[**Nginx+tomcat配置集群负载均衡**](http://www.blogjava.net/niumd/archive/2011/05/10/349958.html)

  作者：niumd

  Blog:http://ari.iteye.com

  转载请注明出处，谢谢

    开发的应用采用F5负载均衡交换机，F5将请求转发给5台hp unix服务器，每台服务器有多个webserver实例，对外提供web服务和socket等接口服务。之初，曾有个小小的疑问为何不采用开源的 apache、Nginx软件负载，F5设备动辄几十万，价格昂贵？自己一个比较幼稚的问题，后续明白：F5是操作于IOS网络模型的传输 层，Nginx、apache是基于http反向代理方式，位于ISO模型的第七层应用层。直白些就是TCP UDP 和http协议的区别，Nginx不能为基于TCP协议的应用提供负载均衡。

      了解了二者之间的区别于应用场景，对Nginx产生浓厚的兴趣，阅读张宴的<实战Nginx>（这个85年的小伙子年轻有为羡慕+妒忌），搞 明白了大致原理和配置，Ubuntu10.10，window下对Nginx+tomcat负载均衡做了配置尝试，将全部请求转发到tomcat，并未做 静态，动态分开，图片防盗链等配置。  
**Nginx 介绍**

     Nginx （发音同 engine x）是一款轻量级的Web 服务器／反向代理服务器及电子邮件（IMAP/POP3）代理服务器，并在一个BSD-like 协议下发行。  其特点是占有内存少，并发能力强，事实上nginx的并发能力确实在同类型的网页伺服器中表现较好.目前中国大陆使用nginx网站用户有：新浪、网易、 腾讯,另外知名的微网志Plurk也使用nginx。

    上面的全是Nginx介绍基本上是废话，下面转入正题，图文结合展示基本配置，首先是window环境、其次是Ubuntu环境（Vbox虚拟）。本文主要基于Nginx下配置两台tomcat，结构如下图：

**Window xp环境：Nginx+Tomcat6**

**1、下载地址**

<http://nginx.org/en/download.html> ，这里我们推荐下载稳定版（stable versions），本文采用nginx-0.8.20。

**2、目录结构**

      Nginx-

               |\_  conf   配置目录

               |\_  contrib

               |\_  docs 文档目录

               |\_  logs  日志目录

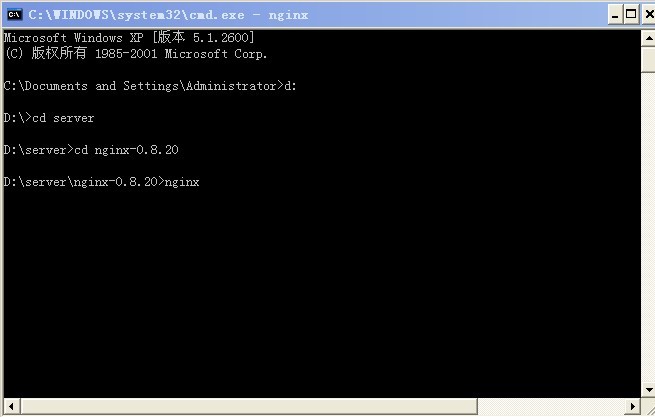
               |\_  temp 临时文件目录

               |\_  html 静态页面目录

               |\_  nginx.exe 主程序

      window下安装Nginx极其简单，解压缩到一个无空格的英文目录即可（个人习惯，担心中文出问题），双击nginx启动，这里我安装到：D:\server目录，下面涉及到的tomcat也安装在此目录。

DOS环境启动



若果想停止nginx，dos环境运行命令：nginx -s stop

**3、nginx.conf配置**

   Nginx配置文件默认在conf目录，主要配置文件为nginx.conf，我们安装在D:\server\nginx-0.8.20、默认主配置文件 为D:\server\nginx-0.8.20\nginx.conf。下面是nginx作为前端反向代理服务器的配置。

#Nginx所用用户和组，window下不指定

#user niumd niumd;

#工作的子进程数量（通常等于CPU数量或者2倍于CPU）

worker\_processes 2;

#错误日志存放路径

#error\_log logs/error.log;

#error\_log logs/error.log notice;

error\_log logs/error.log info;

#指定pid存放文件

pid logs/nginx.pid;

events {

#使用网络IO模型linux建议epoll，FreeBSD建议采用kqueue，window下不指定。

#use epoll;

#允许最大连接数

worker\_connections 2048;

}

http {

include mime.types;

default\_type application/octet-stream;

#定义日志格式

#log\_format main '$remote\_addr - $remote\_user [$time\_local] $request '

# '"$status" $body\_bytes\_sent "$http\_referer" '

# '"$http\_user\_agent" "$http\_x\_forwarded\_for"';

#access\_log off;

access\_log logs/access.log;

client\_header\_timeout 3m;

client\_body\_timeout 3m;

send\_timeout 3m;

client\_header\_buffer\_size 1k;

large\_client\_header\_buffers 4 4k;

sendfile on;

tcp\_nopush on;

tcp\_nodelay on;

#keepalive\_timeout 75 20;

include gzip.conf;

upstream localhost {

#根据ip计算将请求分配各那个后端tomcat，许多人误认为可以解决session问题，其实并不能。

#同一机器在多网情况下，路由切换，ip可能不同

#ip\_hash;

server localhost:18081;

server localhost:18080;

}

server {

listen 80;

server\_name localhost;

location / {

proxy\_connect\_timeout 3;

proxy\_send\_timeout 30;

proxy\_read\_timeout 30;

proxy\_pass http://localhost;

}

}

}

   代理设置如下：

proxy\_redirect off;

proxy\_set\_header Host $host;

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

proxy\_set\_header X-Forwarded-For $proxy\_add\_x\_forwarded\_for;

client\_max\_body\_size 10m;

client\_body\_buffer\_size 128k;

proxy\_connect\_timeout 300;

proxy\_send\_timeout 300;

proxy\_read\_timeout 300;

proxy\_buffer\_size 4k;

proxy\_buffers 4 32k;

proxy\_busy\_buffers\_size 64k;

proxy\_temp\_file\_write\_size 64k;

   gzip压缩相关配置如下：

gzip on;

gzip\_min\_length 1000;

gzip\_types text/plain text/css application/x-javascript;

**4、Tomcat配置**

   对于tomcat大家都很熟悉，只需要修改server.xml配置文件即可，这里我们以apache-tomcat-6.0.14为例，分别在 server目录，解压缩并命名为：apache-tomcat-6.0.14\_1、apache-tomcat-6.0.14\_2。

    第一处端口修改：

<!-- 修改port端口：18006 俩个tomcat不能重复，端口随意，别太小-->

<Server port="18006" shutdown="SHUTDOWN">

   第二处端口修改：

<!-- port="18081" tomcat监听端口，随意设置，别太小 -->

<Connector port="18081" protocol="HTTP/1.1"

connectionTimeout="20000"

redirectPort="8443" />

   第三处端口修改：

<Connector port="8009" protocol="AJP/1.3" redirectPort="8443" />

   Engine元素增加jvmRoute属性：

<Engine name="Catalina" defaultHost="localhost" jvmRoute="tomcat1">

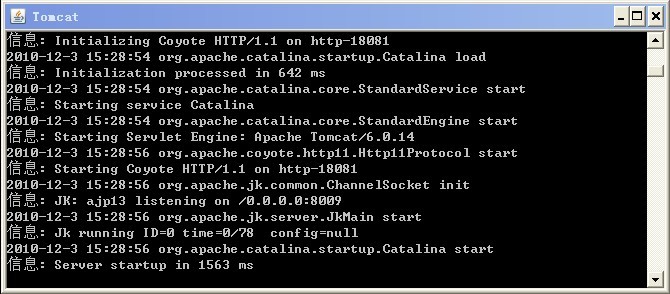
    两个tomcat的端口别重复，保证能启动起来，另一个tomcat配置希捷省略，监听端口为18080，附件中我们将上传所有的配置信息。

**5、验证配置与测试负载均衡**

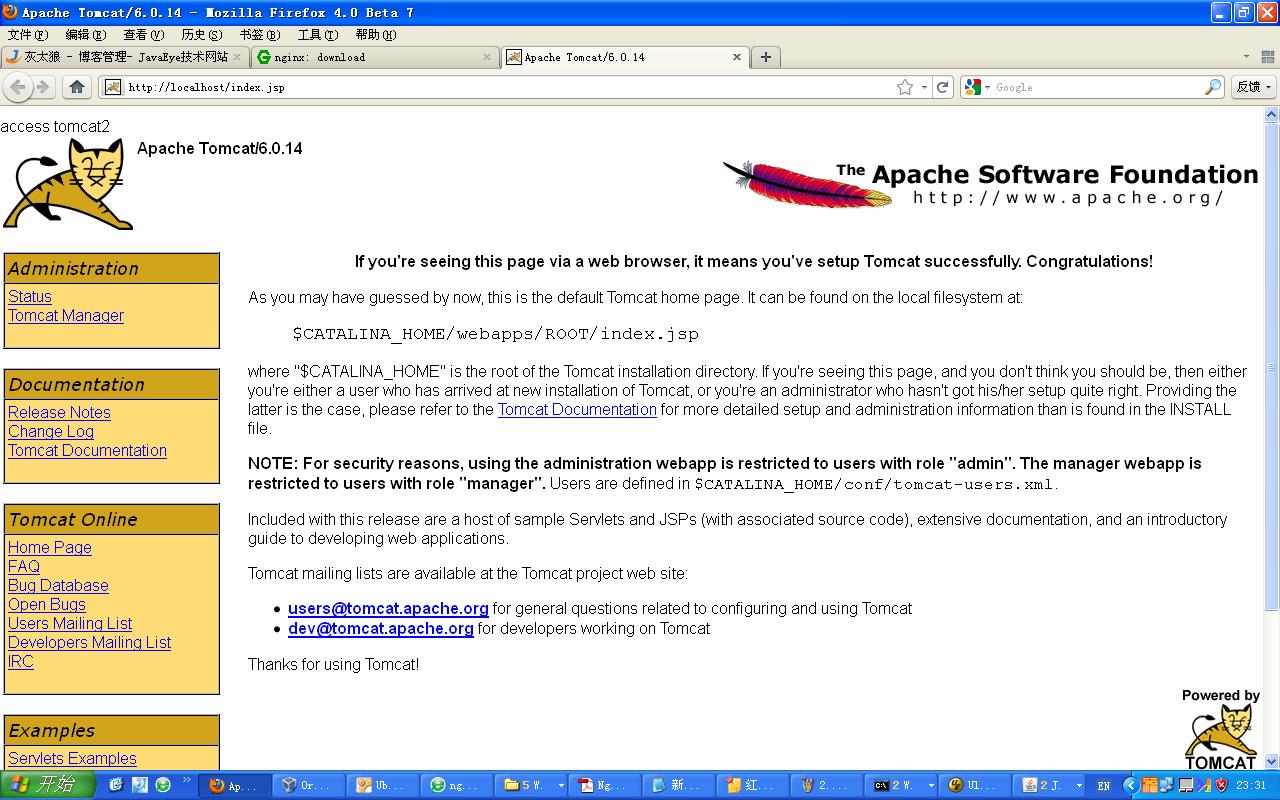
    首先测试nginx配置是否正确，测试命令：nginx -t  (默认验证:conf\nginx.conf),也可以指定配置文件路径。

 此例nginx安装目录：D:\server\nginx-0.8.20，dos环境下图画面成功示例：  


   其次验证tomcat，启动两个tomcat，不出现端口冲突即为成功（tomcat依赖的java等搞“挨踢”的就废话不说了）；



    最后验证配置负载均衡设置，<http://localhost/> 或<http://localhost/index.jsp> 。我修改了index.jsp页面，增加日志输出信息，便于观察。注意：左上角小猫头上的：access tomcat2、access tomcat1。说明访问了不同的tomcat。

  
   
     至此window下nginx+tomcat负载均衡配置结束，关于tomcat Session的问题通常是采用memcached，或者采用nginx\_upstream\_jvm\_route ，他是一个 Nginx 的扩展模块，用来实现基于 Cookie 的 Session Sticky 的功能。如果tomcat过多不建议session同步，server间相互同步session很耗资源，高并发环境容易引起Session风暴。请根据 自己应用情况合理采纳session解决方案。

 作者：niumd

  Blog:http://ari.iteye.com

**Ubuntu10.10环境：Nginx+Tomcat6**

我们下面简单说下ubuntu10.10下如何安装配置，主要以图片为主，简单解释。

**1、下载Nginx**

      地址：<http://nginx.org/en/download.html> ，linux版本：nginx-0.8.20.tar.。解压缩命令：

tar -zxvf nginx-0.8.20.tar.gz

**2、编译安装Nginx**

     Nginx依赖一些其他PCRE、openssl（依赖libssl-dev），本人笔记本Ubuntu环境已经安装PCRE，仅需安装依赖的openssl，下面我们简单说下如何安装PCRE和openssl等

     PCRE下载地址：<ftp://ftp.csx.cam.ac.uk/pub/software/programming/pcre/>

tar zxvf pcre-8.01.tar.gz

cd pcre-8.01

sudo ./configure

sodu make

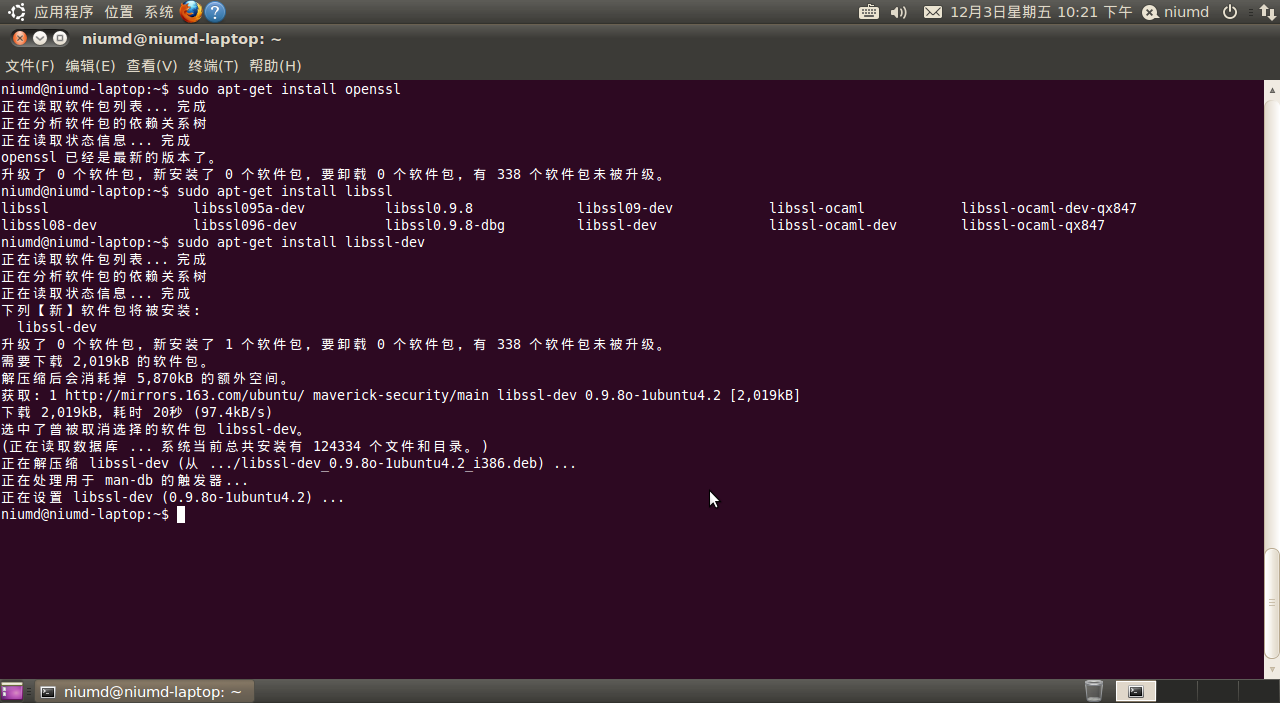
sodu make install

    openssl通过apt-get install安装，命令、截图如下：

sudo apt-get install openssl

sudo apt-get install libssl-dev

//如缺少其他包，请采用此方法安装，ubuntu有依赖提示

  
   
 依赖的软件包安装完毕，下面来编译Nginx：

#将window共享目录软件拷贝到当前工作目录

cp /mnt/fileshare/nginx-0.8.20.tar.gz ./

#解压缩软件包

tar zxvf nginx-0.8.20.tar.gz

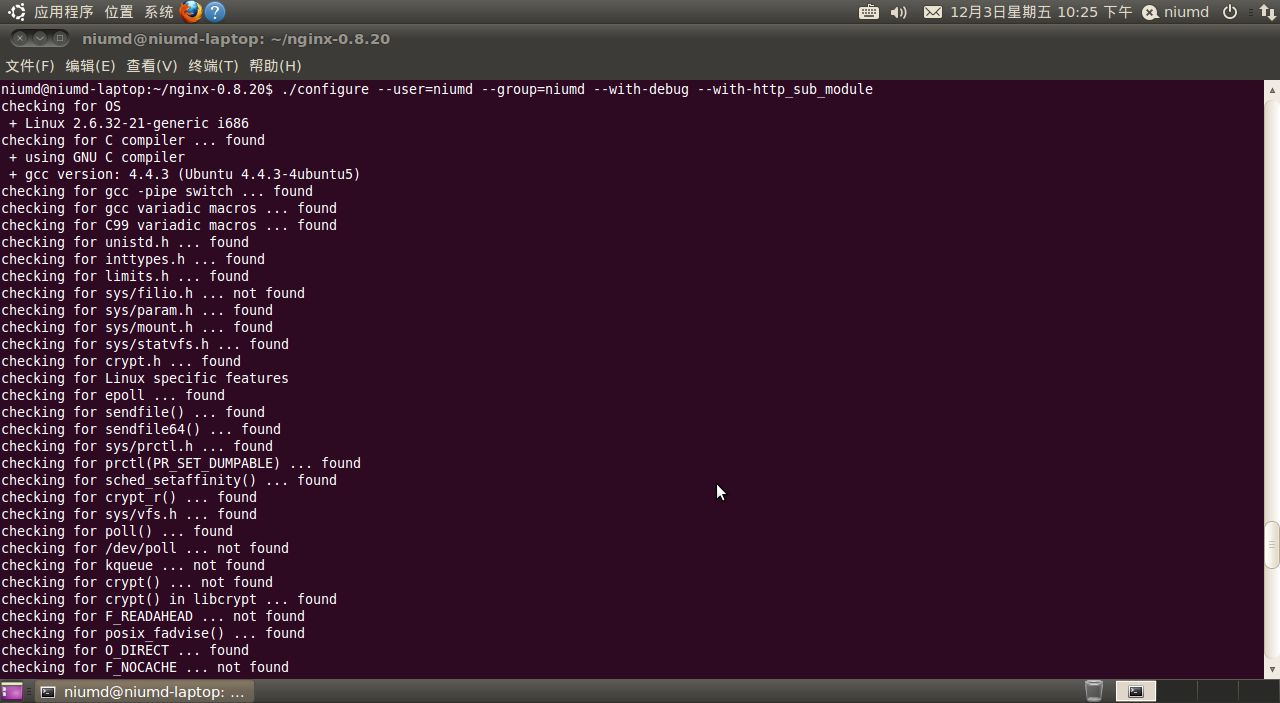
cd nginx-0.8.20

//编译源码,默认使用nobody，指定本机已存在的用户，组，启用nginx-status功能，监控nginx状态。启动debug

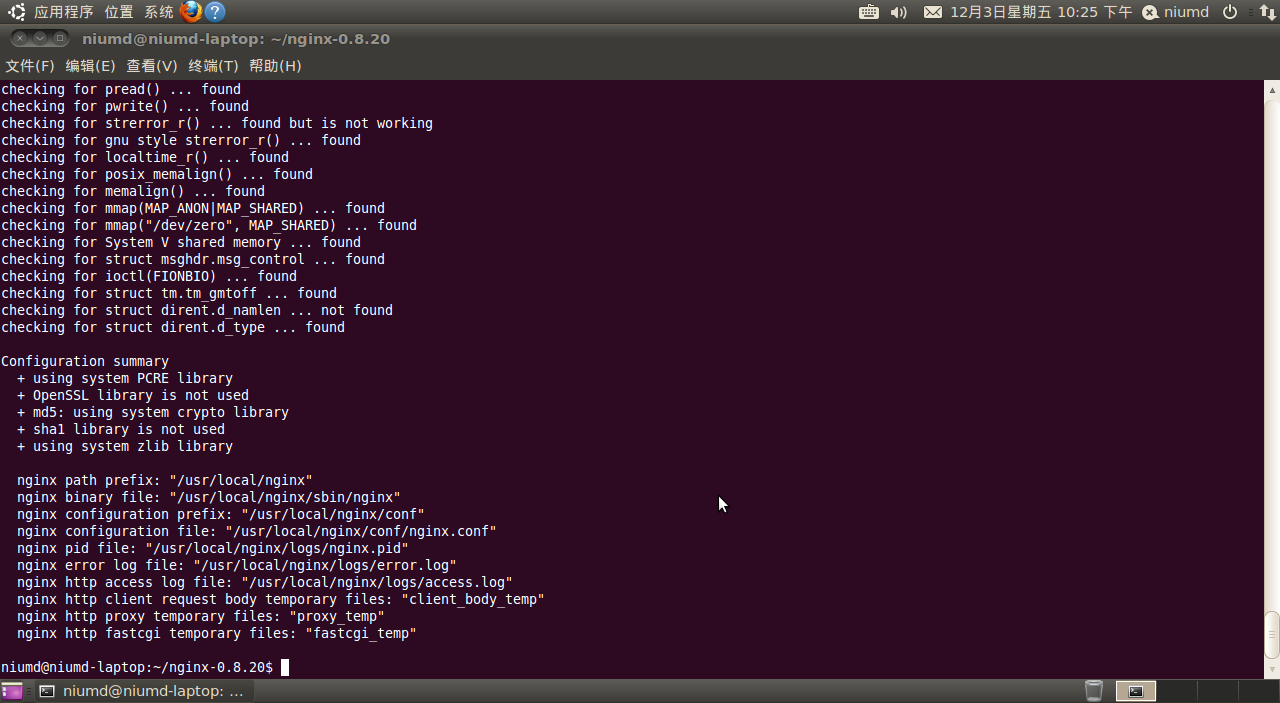
sudo ./configure --user=niumd --group=niumd --with-debug --with-http\_stub\_status\_module

sudo make

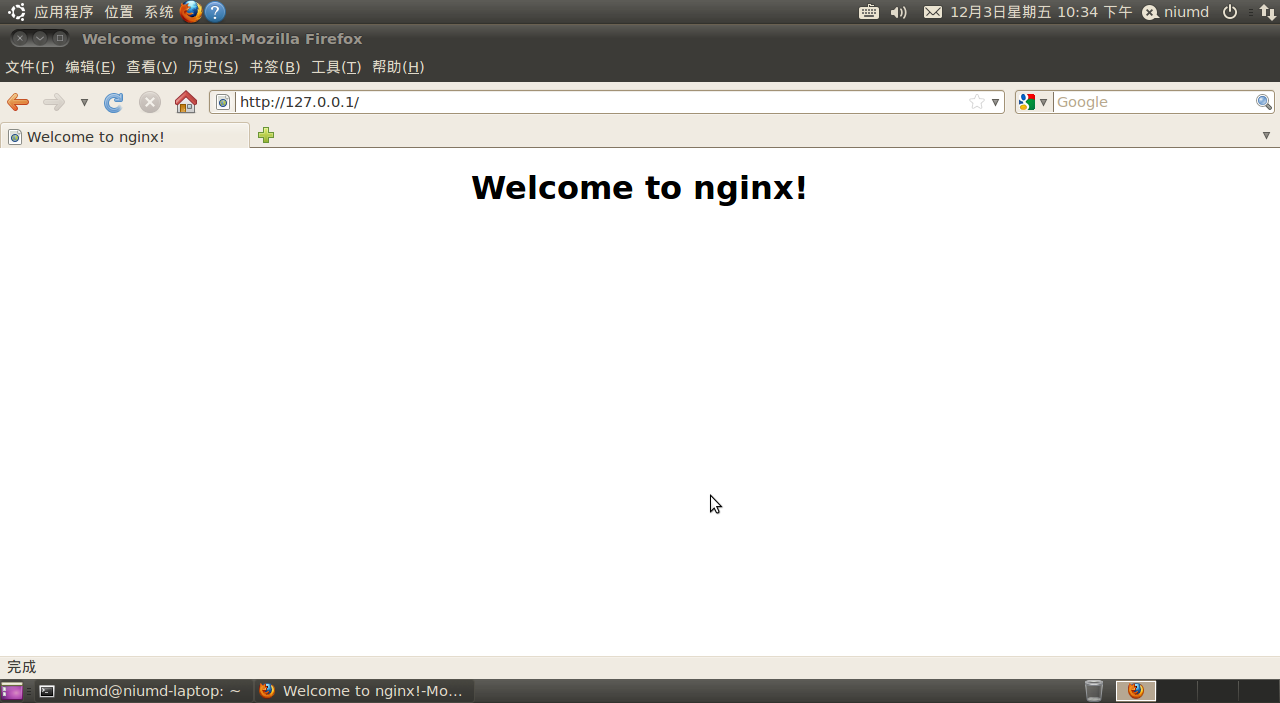
sudo make install

   截图 如下：  


安装结果截图如下：



编译安装正确结束，按照上述window下方法检查默认配置，然后在默认配置下启动nginx，访问[http://127.0.0.1](http://127.0.0.1/) ，如下图说明成功



Nginx配置成功后我们对window下nginx.conf少做修改，如下：

#Nginx所用用户和组

user niumd niumd;

#工作的子进程数量（通常等于CPU数量或者2倍于CPU）

worker\_processes 2;

#错误日志存放路径

#error\_log logs/error.log;

#error\_log logs/error.log notice;

error\_log logs/error.log info;

#指定pid存放文件

pid logs/nginx.pid;

events {

#使用网络IO模型linux建议epoll，FreeBSD建议采用kqueue

use epoll;

#允许最大连接数

worker\_connections 2048;

}

http {

include mime.types;

default\_type application/octet-stream;

#定义日志格式

#log\_format main '$remote\_addr - $remote\_user [$time\_local] $request '

# '"$status" $body\_bytes\_sent "$http\_referer" '

# '"$http\_user\_agent" "$http\_x\_forwarded\_for"';

#access\_log off;

access\_log logs/access.log;

client\_header\_timeout 3m;

client\_body\_timeout 3m;

send\_timeout 3m;

client\_header\_buffer\_size 1k;

large\_client\_header\_buffers 4 4k;

sendfile on;

tcp\_nopush on;

tcp\_nodelay on;

#keepalive\_timeout 75 20;

include gzip.conf;

upstream localhost {

#ip\_hash

#ip\_hash;

server localhost:18081;

server localhost:18080;

}

server {

listen 80;

server\_name localhost;

location / {

proxy\_connect\_timeout 3;

proxy\_send\_timeout 30;

proxy\_read\_timeout 30;

proxy\_pass http://localhost;

}

}

}

  对于上面关于ubuntu下Nginx配置和window下基本相同，区别在使用的IO网络模型，linux下建议使用epoll，另外就是运行所用的用户和组；

**3、配置tomcat**

    请参考window下配置，完全相同。

**4、启动停止nginx**

ubuntu下启动nginx与window稍有不同，大致启动停止方法如下。

#nginx目录执行

sbin/nginx

或通过-c 指定配置文件

sbin/nginx -c usr/local/nginx8.20/conf/nginx/conf

niumd@niumd-laptop:/usr/local/nginx$ pwd

/usr/local/nginx

niumd@niumd-laptop:/usr/local/nginx$ sudo sbin/nginx -t

the configuration file /usr/local/nginx/conf/nginx.conf syntax is ok

configuration file /usr/local/nginx/conf/nginx.conf test is successful

niumd@niumd-laptop:/usr/local/nginx$ sudo sbin/nginx -v

nginx version: nginx/0.8.20

niumd@niumd-laptop:/usr/local/nginx$ sudo sbin/nginx -V

nginx version: nginx/0.8.20

built by gcc 4.4.3 (Ubuntu 4.4.3-4ubuntu5)

configure arguments: --user=niumd --group=niumd --with-debug --with-http\_sub\_module

niumd@niumd-laptop:/usr/local/nginx$ sudo sbin/nginx

niumd@niumd-laptop:/usr/local/nginx$ ps -ef|grep nginx

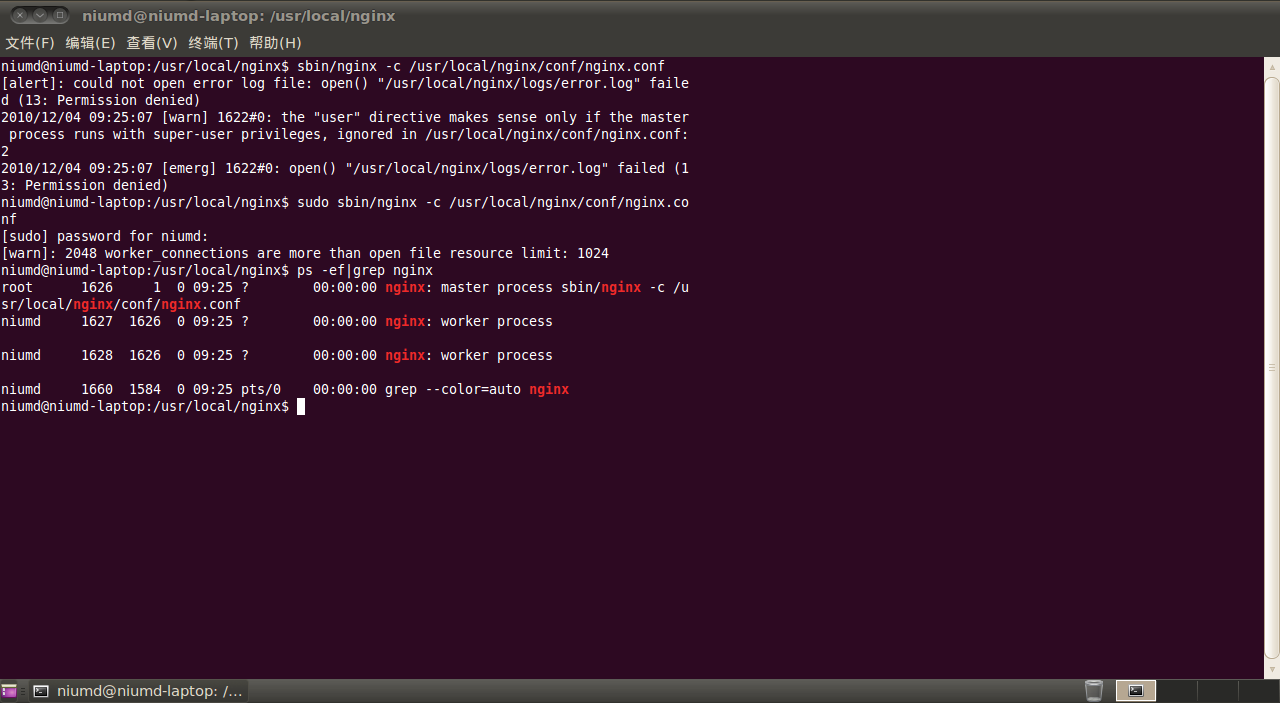
root 5158 1 0 22:32 ? 00:00:00 nginx: master process sbin/nginx

niumd 5159 5158 0 22:32 ? 00:00:00 nginx: worker process

niumd 5161 1577 0 22:32 pts/0 00:00:00 grep --color=auto nginx

niumd@niumd-laptop:/usr/local/nginx$

     我们通过ps  -ef|grep nginx,看到如下结果:



注意：在启动时linux提示一句警告【warn】……，是因为我们设置的 #允许最大连接数 worker\_connections  2048，超过linux默认1024的限制。

       停止：kill -信号类型 pid

       nginx/logs目录下有个nginx。pid的文件，此文件记录了每次运行的pid，也可以通过ps命令查询。

信号类型如下：

|  |  |
| --- | --- |
| **信号控制** | |
| **信号类型** | **描述** |
| RERM.INT | 快速关闭 |
| HUP | 平滑重启，加载配置 |
| USR1 | 重新加载日志 |
| USER2 | 平滑升级执行程序 |
| WINCH | 从容关闭工作进程 |
| QUIT | 从容关闭 |

参考资料：

<http://www.oschina.net/bbs/thread/9301>

oschina.net 生产配置，此网站采用java语言，nginx，tomcat服务器。

<http://nginx.org/>

张宴：<<实战Nginx>>

web服务器集群（多台web服务器）session同步、共享的3种解决方法

在做了web集群后，你肯定会首先考虑session同步问题，因为通过负载均衡后，同一个IP访问同一个页面会被分配到不同的服务器上，如果session不同步的话，一个登录用户，一会是登录状态，一会又不是登录状态。所以本文就根据这种情况给出三种不同的方法来解决这个问题：

一、利用数据库同步session

在做多服务器session同步时我没有用这种方法，如果非要用这种方法的话，我想过二种方法：

1，用一个低端电脑建个数据库专门存放web服务器的session，或者，把这个专门的数据库建在文件服务器上，用户访问web服务器时，会去这个专门的数据库check一下session的情况，以达到session同步的目的。

2，这种方法是把存放session的表和其他数据库表放在一起，如果mysql也做了集群了话，每个mysql节点都要有这张表，并且这张session表的数据表要实时同步。

说明：用数据库来同步session，会加大数据库的负担，数据库本来就是容易产生瓶颈的地方，如果把session还放到数据库里面，无疑是雪上加霜。上面的二种方法，第一点方法较好，把放session的表独立开来，减轻了真正数据库的负担

二、利用cookie同步session

session是文件的形势存放在服务器端的，cookie是文件的形势存在客户端的，怎么实现同步呢？方法很简单，就是把用户访问页面产生的session放到cookie里面，就是以cookie为中转站。你访问web服务器A，产生了session把它放到cookie里面了，你访问被分配到web服务器B，这个时候，web服务器B先判断服务器有没有这个session，如果没有，在去看看客户端的cookie里面有没有这个session，如果也没有，说明session真的不存，如果cookie里面有，就把cookie里面的sessoin同步到web服务器B，这样就可以实现session的同步了。

说明：这种方法实现起来简单，方便，也不会加大数据库的负担，但是如果客户端把cookie禁掉了的话，那么session就无从同步了，这样会给网站带来损失；cookie的安全性不高，虽然它已经加了密，但是还是可以伪造的。

三、利用memcache同步session

memcache可以做分布式，如果没有这功能，他也不能用来做session同步。他可以把web服务器中的内存组合起来，成为一个"内存池"，不管是哪个服务器产生的sessoin都可以放到这个"内存池"中，其他的都可以使用。

优点：以这种方式来同步session，不会加大数据库的负担，并且安全性比用cookie大大的提高，把session放到内存里面，比从文件中读取要快很多。

缺点：memcache把内存分成很多种规格的存储块，有块就有大小，这种方式也就决定了，memcache不能完全利用内存，会产生内存碎片，如果存储块不足，还会产生内存溢出。

四、总结

上面三种方法都是可行的

第一种方法，最影响系统速度的那种，不推荐使用；

第二种方法，效果不错，不过安全隐患一样的存在；

第三种方法，个人觉得第三种方法是最好的，推荐大家使用