# 01【熟悉】服务器概述

**1，目前常见的web服务器**

1，Apache(<http://httpd.apache.org)>

    它是世界上用的最多的web服务器，市场占有率达60%左右，模块非常丰富，系统非常稳定，可移植性好，但是比较消耗资源

2，lighttpd(<http://www.lighttpd.net)>

    它是德国人开发的一个开源软件，目标是提供一个高性能的网站，它具有内存开销低，cpu占用低，效能好及模块丰富，Nginx的重要竞争对手之一

3，tomcat(<http://tomcat.apache.org)>

    是一个开源的软件，运行servlet+jsp web应用软件，对静态文件，高并发的处理能力弱。

4，IBM websphere

    它功能完善，开放的Web应用程序服务器，是IBM电子商务计划的核心部分，它是基于java的应用环境，范围从简单到高级到企业级应用，相于对其它web服务器来说应该比较少

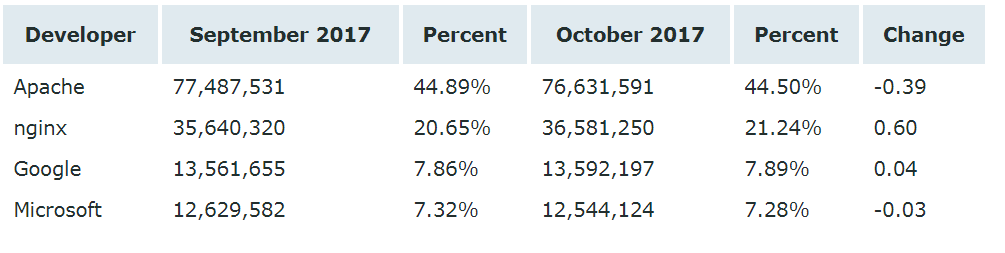
5，Microsoft IIS

     Microsoft的web服务器产品为Internet information Server (IIS)  IIS提供了图形界面管理工具，IIS是一种web服务器组件，其中有 web服务器，FTP服务器，nntp服务器，smtp服务器，因为有window2008和2012的支持，所以IIS也有一定的市场

**2，各web服务器占有率**

查询网站

<https://news.netcraft.com/archives/2017/10/26/october-2017-web-server-survey-13.html>



# 02【熟悉】Nginx概述

作用：提一个web服务器， 是一个高性能的反向代理服务器

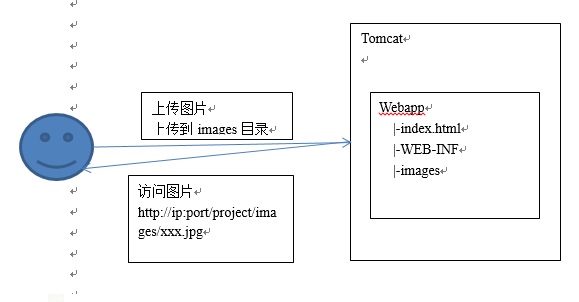
# 1、前言

      为毛要用nginx服务器代理，不直接用tomcat 7.0，还做多了一次接请求？

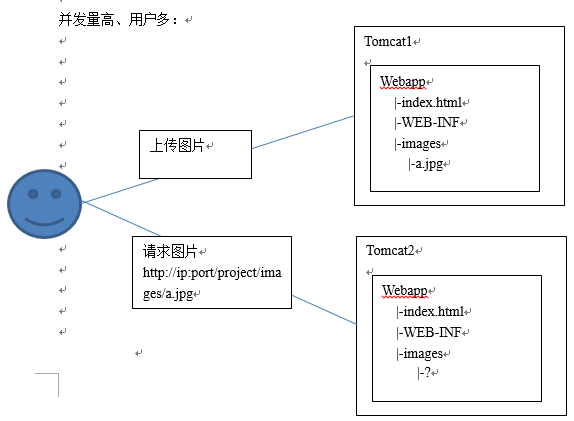
      这个是我想问的，公司的新项目是要用Nginx+tomcat7+jdk开发的，用户命名可以直接访问tomcat，为啥还要用Nginx？这货是个啥玩意？

# 2、为什么使用Nginx？

      在传统的Web项目中，并发量小，用户使用的少。所以在低并发的情况下，用户可以直接访问tomcat服务器，然后tomcat服务器返回消息给用户。比如，我们上传图片：



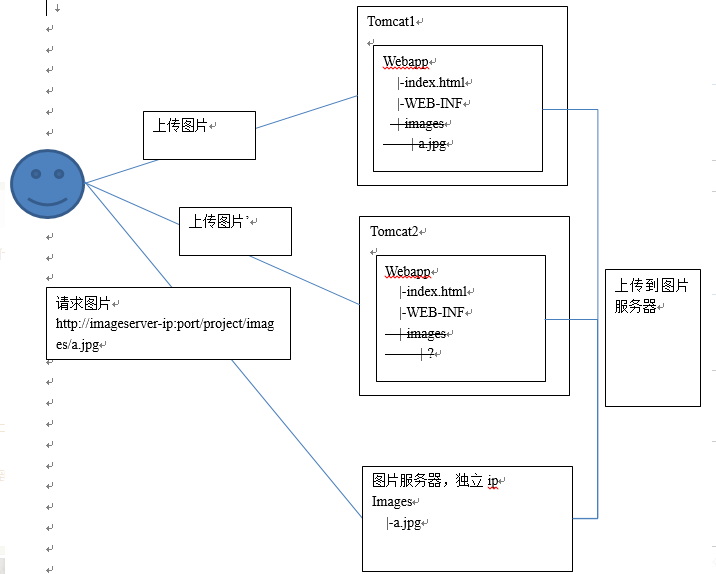
当然我们知道，为了解决并发，可以使用负载均衡：也就是我们多增加几个tomcat服务器。当用户访问的时候，请求可以提交到空闲的tomcat服务器上。



 但是这种情况下可能会有一种这样的问题：上传图片操作。我们把图片上传到了tomcat1上了，当我们要访问这个图片的时候，tomcat1正好在工作，所以访问的请求就交给其他的tomcat操作，而tomcat之间的数据没有进行同步，所以就发生了我们要请求的图片找不到。

      为了解决这种情况，我们就想出了分布式。我们专门建立一个图片服务器，用来存储图片。这样当我们都把图片上传的时候，不管是哪个服务器接收到图片，都把图片上传到图片服务器。

      图片服务器上需要安装一个http服务器，可以使用tomcat、apache、nginx。



 看到这里大家可能会问，既然我们要选择的是http服务器，为什么不继续使用tomcat，而要使用Nginx？

      原因如下：nginx常用做静态内容服务和代理服务器（不是你FQ那个代理），直面外来请求转发给后面的应用服务（tomcat，django什么的），tomcat更多用来做做一个应用容器，让java web app跑在里面的东西，对应同级别的有jboss,jetty等东西。

# 3、什么是Nginx？

      根据前面的对比，我们可以了解到Nginx是一个[http服务器](http://baike.baidu.com/link?url=a2dLY11NbWgcCzbX1s7JDyWLOh_QFjVlC2wys--TLKbZybTCA8oEP7c-5gEDCK35jFmQHG0YFRoAVEI8M7cbFcp74nDVgz1ckZiWAuntvCF_oxMSMDlDIWEGGN-63mTb)。是一个使用c语言开发的高性能的http服务器及反向代理服务器。Nginx是一款高性能的http 服务器/反向代理服务器及电子邮件（IMAP/POP3）代理服务器。由俄罗斯的程序设计师Igor Sysoev所开发，官方测试nginx能够支支撑5万并发链接，并且cpu、内存等资源消耗却非常低，运行非常稳定。

## 3.1 Nginx的应用场景

      1、 http服务器。Nginx是一个http服务可以独立提供http服务。可以做网页静态服务器。

      2、 虚拟主机。可以实现在一台服务器虚拟出多个网站。例如个人网站使用的虚拟主机。

* 基于端口的，不同的端口
* 基于域名的，不同域名

      3、 反向代理，负载均衡。当网站的访问量达到一定程度后，单台服务器不能满足用户的请求时，需要用多台服务器集群可以使用nginx做反向代理。并且多台服务器可以平均分担负载，不会因为某台服务器负载高宕机而某台服务器闲置的情况。

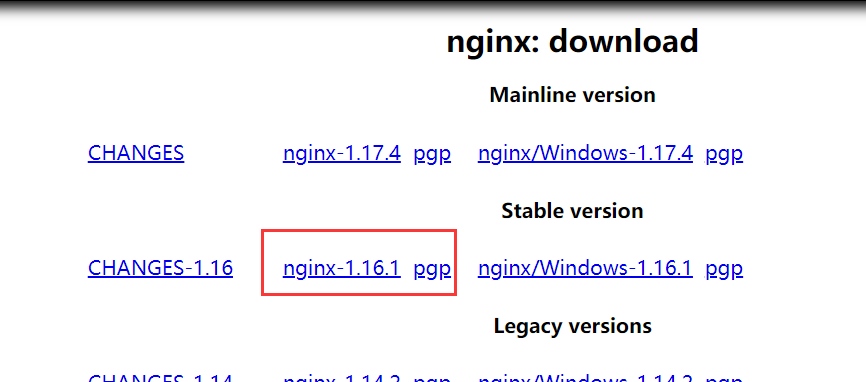
# 4、小结

      通过了解Nginx也算是多了一门技术，而且还是挺简单的，使用也比较方便。重点是可以给系统带来很大的性能提升。

# 03【掌握】在Cento7系统下安装Nginx

## 1、下载Nginx

<http://nginx.org/en/download.html>





## 2、安装

### 2.1 环境要求

* Centos 7
* nginx-1.16.1.tar.gz

### 2.2 先安装nginx依赖的包

      nginx是C语言开发，建议在linux上运行，所以我们选择在linux上安装Nginx

* gcc

      安装nginx需要先将官网下载的源码进行编译，编译依赖gcc环境，如果没有gcc环境，需要安装gcc

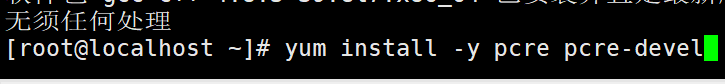


* PCRE

      PCRE(Perl Compatible Regular Expressions)是一个Perl库，包括 perl 兼容的正则表达式库。nginx的http模块使用pcre来解析正则表达式，所以需要在linux上安装pcre库。

yum install -y pcre pcre-devel

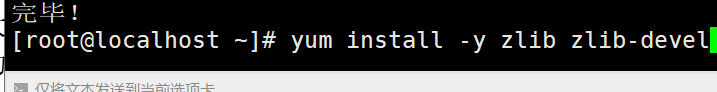
      注：pcre-devel是使用pcre开发的一个二次开发库。nginx也需要此库。



* zlib

      zlib库提供了很多种压缩和解压缩的方式，nginx使用zlib对http包的内容进行gzip，所以需要在linux上安装zlib库。

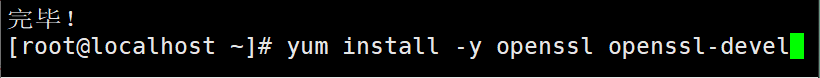
yum install -y zlib zlib-devel



* openssl

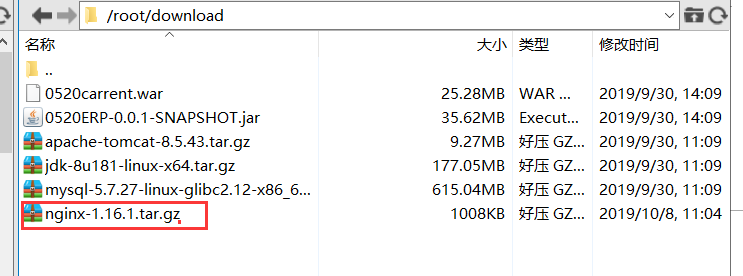
      OpenSSL 是一个强大的安全套接字层密码库，囊括主要的密码算法、常用的密钥和证书封装管理功能及SSL协议，并提供丰富的应用程序供测试或其它目的使用。   
      nginx不仅支持http协议，还支持https（即在ssl协议上传输http），所以需要在linux安装openssl库。

yum install -y openssl openssl-devel



### 2.3 把nginx的源码上传到linux系统，并解压

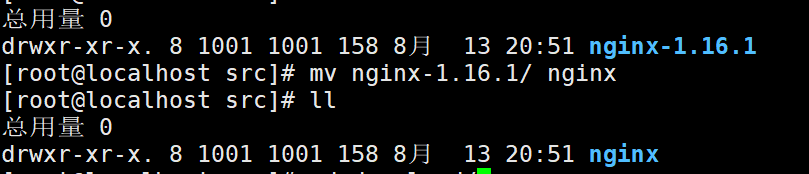
      这一步可以使用软件，也可以用代码的方式。nginx-1.16.1.tar.gz包拷贝到“/usr/local/src”下。然后解压。



解压

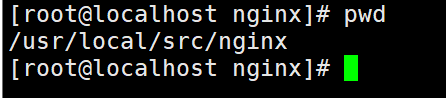


改名

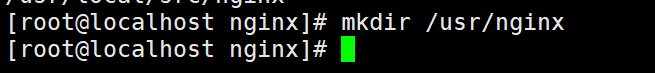


### 2.4 安装

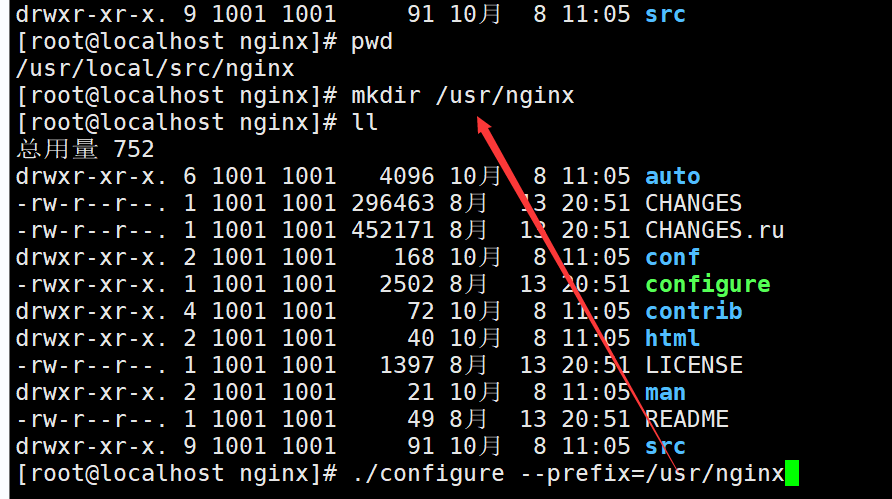
  1，    上面的步骤操作完成后，在/usr/local/src里面会有一个nginx的文件夹，进入后



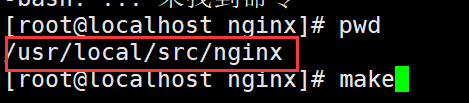
2，创建usr/nginx目录 ---安装目录



3,运行configure



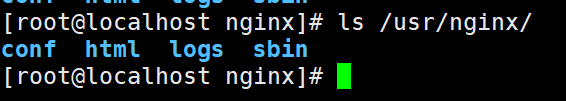
make编译



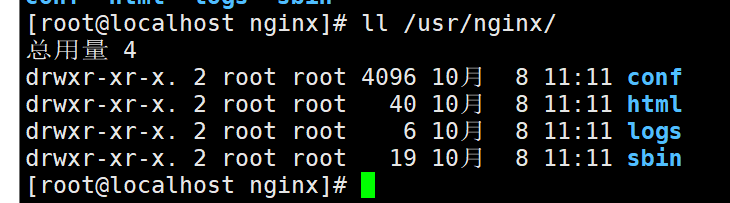
编译安装

make install

查询是否安装成功



### 2.5 Nginx的目录说明



Conf 配置文件

html 静态文件

logs 日志文件

sbin 命令 启动 停止等等命令

### 2.6 Nginx的​启动、停止

启动

Cd /usr/nginx

Cd sbin

./nginx

停止

./nginx -s stop

刷新:当修改了conf文件夹里面配置文件之后要执行的命令

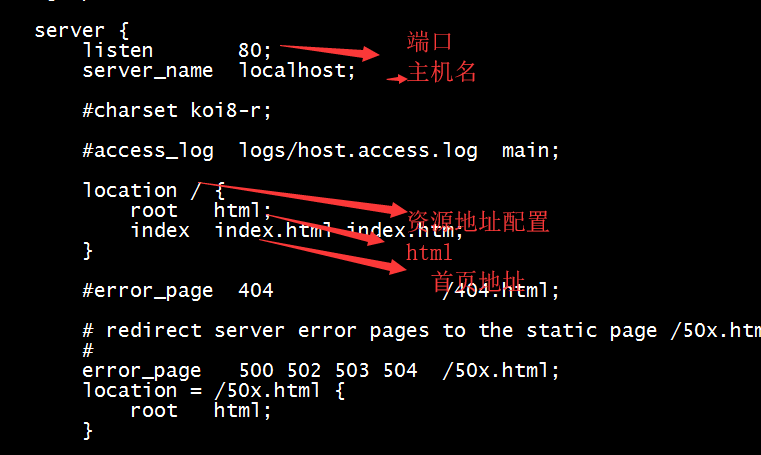
./nginx -s reload

# 04【掌握】Nginx基础配置详解

以conf/nginx.conf

# 1、Nginx的配置

      在/usr/nginx/conf目录下nginx.conf文件是nginx的配置文件。



## 1.1 端口和目录的配置

      在nginx.conf文件中添加一个server节点，修改端口号就可以【自行添加。不影响原来的】

也就是一个nginx里面可以跑多个端口的项目，这个是tomcat是有本质的区别的

一个tomcat只能对应一个端口的多个项目服务

一个nginx可以对应多个端口下面的多个项目服务

server {

listen 80;

server\_name localhost;

location / {

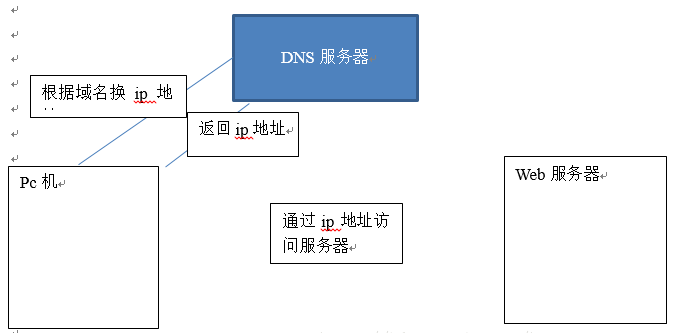
root html;

index index.html index.htm;

}

}

## 1.2 通过域名虚拟机



      1,需要修改nginx.conf配置文件，修改配置后需要重新加载配置文件：

server {

listen 80;

server\_name test1.whsxt.com;

location / {

root test1;

index index.html index.htm;

}

}

server {

listen 80;

server\_name test2.whsxt.com;

location / {

root test2;

index index.html index.htm;

}

}

2,在nginx目录下创建两个文件夹并把html里面的index.html复制进去

cd /usr/nginx

mkdir test1 test2

cp html/index.html test1

cp html/index.html test1

3,刷新nginx的配置

./sbin/nginx -s reload

可以通过修改windows 的host文件指定域名的ip地址

修改windows

C:\Windows\System32\drivers\etc

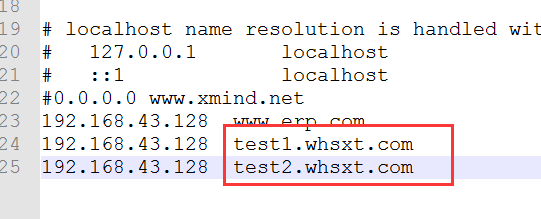
在后面加上

192.168.15.131  test1.whsxt.com

192.168.15.131  test2.whsxt.com

      然后可以通过访问test1.whsxt.com或者test2.whsx.com进行访问nginx。

      需要注意的是host文件的名字要和nginx.conf文件的server\_name相同。



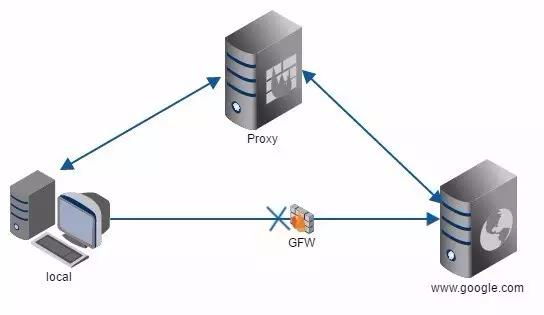
# 05【掌握】反向代理+负载均衡

## 1.1.正向代理

拿借钱打个比方，A想向C借钱，但是C不认识A所以不借给他，然后A就通过B向C借钱，B借到钱之后再转交给A，在这个过程中B就扮演了一个正向代理的角色，这个过程中，真正借钱的人是谁，C是不知道的~

我们常说的代理也就是指正向代理，正向代理的过程，它隐藏了真实的请求客户端，服务端不知道真实的客户端是谁，客户端请求的服务都被代理服务器代替来请求，科学上网工具 Shadowsocks 扮演的就是典型的正向代理角色。

比如我想访问www.google.com，要想翻越这堵墙，你可以在国外用Shadowsocks来搭建一台代理服务器，代理帮我们请求www.google.com，代理再把请求响应结果再返回给我。

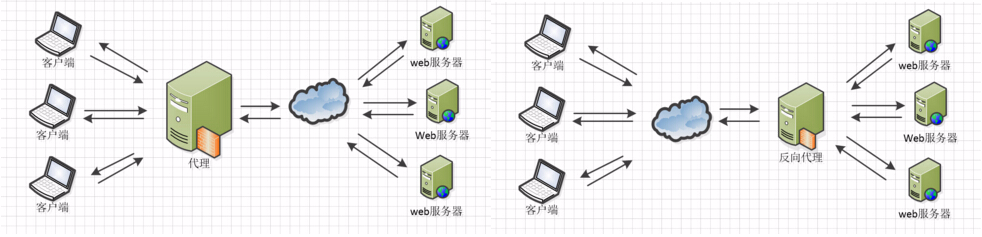


## 1.2.反向代理

还用借钱的例子，A想向C借钱，然后C借给他了，但是实际上这个钱可能C向B借的~至于钱到底是谁的，A是不知道的~

这里的C扮演着一个反向代理的角色，客户不知道真正提供服务的人是谁。

反向代理隐藏了真实的服务端，当我们访问www.baidu.com的时候，背后可能有成千上万台服务器为我们服务，但具体是哪一台，你不知道，也不需要知道，你只需要知道反向代理服务器是谁就好了。www.baidu.com就是我们的反向代理服务器，反向代理服务器会帮我们把请求转发到提供真实服务的服务器那里去。Nginx就是性能非常好的反向代理服务器，它可以用来做负载均衡。



## 2，负载均衡

网站的访问量越来越大，服务器的服务模式也得进行相应的升级，比如分离出数据库服务器、分离出图片作为单独服务，这些是简单的数据的负载均衡，将压力分散到不同的机器上。有时候来自web前端的压力，也能让人十分头痛。怎样将同一个域名的访问分散到两台或更多的机器上呢？这其实就是另一种负载均衡了，nginx自身就可以做到，只需要做个简单的配置就行。

　　nginx不单可以作为强大的web服务器，也可以作为一个反向代理服务器，而且nginx还可以按照调度规则实现动态、静态页面的分离，可以按照轮询、ip哈希、URL哈希、权重等多种方式对后端服务器做负载均衡，同时还支持后端服务器的健康检查。

Nginx负载均衡一些基础知识：

nginx 的 upstream目前支持 4 种方式的分配

1)、轮询（默认）

　　每个请求按时间顺序逐一分配到不同的后端服务器，如果后端服务器down掉，能自动剔除。

2)、weight

　　指定轮询几率，weight和访问比率成正比，用于后端服务器性能不均的情况。

2)、ip\_hash

　　每个请求按访问ip的hash结果分配，这样每个访客固定访问一个后端服务器，可以解决session的问题。

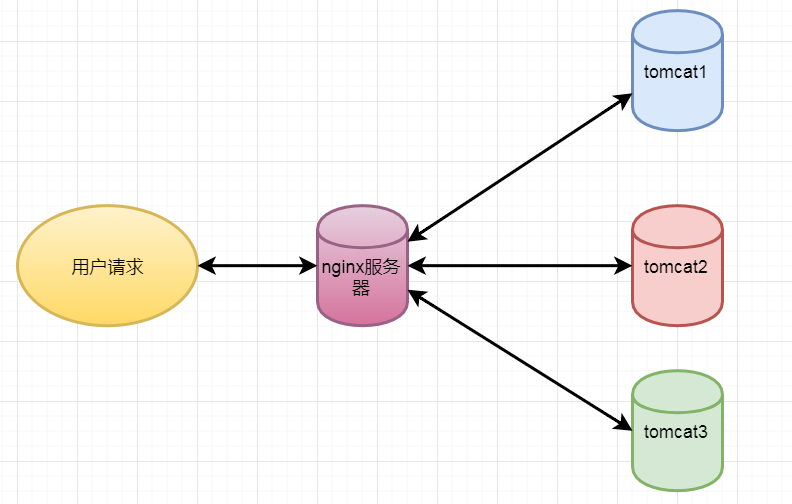
3)、fair（第三方）

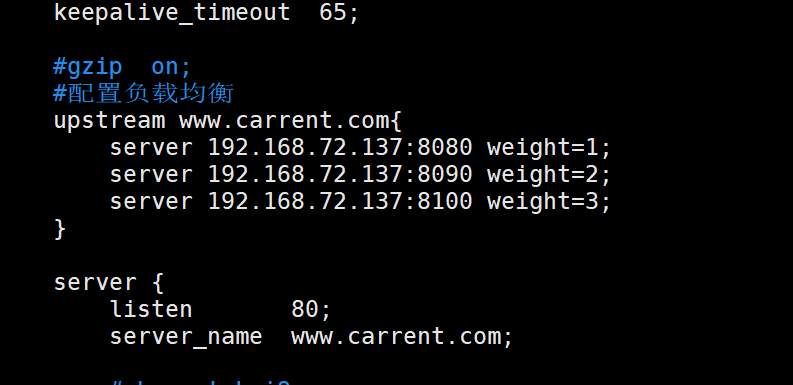
　　按后端服务器的响应时间来分配请求，响应时间短的优先分配。

4)、url\_hash（第三方）

## 3，反向代理+负载均衡的配置

**1，架构思想**





**2，准备三个tomcat**

#把原来的tomcat改个名这字

mv apache-tomcat-7.0.90 tomcat1

#复制两个

cp -r tomcat1 tomcat2/

cp -r tomcat1 tomcat3/

​

#修改端口号

tomcat1 8005 8080 8443 8009 8443 保持默认

tomcat2 8015 8090 8453 8019 8453

tomcat3 8025 8100 8463 8029 8463

​

#修改三个tomcat里面的webapps里面页面。让其有点区别【这里不做代码展示，自行去改】

​

#启动三个tomcat

cd tomcat1/bin

./startup

cd tomcat2/bin

./startup

cd tomcat3/bin

./startup

3，配置nginx服务器

#打开/usr/nginx

cd /usr/nginx

#打开conf配置目录

cd conf

#备份里面的nginx.conf[防止改错]

cp -r nginx.conf nginxbak.confbak

#编辑里面的nginx.conf

vi nginx.conf

​

​

#在server节点上加入以下配置

upstream www.carrent.com {

server 192.168.15.131:8080;

server 192.168.15.131:8090;

server 192.168.15.131:8100;

#ip\_hash;这是是用来解决登陆的session的问题

}

#修改location的配置

server {

listen 80;

server\_name www.carrent.com;#配置访问的域名

location / {

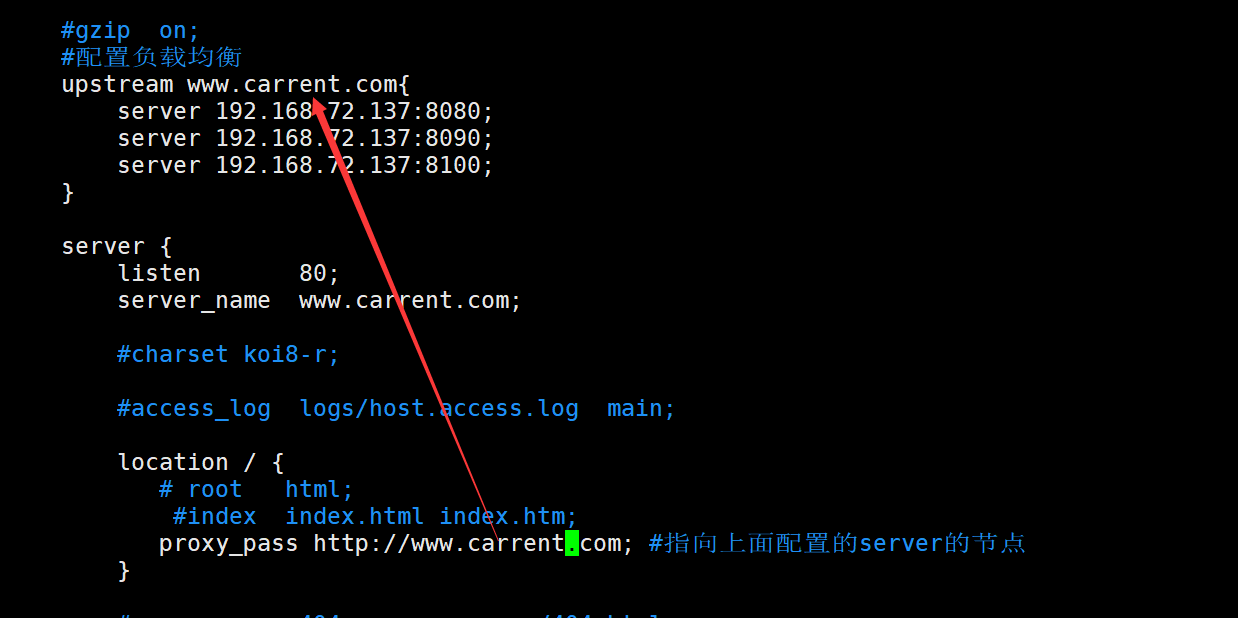
# root html;

# index index.html index.htm;

proxy\_pass http://www.carrent.com; #指向上面配置的server的节点

}

}



**4，启动nginx和访问**

#启动

./nginx

#访问

http://192.168.15.131/erp

**5，解决请求过慢的问题**

#在nginx.conf文件的location里面加入如下配置

proxy\_redirect off ;

proxy\_set\_header Host $host;

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

proxy\_set\_header REMOTE-HOST $remote\_addr;

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

client\_max\_body\_size 50m; #允许客户端请求的最大单文件字节数

client\_body\_buffer\_size 256k;#缓冲区代理缓冲用户端请求的最大字节数，

proxy\_connect\_timeout 1; #nginx跟后端服务器连接超时时间(代理连接超时)

proxy\_send\_timeout 30; #后端服务器数据回传时间(代理发送超时)

proxy\_read\_timeout 60; #连接成功后，后端服务器响应时间(代理接收超时)

proxy\_buffer\_size 256k;#设置代理服务器（nginx）保存用户头信息的缓冲区大小

proxy\_buffers 4 256k; #proxy\_buffers缓冲区，网页平均在256k下，这样设置

proxy\_busy\_buffers\_size 256k; #高负荷下缓冲大小（proxy\_buffers\*2）

proxy\_temp\_file\_write\_size 256k; #设定缓存文件夹大小

proxy\_next\_upstream error timeout invalid\_header http\_500 http\_503 http\_404;

proxy\_max\_temp\_file\_size 128m;

**6，可以配置window的host去指向**[**http://www.erp.com**](http://www.erp.com)

打开C:\Windows\System32\drivers\etc\hosts

添加如下配置

192.168.15.131 www.carrent.com

**7，配置hosts不生效的解决方法**

1，关闭浏览器再打开

2，刷新DNS

|--打开dos窗口

|--输入ipconfig /flushdns

## 4，其它配置

权重配置

upstream www.erp.com {

server 192.169.15.131:8080 weight=5;

server 192.168.15.131:8090 weight=3;

server 192.168.15.131:8100 weight=1;

#ip\_hash;这是是用来解决登陆的session的问题

}

#以下的配置是权重 最大请求失败次数 失败超时时间

#server 192.168.15.131:8080 weight=5 max\_fails=2 fail\_timeout=30s

## 5， 登陆session的配置问题

**1、不使用session，换用cookie**

session是存放在服务器端的，cookie是存放在客户端的，我们可以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然后把它放到cookie里面，当你的请求被分配到B服务器时，服务器B先判断服务器有没有这个session，如果没有，再去看看客户端的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。

**2、session存在数据库（MySQL等）中**

可以配置将session保存在数据库中，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IO，增加数据库的负担。而且数据库读写速度较慢，不利于session的适时同步。

**3、session存在memcache或者redis中**

memcache可以做分布式，php配置文件中设置存储方式为memcache，这样php自己会建立一个session集群，将session数据存储在memcache中。

说明：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。但是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。

**4、ip\_hash**

upstream www.erp.com {

server 192.168.15.131:8080;

server 192.168.15.131:8090;

server 192.168.15.131:8100;

ip\_hash;#一种IP的算法，只要用户使用同一个IP请求，nginx会分流到之前请求的服务器

}

nginx中的ip\_hash技术能够将某个ip的请求定向到同一台后端应用服务器，这样一来这个ip下的某个客户端和某个后端就能建立起稳固的session，ip\_hash是在upstream配置中定义的：

说明：因为这种方式只能用IP来分配后端，所以要求nginx一定要是最前端的服务器，否则nginx会取不到真实的客户端ip，那ip\_hash就失效了。例如在服务器架构中使用squid做前端高速缓存，那么nginx取到的就是squid服务器的ip，用这个ip来做ip\_hash肯定是不对的。再有，如果nginx的后端还有其他的负载均衡，将请求又分流了，那么对于某个客户端的请求，肯定不能定位到同一台应用服务器（例如php的fast-cgi服务器等），这样也不能做到session共享，如果在nginx后面再做负载均衡，我们可以再搭一台squid，然后再直接到应用服务器，或者用 location作一次分流，将需要session的部分请求通过ip\_hash分流，剩下的走其它后端。

# 06【掌握】动静分离

## 1，概述

    静动分离，就是将css、js、jpg等静态资源和jsp等动态资源分开处理，以此提高服务器响应速度，提高性能。

## 2，动静分离的配置

接上面的例子，我们使用的erp系统。查看erp里面的静态文件的路径可以看出

<link rel=*"stylesheet"* href=*"*${ctx }*/resources/layui/css/layui.css"* media=*"all"* />

因为项目的发布名为erp所有要按下面的规则来设置动静分离

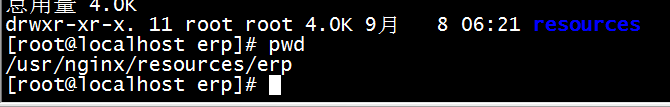
1，把tomcat1里面的webapps/erp/resources 拷贝到nginx/里面

#在nginx下创建resources目录

mkdir resources

#在resources下面创建erp目录

cp -r /root/software/tomcat1/webapps/erp/resources/ /usr/nginx/



2，修改conf/nginx.conf的server里面的配置

#动态分离的配置

location ~\* \.(css|js|html)$ {

root resources;#可以使用相对路径和绝对路径 /usr/resources就是绝对路径

expires 7d;#有效天数

}

​

location ~\* \.(avi|txt|png|gif|jpg|mp3|mp4|rmvb)$ {

root resources;

expires 20d;

}

## 3，动静分离的配置正则说明

http.server.localtion语法

​

基本语法： location [=|~|~\*|^~]/uri/{...}

= 严格匹配, 如果这个查询匹配，将停止搜索并立即处理此请求

~ 区分大小写匹配（可用正则表达式）

~\* 不区分大小写匹配（可用正则表达式）

!~ 区分大小写匹配

!~\* 不区分大小写匹配

^~ 如果把这个前缀用于一个常规字符串，那么告诉nginx如果路径匹配那么不测试正则表达式

​

location = /{

#只匹配/查询

}

​

location /{

#匹配任何查询，因为所有请求都以/开头。但是正则表达式和长的块规则将被优先查询匹配

}

​

location ^~ /images/{

#匹配任何以/images/开头的查询并停止搜索。任何正则表达式将不会被测试。

}

​

location ~\*.(gif|jpg|jpeg)${

#匹配任何以gif|jpg|jpeg结尾的请求

}