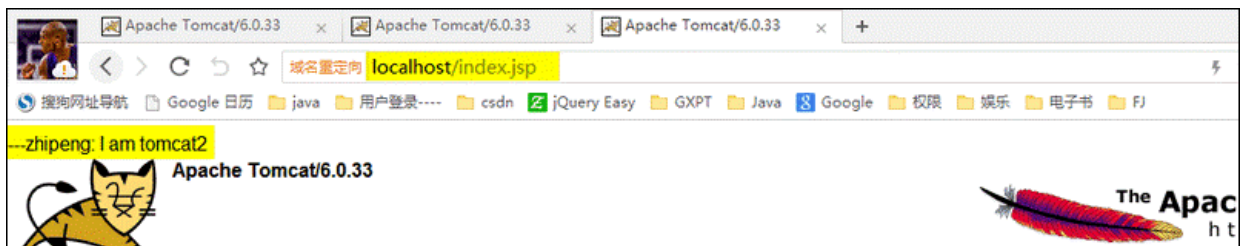
- 1、进入到Nginx的目录 (Enter the Nginx directory): `cd E:\servers\nginx-1.8.0`
- 2、启动Nginx (Start Nginx): `start nginx`

8、然后我们即可输入：localhost/index.jsp查看运行状况了

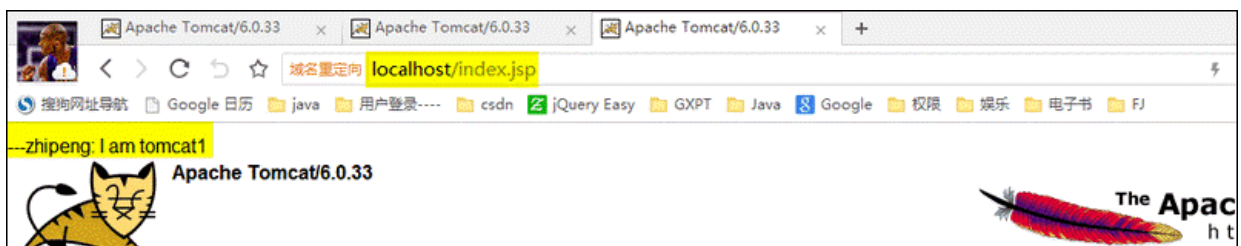
第一次访问，发现访问的是Tomcat2上的程序：



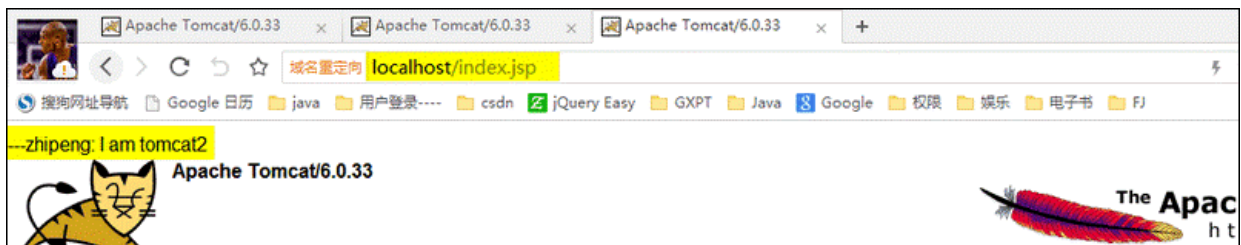
然后刷新，访问的还是Tomcat2上的程序：



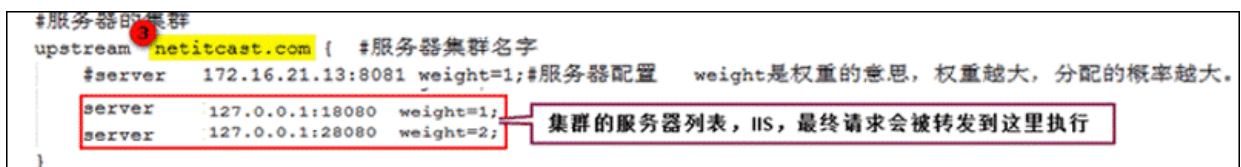
再刷新，发现变为了Tomcat1上的程序：



再刷新，发现又变为了Tomcat2上的程序：



到此，我们利用Nginx已经实现了负载均衡的Tomcat集群。我们不断的刷新，发现访问Tomcat2的概率大概是Tomcat1的2倍，这是因为我们在Nginx中配置的两台Tomcat的权重起的作用，如下图：



四、总结

谁能想到实现一个高性能的负载均衡集群会如此简单。Nginx的功能如此强大，配置却如此简单，我们还有什么理由拒绝它呢？这比我们动不动就十多万至几十万人民币的F5 BIG-IP、NetScaler等硬件负载均衡交换机廉价了不知多少。此外，大家别忘了Nginx不仅仅是一个反向代理服务器，它本身也可以托管网站，作为Web服务器，进行Http服务处理。