**Nginx**

## 一、Ngin概述

**1 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。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。C语言开发。

**2 nginx是一款自由的、开源的、高性能的HTTP服务器和反向代理服务器**；同时也是一个IMAP、POP3、SMTP代理服务器；nginx可以作为一个HTTP服务器进行网站的发布处理，另外nginx可以作为**反向代理、负载均衡、动静分离**的实现。

3 进入http://nginx.org/en/download.html 下载nginx1.8.0版本（当前最新稳定版本）。

4安装nginx依赖gcc环境，需要安装gcc：yum –y install gcc-c++

5 将nginx-1.8.0.tar.gz拷贝至linux服务器。

tar -zxvf nginx-1.8.0.tar.gz

6 启动nginx,**15098**是nginx主进程的进程id，**15099**是nginx工作进程的进程id

cd /usr/local/nginx/sbin/

./nginx -c /usr/local/nginx/conf/nginx.conf (不指定-c，默认加载conf/nginx.conf文件)

./nginx -s stop

./nginx -s reload

## 二、Nginx配置

**1、全局块：**配置影响nginx全局的指令。一般有运行nginx服务器的用户组，nginx进程pid存放路径，日志存放路径，配置文件引入，允许生成worker process数等。

**2、events块：**配置影响nginx服务器或与用户的网络连接。有每个进程的最大连接数，选取哪种事件驱动模型处理连接请求，是否允许同时接受多个网路连接，开启多个网络连接序列化等。

**3、http块：**可以嵌套多个server，配置代理，缓存，日志定义等绝大多数功能和第三方模块的配置。如文件引入，mime-type定义，日志自定义，是否使用sendfile传输文件，连接超时时间，单连接请求数等。

**4、server块：**配置虚拟主机的相关参数，一个http中可以有多个server。

**5、location块：**配置请求的路由，以及各种页面的处理情况。

#全局设置

main

# 运行用户

user www-data;

# 启动进程,通常设置成和cpu的数量相等

worker\_processes 1;

# 全局错误日志及PID文件

error\_log /var/log/nginx/error.log;

pid /var/run/nginx.pid;

# 工作模式及连接数上限

events {

use epoll; #epoll是多路复用IO(I/O Multiplexing)中的一种方式,但是仅用于linux2.6以上内核,可以大大提高nginx的性能

worker\_connections 1024; #单个后台worker process进程的最大并发链接数

# multi\_accept on;

}

#设定http服务器，利用它的反向代理功能提供负载均衡支持

http {

#设定mime类型,类型由mime.type文件定义

include /etc/nginx/mime.types;

default\_type application/octet-stream;

#设定日志格式

access\_log /var/log/nginx/access.log;

#sendfile 指令指定 nginx 是否调用 sendfile 函数（zero copy 方式）来输出文件，对于普通应用，

#必须设为 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uptime.

sendfile on;

#将tcp\_nopush和tcp\_nodelay两个指令设置为on用于防止网络阻塞

tcp\_nopush on;

tcp\_nodelay on;

#连接超时时间

keepalive\_timeout 65;

#开启gzip压缩

gzip on;

gzip\_disable "MSIE [1-6]\.(?!.\*SV1)";

#设定请求缓冲

client\_header\_buffer\_size 1k;

large\_client\_header\_buffers 4 4k;

include /etc/nginx/conf.d/\*.conf;

include /etc/nginx/sites-enabled/\*;

#设定负载均衡的服务器列表

upstream mysvr {

#weigth参数表示权值，权值越高被分配到的几率越大

#本机上的Squid开启3128端口

server 192.168.8.1:3128 weight=5;

server 192.168.8.2:80 weight=1;

server 192.168.8.3:80 weight=6;

}

server {

#侦听80端口

listen 80;

#定义使用www.xx.com访问

server\_name www.xx.com;

#设定本虚拟主机的访问日志

access\_log logs/www.xx.com.access.log main;

#默认请求

location / {

root /root; #定义服务器的默认网站根目录位置

index index.php index.html index.htm; #定义首页索引文件的名称

fastcgi\_pass www.xx.com;

fastcgi\_param SCRIPT\_FILENAME $document\_root/$fastcgi\_script\_name;

include /etc/nginx/fastcgi\_params;

}

# 定义错误提示页面

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

location = /50x.html {

root /root;

}

#静态文件，nginx自己处理

location ~ ^/(images|javascript|js|css|flash|media|static)/ {

root /var/www/virtual/htdocs;

#过期30天，静态文件不怎么更新，过期可以设大一点，如果频繁更新，则可以设置得小一点。

expires 30d;

}

#PHP 脚本请求全部转发到 FastCGI处理. 使用FastCGI默认配置.

location ~ \.php$ {

root /root;

fastcgi\_pass 127.0.0.1:9000;

fastcgi\_index index.php;

fastcgi\_param SCRIPT\_FILENAME /home/www/www$fastcgi\_script\_name;

include fastcgi\_params;

}

#设定查看Nginx状态的地址

location /NginxStatus {

stub\_status on;

access\_log on;

auth\_basic "NginxStatus";

auth\_basic\_user\_file conf/htpasswd;

}

#禁止访问 .htxxx 文件

location ~ /\.ht {

deny all;

}

}

#第一个虚拟服务器

server {

#侦听192.168.8.x的80端口

listen 80;

server\_name 192.168.8.x;

#对aspx后缀的进行负载均衡请求

location ~ .\*\.aspx$ {

root /root;#定义服务器的默认网站根目录位置

index index.php index.html index.htm;#定义首页索引文件的名称

proxy\_pass http://mysvr;#请求转向mysvr 定义的服务器列表

#以下是一些反向代理的配置可删除.

proxy\_redirect off;

#后端的Web服务器可以通过X-Forwarded-For获取用户真实IP

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; #允许客户端请求的最大单文件字节数

client\_body\_buffer\_size 128k; #缓冲区代理缓冲用户端请求的最大字节数，

proxy\_connect\_timeout 90; #nginx跟后端服务器连接超时时间(代理连接超时)

proxy\_send\_timeout 90; #后端服务器数据回传时间(代理发送超时)

proxy\_read\_timeout 90; #连接成功后，后端服务器响应时间(代理接收超时)

proxy\_buffer\_size 4k; #设置代理服务器（nginx）保存用户头信息的缓冲区大小

proxy\_buffers 4 32k; #proxy\_buffers缓冲区，网页平均在32k以下的话，这样设置

proxy\_busy\_buffers\_size 64k; #高负荷下缓冲大小（proxy\_buffers\*2）

proxy\_temp\_file\_write\_size 64k; #设定缓存文件夹大小，大于这个值，将从upstream服务器传

}

}

}

http {

    include       mime.types;

    default\_type  application/octet-stream;

     sendfile        on；

    keepalive\_timeout  65;

    proxy\_cache\_path /cache/nginx/ levels=1:1 keys\_zone=mycache:23m;

     #建立缓存目录，目录必须是nginx用户，levels是可建目录几级，mycache是名称，开启缓存，只能在http中开启，具体功能可在http,server,location,if中使用

        fastcgi\_cache\_path /cache/fastcgi levels=1:1 keys\_zone=fcgicache:10m inactive=3m max\_size=1g;

    #factcgi接口使用的命令，建立fastcgi的缓存

       upstream upservers {

               #建立后端负载均衡技术

                #ip\_hash;负载均衡中的指令，可不加

                server 192.168.1.11 max\_fails=2 fail\_timeout=1 weight=2;

                 #添加一台服务器，max\_fails失败两次即暂停，fail\_timeout 失败后暂停的时间,weigth权重

                server 192.168.1.12  max\_fails=2 fail\_timeout=1;

                  #后加down即人为停掉这台服务

                  }

 server {

        listen       80;

        server\_name  localhost;

        add\_header X-Via $server\_addr;   # 可以放在任意里面，首部自定义添加内容

        add\_header X-cache $upstream\_cache\_status;    #根据是否缓存命中给出状态，如miss，hit

       location / {

           proxy\_pass http://upservers/;

              #按组的形式，把后端已经作成组的服务器加进来

        }

        location /root {

                proxy\_pass http://192.168.1.11/root/;   #代理后端服务器

                proxy\_set\_header Host $remote\_addr;   #日志会记录源ip的日志，而不是代理服务器请求的日志，还需要修改后端服务器的日志格式logFormat "%{X-Real-IP}i"

                proxy\_cache mycache;  #使用mycahce缓存，http中定义

                proxy\_cache\_valid 200 1d; #根据响应码确定缓存市场，可多次定义

                proxy\_cache\_valid 301 10m;

                proxy\_cache\_valid any 1m; #其他未定的一律1m缓存

                proxy\_cache\_use\_stale error timeout invalid\_header http\_500 http\_502 http\_504; #缓存过期后，哪些情况可以继续使用缓存，以上缓存命令在其他location 中可继续使用

        }

        location /forum/ { #此处可以用其他名字，可以不同

                        proxy\_pass http://192.179.1.11/bbs/;  #此处必须加上/bbs/

                      }

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

                          proxy\_pass http://192.168.1.12;  #因为是正则表达式，所以不能加，表示所有访问图片类型的都走这台服务器

                        }

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

        location = /50x.html {

            root   html;

        }

        location ~ \.php$ {

               #动静分离，动态页面全走这里

                #注意此处开启php动态功能，需要安装php-fpm,并重启服务

                #如果想用mysql，可安装php-msyql,  mysql-server,并重启服务即可

           root  /usr/share/nginx/html;

           fastcgi\_pass   127.0.0.1:9000;

           fastcgi\_index  index.php;

            fastcgi\_param  SCRIPT\_FILENAME  /scripts$fastcgi\_script\_name;

           include        fastcgi\_params;

        }

}

## 三、Nginx反向代理

**1 正向代理：**通常的代理服务器，用于代理内部网络对Internet的连接请求，客户机必须指定代理服务器,并将本来要直接发送到Web服务器上的http请求发送到代理服务器中由代理服务器向Internet上的web服务器发起请求，最终达到客户机上网的目的。

**2 反向代理：**以代理服务器来接受internet上的连接请求，然后将请求转发给内部网络上的服务器，并将从服务器上得到的结果返回给internet上请求连接的客户端，此时代理服务器对外就表现为一个反向代理服务器。

通过上述的图解大家就可以看清楚了，多个客户端给服务器发送的请求，nginx服务器接收到之后，按照一定的规则分发给了后端的业务处理服务器进行处理了。此时~请求的来源也就是客户端是明确的，但是请求具体由哪台服务器处理的并不明确了，nginx扮演的就是一个反向代理角色。

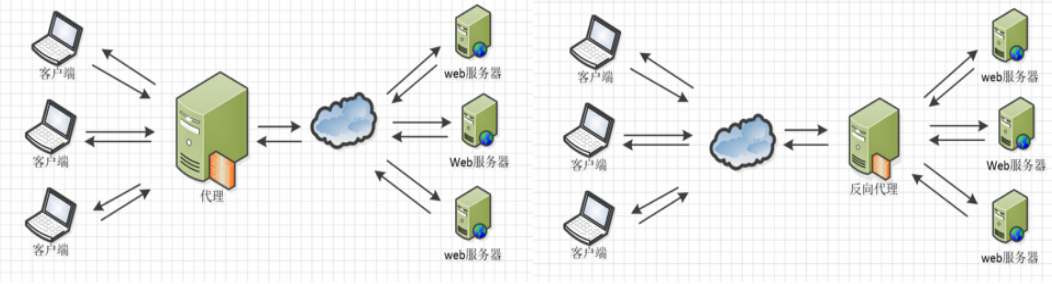
**反向代理，主要用于服务器集群分布式部署的情况下，反向代理隐藏了服务器的信息！当然反向代理的另一大用处就是隐藏后面的实际服务，以此来达到一定的安全性。**

**正向代理：**

**client —(send request)—> clinet proxy –(send request)—> server**

**反向代理：**

**clinet –(send request)–> server proxy –(send request)–>other server**



**3 两个tomcat服务通过nginx反向代理，本例子使用三台虚拟机进行测试，**

nginx服务器：192.168.101.3

tomcat1服务器：192.168.101.5

tomcat2服务器：192.168.101.6

#配置一个代理即tomcat1服务器

upstream tomcat\_server1 {

server 192.168.101.5:8080;

}

#配置一个代理即tomcat2服务器

upstream tomcat\_server2 {

server 192.168.101.6:8080;

}

#配置一个虚拟主机

server {

listen 80;

server\_name aaa.test.com;

location / {

#域名aaa.test.com的请求全部转发到tomcat\_server1即tomcat1服务上

proxy\_pass http://tomcat\_server1;

#欢迎页面，按照从左到右的顺序查找页面

index index.jsp index.html index.htm;

}

}

server {

listen 80;

server\_name bbb.test.com;

location / {

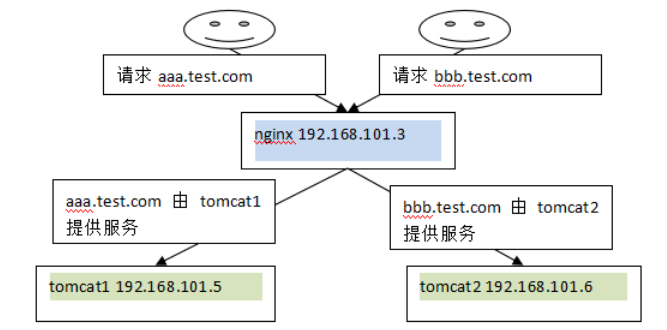
#域名bbb.test.com的请求全部转发到tomcat\_server2上

proxy\_pass http://tomcat\_server2;

index index.jsp index.html index.htm;

}

}



## 四、Nginx负载均衡

**1 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。

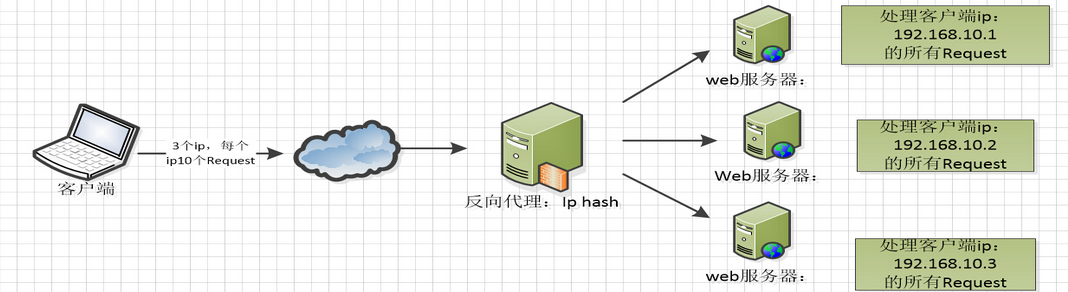
**2 种方式的分配**

**@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。 **@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。

**@ip\_hash/url\_ip** 每个请求按访问ip或者url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。

**@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。



**3 负载均衡配置**

upstream wade

{

server 10.0.1.50:8080;

server 10.0.1.51:8080;

}

upstream james

{

server 10.0.1.50:8080;

server 10.0.1.51:8080;

}

server

{

listen 80;

server\_name www.linuxidc.com;

location / {

proxy\_pass http://wade;

proxy\_set\_header Host $host;

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

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

}

}

server

{

listen 80;

server\_name blog.linuxidc.com wode.linuxidc.com;

location / {

proxy\_pass http://james;

proxy\_set\_header Host $host;

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

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

}

}

## 五、Nginx高可用

**1 使用集群是网站解决高并发、海量数据问题的常用手段。**当一台服务器的处理能力、存储空间不足时，不要企图去换更强大的服务器，对大型网站而言，不管多么强大的服务器，都满足不了网站持续增长的业务需求。这种情况下，更恰当的做法是增加一台服务器分担原有服务器的访问及存储压力。通过负载均衡调度服务器，将来自浏览器的访问请求分发到应用服务器集群中的任何一台服务器上，如果有更多的用户，就在集群中加入更多的应用服务器，使应用服务器的负载压力不再成为整个网站的瓶颈。

**2 Keepalived软件起初是专为LVS负载均衡软件设计的**，用来管理并监控LVS集群系统中各个服务节点的状态，后来又加入了可以实现高可用的VRRP功能。因此，Keepalived除了能够管理LVS软件外，**还可以作为其他服务（例如：Nginx、Haproxy、MySQL等）的高可用解决方案软件**。

Keepalived软件主要是通过VRRP协议实现高可用功能的。VRRP是Virtual Router RedundancyProtocol(虚拟路由器冗余协议）的缩写，VRRP出现的目的就是为了**解决静态路由单点故障问题的，它能够保证当个别节点宕机时，整个网络可以不间断地运行**。

**3 VRRP ,全 称 Virtual Router Redundancy Protocol ,中文名为虚拟路由冗余协议 ，VRRP的出现就是为了解决静态踣甶的单点故障问题，VRRP是通过一种竞选机制来将路由的任务交给某台VRRP路由器的。**

**4 节点信息**

**节点 服务 安装目录**

192.168.100.10:80 nginx1 /usr/local/nginx

192.168.100.11:80 nginx2 /usr/local/nginx

192.168.100.100 keepalived1 /usr/local/keepalived

192.168.100.101 keepalived2 /usr/local/keepalived

192.168.100.10:8080 tomcat1 /usr/local/nginx/tomcat/tomcat1

192.168.100.10:8081 tomcat2 /usr/local/nginx/tomcat/tomcat2

192.168.100.11:8080 tomcat3 /usr/local/nginx/tomcat/tomcat1

192.168.100.11:8081 tomcat4 /usr/local/nginx/tomcat/tomcat2

**5 Tomcat端口分配**

**节点 SHUTDOWN(端口) HTTP/1.1(端口) AJP/1.3(端口)**

192.168.66.100:8080 8005 8081 8009

192.168.100.11:8081 8006 8082 8010

192.168.100.11:8080 8005 8081 8009

192.168.100.11:8081 8006 8082 8010

**6 keepalived.conf配置**

#修改配置

vim /etc/keepalived/keepalived.conf

#配置文件

! Configuration File for keepalived

global\_defs {

#从服务器最改一下这个

router\_id node1 #表示运行keepalived服务器的一个标识。发邮件时显示在邮件主题的信息

}#监控服务.NGINX mysql等

vrrp\_script chk\_nginx {

script "/usr/local/keepalived/check\_nginx.sh"

#每2s检查一次

interval 2

#每次检查-20

weight -20

}

vrrp\_instance VI\_1 {

##主从设置 MASTER/BACKUP

state MASTER

#网卡名称

interface eth0

#同一个集群下这个 router\_id是一样的

virtual\_router\_id 51

#本机的ip，需要修改

mcast\_src\_ip 192.168.100.10

#优先级，从节点 配置，需要小于主节点

priority 100

#设定MASTER与BACKUP负载均衡器之间同步检查的时间间隔，单位是秒

advert\_int 1

#认证的密码

authentication {

auth\_type PASS

#设定授权密码，密码相同的为一个集群

auth\_pass yellowcong

}

#触发的脚本

track\_script {

chk\_nginx #检测脚本，上面配置的

}

#虚拟ip地址

virtual\_ipaddress {

192.168.100.100

192.168.100.101

}

}

**7 check\_nginx.sh配置**

touch check\_nginx.sh && chmod a+x check\_nginx.sh

#编辑文件

vim check\_nginx.sh

#################下面是内容#####################!/bin/bash

COUNT=$(ps -C nginx --no-header |wc -l)

echo $COUNT

#判断Nginx 是否都挂掉了if [ $COUNT -eq 0 ]then

#如果挂掉了，就启动nginx

/usr/local/nginx/sbin/nginx

echo "重启nginx"

#等5秒钟后，再次查看是否 启动成功

sleep 5

#如果nginx没有启动起来，就直接干掉keepalived

COUNT=$(ps -C nginx --no-header |wc -l)

if [ $COUNT -eq 0 ]

then

echo "干掉keepalived"

#如果killall命令不能使用，就需要安装psmisc工具了

#yum install -y psmisc

killall keepalived

fifi

**8 nginx.conf配置**

http {

include mime.types;

default\_type application/octet-stream;

sendfile on;

keepalive\_timeout 65;

#负载均衡

upstream myapp{

server 192.168.100.11:8080 weight=1 max\_fails=2 fail\_timeout=30s;

server 192.168.100.11:8081 weight=1 max\_fails=2 fail\_timeout=30s;

server 192.168.100.10:8081 weight=1 max\_fails=2 fail\_timeout=30s;

server 192.168.100.10:8081 weight=1 max\_fails=2 fail\_timeout=30s;

}

server {

listen 80;

server\_name localhost;

location / {

proxy\_set\_header X-real-ip $remote\_addr;

proxy\_pass http://myapp;

}

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

location = /50x.html {

root html;

}

}

}

## 六、Nginx web缓存

**1 导入配置文件**

include vhosts/dev-a/\*.conf;

include vhosts/dev-b/\*.conf;

include vhosts/test-a/\*.conf;

include vhosts/test-b/\*.conf;

include vhosts/hotfix/\*.conf;

**2 Web缓存使用**

 upstream webserver {

server 192.168.0.201 weight=1 max\_fails=2 fail\_timeout=2;

server 192.168.0.202 weight=1 max\_fails=2 fail\_timeout=2;

}

server {

listen 9008;

server\_name localhost;

root html/error;

index index.html;

}  
  
proxy\_cache\_path /nginx/cache/first levels=1:2 keys\_zone=first:20m max\_size=1g; #缓存目录不能定义在server块内，要定义在http块中  
#/nginx/cache/first定义缓存目录参数

#evels=1:2 定义两层目录，第一层一个字符名称，第二个两个字符名称

#keys\_zone=first:20m 每个缓存都是一个共享内存空间。这就是用户定义共享内存空间地址的名称

#max\_size=1g 定义目录最大空间为1g，因为缓存空间越大搜索数据越慢，因此不宜太大。

server {

server\_name hfnginx.chinacloudapp.cn;

location / {

add\_header X\_Via $server\_addr; #添加服务器地址到报文头部

add\_header X-Cache $upstream\_cache\_status; #添加缓存状态到报文头部

proxy\_pass http://webserver;

proxy\_cache first; #调用缓存

proxy\_cache\_valid 200 10m; #定义缓存失效时间，200是状态码，即缓存状态码是200请求成功的数据，10m是10分钟，即缓存的数据的超时时间10分钟，10分钟后即过期，不定义则缓存不生效

}

location ~\* ^/form {

proxy\_cache cache\_one;

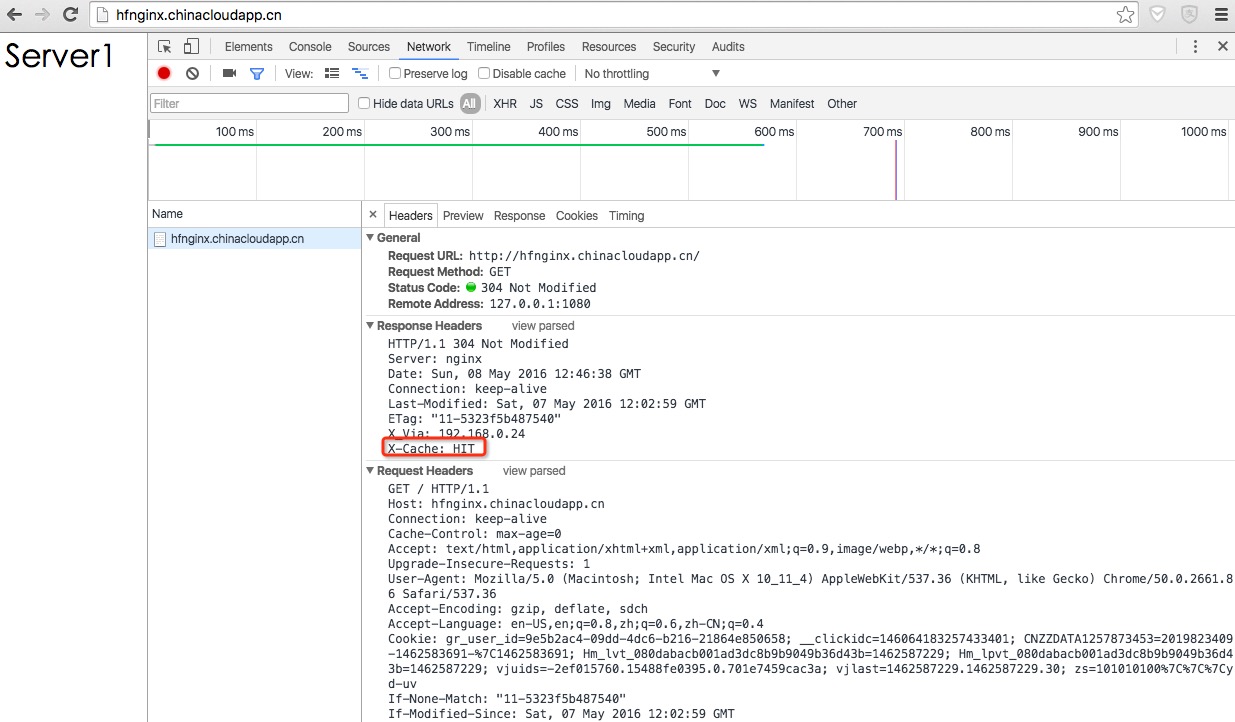
proxy\_pass http://webserver;

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

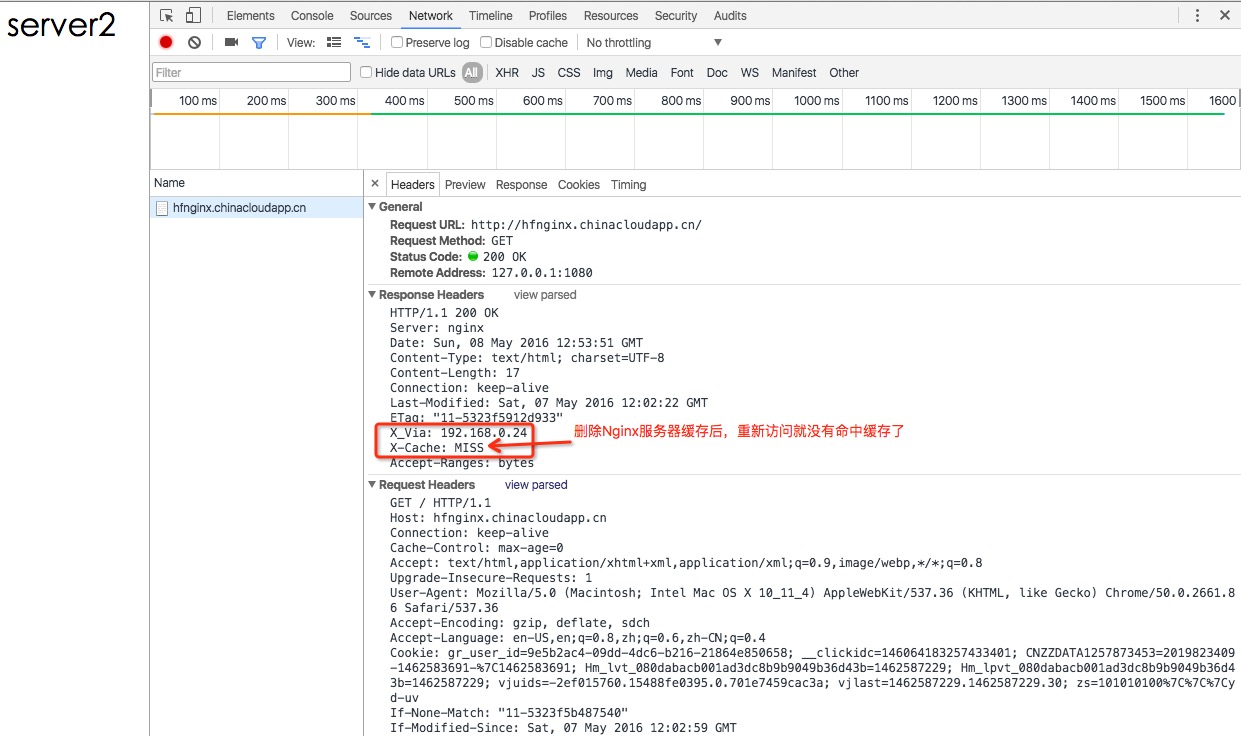
}

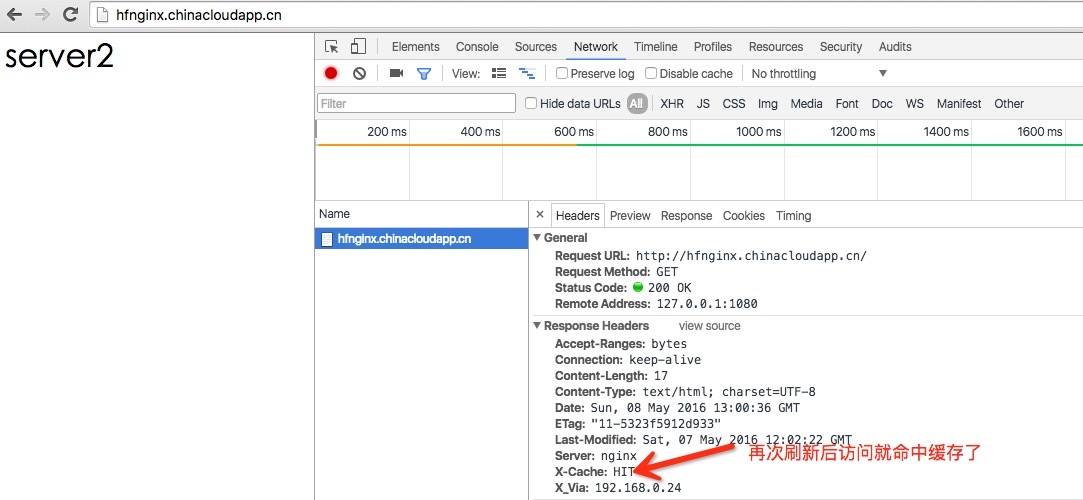
}

**缓存是缓存nginx服务器接收请求过的数据，数据超时时间不能太长，因为数据可能会发生变化，但是nginx服务器内部的缓存的数据还没有更细，会导致客户端请求的数据不是最新数据的问题，数据缓存目录不能定义在server快内，要定义在http块中**



**注：X\_Via返回的响应客户端请求报文的服务器，将有Nginx构建报文响应客户端请求，所以显示的是Nginx服务器的IP地址，X-Cache标记是否缓存，HIT是缓存过的数据，MISS是没有缓存的数据。**





**3 动静分离**

upstream web {

server 192.168.0.1 weight=1 max\_fails=2 fail\_timeout=2;

server 192.168.0.2 weight=1 max\_fails=2 fail\_timeout=2;

}

upstream image {

server 192.168.0.3 weight=1 max\_fails=2 fail\_timeout=2;

server 192.168.0.4 weight=1 max\_fails=2 fail\_timeout=2;

}

upstream php {

server 192.168.0.5 weight=1 max\_fails=2 fail\_timeout=2;

server 192.168.0.6 weight=1 max\_fails=2 fail\_timeout=2;

}

location /{

root html/web;

index index.php index.html;

}

location ~\* \.php$ {

fastcgi\_proxy http://php;}

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

proxy\_pass http://image;

}

**4 拦截upload请求到本地硬盘**

server {

listen 80;

server\_name erp.zb25.com.cn;

access\_log logs/dev.log main;

error\_log logs/dev-error.log info;

location /upload {

alias /home/test/data/apache\_new/opserp/upload/;

}

location / {

proxy\_buffering off;

proxy\_pass http://10.10.50.156:8030;

}

}