**E 基于nginx的tomcat负载均衡跟集群(超简单)**

基于nginx的tomcat负载均衡和集群(超简单)

今天看到"**基于apache的tomcat负载均衡和集群配置**"这篇文章成为javaEye热点。

略看了一下，感觉太复杂，要配置的东西太多，因此在这里写出一种更简洁的方法。

要集群tomcat主要是解决SESSION共享的问题，因此我利用memcached来保存session，多台TOMCAT服务器即可共享SESSION了。

你可以自己写tomcat的扩展来保存SESSION到memcached。

这里推荐使用memcached-session-manager这个开源项目（http://code.google.com/p/memcached-session-manager/ ），下面简称msm。

如何安装nginx、memcached、tomcat这些就不多说了。

先说明一下测试环境:

tomcat1、nginx、memcached安装在192.168.1.11

tomcat2安装在192.168.1.101

下面分步实现基于nginx的tomcat负载均衡和集群配置

**一，tomcat集群**

    1，先下载msm及其依赖包

    http://memcached-session-manager.googlecode.com/files/memcached-session-manager-1.3.0.jar

    http://memcached-session-manager.googlecode.com/files/msm-javolution-serializer-jodatime-1.3.0.jar

http://memcached-session-manager.googlecode.com/files/msm-javolution-serializer-cglib-1.3.0.jar

http://spymemcached.googlecode.com/files/memcached-2.4.2.jar

http://memcached-session-manager.googlecode.com/files/javolution-5.4.3.1.jar

2，将这5个包放到$TOMCAT\_HOME/lib目录下

3，修改$TOMCAT\_HOME/conf/server.xml

<Context docBase="E:/java\_codes/TestSession/WebContent" path="" reloadable="true" >

<Manager className="de.javakaffee.web.msm.MemcachedBackupSessionManager"

memcachedNodes="n1:localhost:11211"

requestUriIgnorePattern=".\*\.(png|gif|jpg|css|js)$"

sessionBackupAsync="false"

sessionBackupTimeout="100"

transcoderFactoryClass="de.javakaffee.web.msm.serializer.javolution.JavolutionTranscoderFactory"

copyCollectionsForSerialization="false"

/>

</Context>

 这里的memcachedNodes是填写memcached节点,多个节点时可以以空隔分开，如:

 n1:localhost:11211 n2:localhost:11212

 sessionBackupTimeout的单位为分钟

 E:/java\_codes/TestSession/WebContent 替换成你的WEB目录

  修改后重启两个TOMCAT即可,这个时候已经解决SESSION的共享问题.

**二，配置nginx实现负载均衡**

   以我的nginx.conf为例

#user nobody;

worker\_processes 1;

error\_log logs/error.log;

events {

worker\_connections 1024;

}

http {

include mime.types;

default\_type application/octet-stream;

sendfile on;

keepalive\_timeout 65;

#gzip on;

upstream www.docyeah.com {

server 192.168.1.11:8080;

server 192.168.1.101:8080;

}

server {

listen 80;

server\_name www.docyeah.com;

charset utf-8;

location / {

root html;

index index.html index.htm;

proxy\_pass http://www.docyeah.com;

proxy\_set\_header X-Real-IP $remote\_addr;

client\_max\_body\_size 100m;

}

location ~ ^/(WEB-INF)/ {

deny all;

}

error\_page 500 502 503 504 /50x.html;

location = /50x.html {

root html;

}

}

}

将www.docyeah.com替换成你的域名

192.168.1.11和192.168.1.101替换成你服务器的IP

OK，已经完成。启动nginx即可。

这是我采用的负载均衡及集群方案，希望大家拍砖.

ps:javaeye的编辑器有问题，代码里加颜色后居然变乱了

8 楼 jiopi 2010-05-28

前提：以下论述不包含 服务器 宕机 的有效处理  
  
其实同步session只是为了保证一个用户在访问时，无论访问到哪台服务器，session都要一致，那为什么要让一个用户去访问1台以上的服务器呢？  
  
ngnix做基于访问IP地址的分发策略 跟 session的策略不太一致，session 的 key一般是保存在cookie中，而 一个用户却可能改变IP，但cookie不变，session就应当保持  
  
所以我做了nginx的基于cookie中某个值的分发策略，在基于IP的策略中，nginx可以保证IP相同的必定访问同一台机器，基于cookie中某个值的分发策略则可以保证只要cookie不丢，就可以访问同一台机器，避免因为同一用户可能在集群环境中访问多台服务器而必须考虑session共享的复杂问题

9 楼 lnaigg 2010-05-28

**jiopi 写道**

前提：以下论述不包含 服务器 宕机 的有效处理  
  
其实同步session只是为了保证一个用户在访问时，无论访问到哪台服务器，session都要一致，那为什么要让一个用户去访问1台以上的服务器呢？  
  
ngnix做基于访问IP地址的分发策略 跟 session的策略不太一致，session 的 key一般是保存在cookie中，而 一个用户却可能改变IP，但cookie不变，session就应当保持  
  
所以我做了nginx的基于cookie中某个值的分发策略，在基于IP的策略中，nginx可以保证IP相同的必定访问同一台机器，基于cookie中某个值的分发策略则可以保证只要cookie不丢，就可以访问同一台机器，避免因为同一用户可能在集群环境中访问多台服务器而必须考虑session共享的复杂问题

同意，基于IP分发的策略已经能解决绝大部分需求。