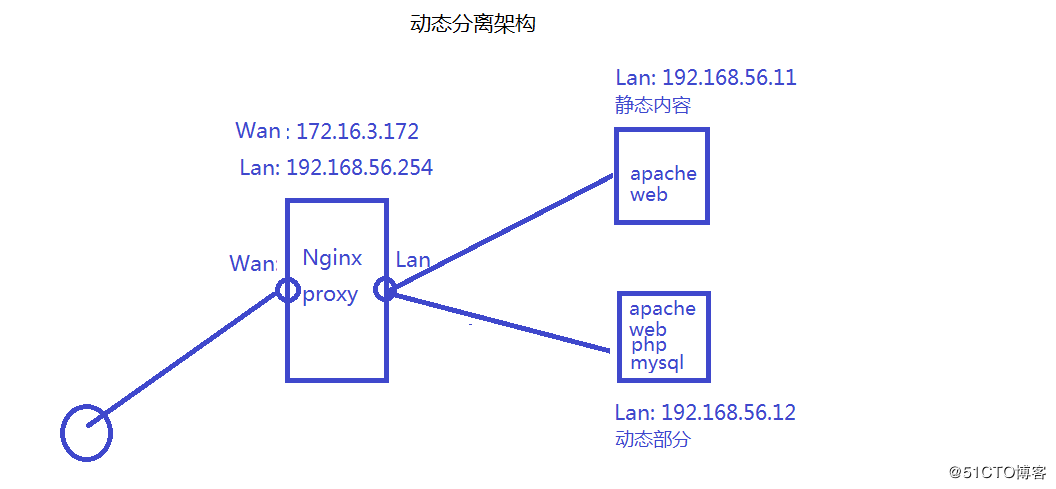
**一、概述:**

我之前有一篇文章写了[Nginx作为web服务器的http与https的初探](http://blog.51cto.com/dyc2005/2053027)作为nginx基础介绍,作为web服务器使用;今天要介绍的是nginx作为代理服务器和七层调度负载均衡的入门介绍;  
**实现内容:**  
通过nginx代理后端phpadmin网站,并通过nginx的代理达到动静内容分离;在代理上做缓存;

**实验环境:**  
proxy:Centos7 模拟外网ip:172.16.3.152 内网Lan ip:192.168.56.254  
后端静态节点n1.pkey.cn:  
CentOS7 内网Lan ip:192.168.56.11  
后端静态节点n2.pkey.cn:  
CentOS7 内网Lan ip:192.168.56.12  
测试客户端:ubuntu 16.04 ip:172.16.3.140

**动静分离架构图**  


**二、ngin代理之动静分离**

这回我们动态内容是php php-fpm mariadb phpadmin  
动态web 的配置  
**1、软件安装配置**

*#软件安装*

[root@n2 ~]*# yum install php php-fpm php-mysql php-mbstring php-mcrypt mariadb-server -y*

*#配置php-fpm*

[root@n2 ~] cat /etc/php-fpm.d/www.conf

[www]

listen = 0.0.0.0:9000

listen.allowed\_clients = any

user = apache

group = apache

pm = dynamic

pm.max\_children = 150

pm.start\_servers = 5

pm.min\_spare\_servers = 5

pm.max\_spare\_servers = 35

pm.status\_path = /status

ping.path = /ping

slowlog = /var/log/php-fpm/www-slow.log

php\_admin\_value[error\_log] = /var/log/php-fpm/www-error.log

php\_admin\_flag[log\_errors] = on

php\_value[session.save\_handler] = files

php\_value[session.save\_path] = /var/lib/php/session

*#创建对应的php session目录*

[root@n2 ~]*# mkdir /var/lib/php/session -pv*

[root@n2 ~]*# chown apache:apache /var/lib/php/session*

**2、安装phpadmin**  
[root@n2 ~]# wget <https://files.phpmyadmin.net/phpMyAdmin/4.0.10.20/phpMyAdmin-4.0.10.20-all-languages.tar.gz>  
[root@n2 ~]# mkdir -pv /data/apps  
[root@n2 ~]# tar xvf phpMyAdmin-4.0.10.20-all-languages.tar.gz -C /data/apps/  
[root@n2 apps]# ln -sv phpMyAdmin-4.0.10.20-all-languages/ pma  
[root@n2 pma]# cp config.sample.inc.php config.inc.php  
#只需要修改17行  
$cfg['blowfish\_secret'] = 'a8bdafda7c6d'; /

**3、数据库配置**

[root@n2 ~]*# vim /etc/my.cnf.d/server.cnf*

添加以下两行

[mysqld]

skip\_name\_resolve=ON

innodb\_file\_per\_table=ON

启动数据库

[root@n2 ~]*# systemctl restart mariadb*

[root@n2 ~]*# systemctl enable mariadb*

*#设置root密码*

MariaDB [(none)]> set password **for** root@localhost=password("redhat");

Query OK, 0 rows affected (0.00 sec)

**4、静态web配置**

[root@n1 ~]*# yum install nginx -y*

[root@n1 ~]*# mkdir -pv /data/nginx/html*

[root@n1 ~]*# cd /data/nginx/html*

[root@n2 ~]*# wget https://files.phpmyadmin.net/phpMyAdmin/4.0.10.20/phpMyAdmin-4.0.10.20-all-languages.tar.gz*

[root@n1 ~]*# tar xvf phpMyAdmin-4.0.10.20-all-languages.tar.gz -C /data/nginx/html*

[root@n1 ~]*# cd /data/nginx/html*

[root@n1 ~]*# ln -sv phpMyAdmin-4.0.10.20-all-languages/ pma*

[root@n2 pma]*# cp config.sample.inc.php config.inc.php*

*#只需要修改17行*

$cfg['blowfish\_secret'] = 'a8bdafda7c6d'; / *#可与动态上的不一样内容*

*#nginx配置*

cat /etc/nginx/conf.d/phpadmin.conf

server{

listen 80;

server\_name 192.168.56.11;

root /data/apps;

}

静态web中的作用就是提供phpadmin的静态内容 ,如html和图片等静态内容~

**5、nginx proxy配置**  
安装nginx并配置代理

[root@proxy ~]*# yum install nginx -y*

cat /etc/nginx/conf.d/phpadm.conf

server{

**listen** 80;

server\_name phpadmin.pkey.cn;

**index** index.php index.html;

location / {

root /data/nginx/html;

proxy\_pass http://192.168.56.11:80; *#静态反代到n1*

}

location ~\* \.php$ {

fastcgi\_pass 192.168.56.12:9000; *#php动态反代到n2*

fastcgi\_index index.php;

include fastcgi\_params;

fastcgi\_param SCRIPT\_FILENAME /data/apps/$fastcgi\_script\_name;

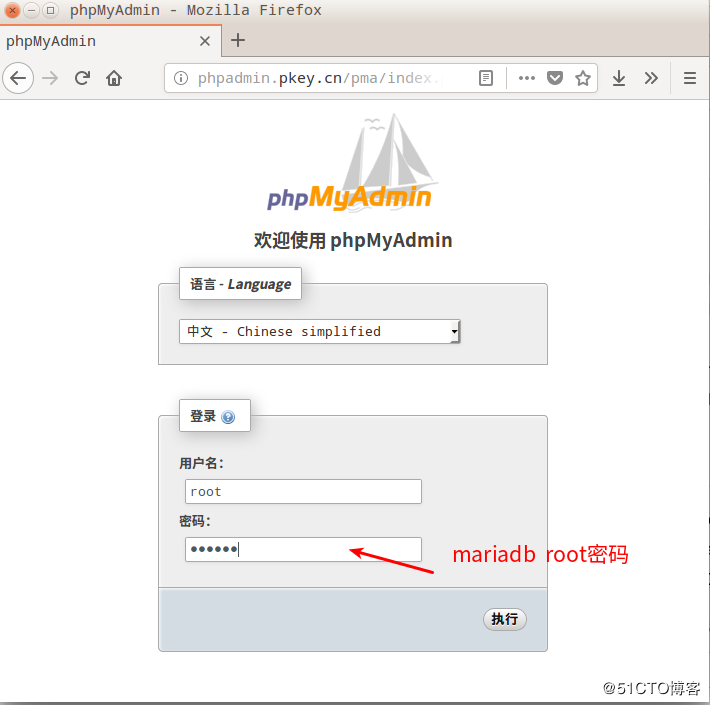
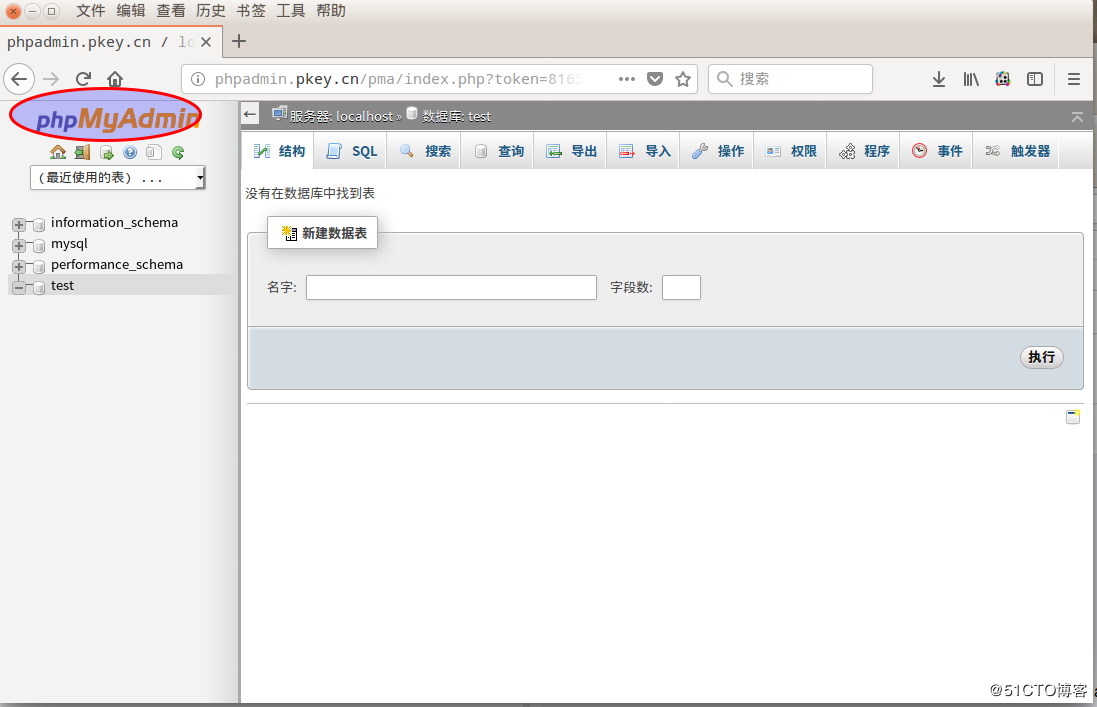
}

}

*##检查并启动nginx服务*

[root@proxy ~]*# nginx -t*

[root@proxy ~]*# systemctl start nginx*

以上配置说明了:凡是php结尾的动态请求都发往192.168.56.12:9000处理;其他的则发往192.168.56.11:80来处理;  
我的测试 是在ubuntu下的的  
添加172.16.3.172 phpadmin.pkey.cn 条目到/etc/hosts中  
打开浏览器访问:<http://phpadmin.pkey.cn/pma/index.php> 如图:  
  
  
以上的两个网页中图片静态资源也能正常访问,我们检验下静态的图片是不是从n1的网站提供的;  
到n1.pkey.cn静态web上看日志 :

[root@n1 conf.d]*# tail -f /var/log/nginx/access.log*

192.168.56.254 - - [11/Jan/2018:02:58:11 -0500] "GET /pma/themes/dot.gif HTTP/1.0" 200 43 "-" "Mozilla/5.0 (X11; Ubuntu; Linux x86\_64; rv:57.0) Gecko/20100101 Firefox/57.0" "-"

192.168.56.254 - - [11/Jan/2018:02:58:11 -0500] "GET /pma/themes/pmahomme/img/ajax\_clock\_small.gif HTTP/1.0" 200 1810 "-" "Mozilla/5.0 (X11; Ubuntu; Linux x86\_64; rv:57.0) Gecko/20100101 Firefox/57.0" "-"

.......以下省略.......

可以看在刷新http://phpadmin.pkey.nc/pma/index.php时日志上都是"GET /pma/themes/dot.gif"类型的图片文件访问记录另外看到日志上的访问是代理的内网ip即:192.168.56.254,如何给出真实请求ip?

**6、代理转发客户端请求真实ip**  
需要在nginx proxy上添加如下信息:

location / {

root /data/nginx/html;

proxy\_pass http://192.168.56.11:80; *#静态反代到n1*

proxy\_set\_header X-Forwarded-For $remote\_addr;

}

重新加载nginx  
再次访问http://phpadmin.pkey.cn/pma/index.php  
查看n1上的日志信息:

[root@n1 nginx]*# tail -f access.log*

192.168.56.254 - - [11/Jan/2018:03:23:35 -0500] "GET /pma/themes/dot.gif HTTP/1.0" 200 43 "-" "Mozilla/5.0 (X11; Ubuntu; Linux x86\_64; rv:57.0) Gecko/20100101 Firefox/57.0" "172.16.3.140"

192.168.56.254 - - [11/Jan/2018:03:23:35 -0500] "GET /pma/themes/pmahomme/img/sprites.png HTTP/1.0" 200 61899 "-" "Mozilla/5.0 (X11; Ubuntu; Linux x86\_64; rv:57.0) Gecko/20100101 Firefox/57.0" "172.16.3.140"

192.168.56.254 - - [11/Jan/2018:03:25:40 -0500] "GET /pma/themes/pmahomme/jquery/jquery-ui-1.9.2.custom.css HTTP/1.0" 404 169 "-" "Mozilla/5.0 (X11; Ubuntu; Linux x86\_64; rv:57.0) Gecko/20100101 Firefox/57.0" "172.16.3.140"

192.168.56.254 - - [11/Jan/2018:03:25:40 -0500] "GET /pma/themes/pmahomme/img/logo\_right.png HTTP/1.0" 404 169 "-" "Mozilla/5.0 (X11; Ubuntu; Linux x86\_64; rv:57.0) Gecko/20100101 Firefox/57.0" "172.16.3.140"

192.168.56.254 - - [11/Jan/2018:03:25:40 -0500] "GET /pma/themes/dot.gif HTTP/1.0" 404 169 "-" "Mozilla/5.0 (X11; Ubuntu; Linux x86\_64; rv:57.0) Gecko/20100101 Firefox/57.0" "172.16.3.140"

可以看到日志每条的最后面都有172.16.3.140 这个地址就是我的测试机器的ip;

**7、给代理服务器加上缓存**  
a、针对静态内容的cache,由ngx\_http\_proxy\_module模块中的 proxy\_cache\_path 指令实现,必须放在http区域中  
以下是应用实例:

[root@proxy ~]*# cat /etc/nginx/nginx.conf*

...

http{

...

proxy\_cache\_path /data/nginx/cache levels=1:2 keys\_zone=scache:10m inactive=3m max\_size=2g;

...

}

[root@proxy ~]*# mkdir -pv /data/nginx*

proxy\_cache\_path参数说明:  
/data/nginx/cache #缓存路径  
levels=1:2 #路径的结构  
keys\_zone=scache:10m #键区域名及大小  
inactive=3m #活动时间  
max\_size=2g #使用2g硬盘做缓存  
更详细使用说明了看[官方文档](http://nginx.org/en/docs/http/ngx_http_proxy_module.html#proxy_cache_path)

b、php-fpm fastcgi缓存配置  
针对动态的php 缓存配置同样需要配置在http区域中如下:

[root@proxy ~]*# cat /etc/nginx/nginx.conf*

...

http{

...

proxy\_cache\_path /data/nginx/cache levels=1:2 keys\_zone=scache:10m inactive=3m max\_size=2g;

fastcgi\_cache\_path /data/nginx/fcgicache levels=2:1 keys\_zone=fcache:10m max\_size=2g;

...

}

[root@proxy ~]*# mkdir -pv /data/nginx*

这里的缓存名叫fcache  
参数意义与静态的意义类似更多的使用说明请看[官方文档](http://nginx.org/en/docs/http/ngx_http_fastcgi_module.html" \l "fastcgi_cache_path)

最终proxy 上phpadmin.conf配置文件如下:

[root@proxy ~]*#cache /etc/nginx/conf.d/phpadmin.conf*

server{

**listen** 80;

server\_name phpadmin.pkey.cn;

**index** index.php index.html;

location / {

proxy\_pass http://192.168.56.11:80; *#静态反代到n1*

proxy\_set\_header X-Forwarded-For $remote\_addr;

proxy\_cache scache; *#启用定义缓存*

proxy\_cache\_valid 200 302 10**m**; *#各状态缓存时间*

proxy\_cache\_valid 301 1h;

proxy\_cache\_valid any 1**m**;

}

location ~\* \.php$ {

fastcgi\_pass 192.168.56.12:9000; *#php动态反代到n2*

fastcgi\_index index.php;

include fastcgi\_params;

fastcgi\_param SCRIPT\_FILENAME /data/apps/$fastcgi\_script\_name;

fastcgi\_cache fcache; *#启用定义好的缓存*

fastcgi\_cache\_key $request\_uri;

fastcgi\_cache\_valid 200 302 10**m**; *#各状态缓存时间*

fastcgi\_cache\_valid 301 1h;

fastcgi\_cache\_valid any 1**m**;

}

}

*#检查配置并重启nginx服务*

[root@proxy ~]*# nginx -t*

[root@proxy ~]*# systemctl restart nginx*

再次请求访问http://phpadmin.pkey.cn/pma/index.php

查看缓存:

[root@proxy nginx]*# cd cache/*

[root@proxy cache]*# ls*

2 6 9 d e

[root@proxy cache]*# cd ../fcgicache/*

[root@proxy fcgicache]*# ls -alh*

总用量 0

drwx------ 4 nginx root 24 1月 11 17:05 .

drwxr-xr-x 4 root root 34 1月 11 16:53 ..

drwx------ 3 nginx nginx 14 1月 11 17:05 d4

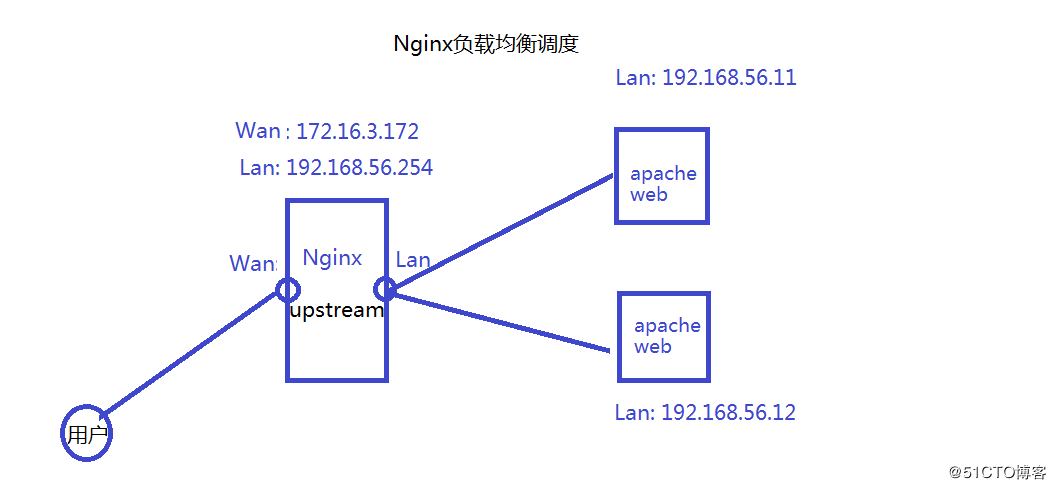
drwx------ 3 nginx nginx 14 1月 11 17:05 d7

可以看出访问的内容已经被缓存;以上就是动静分离并分别缓存的实例!  
可以自行在其他客户端上用ab工具测试,使用与不使用缓存访问的效率与速度   
ab -c 100 -n 2000 <http://phpadmin.pkey.cn/pma/index.php>  
这里就不贴过程了,效果还是很明显的!

**三、nginx负载均衡实现**

**1、实现内容:**  
nginx通过ngx\_http\_upstream\_module模块实现(七层)负载均衡向后端调度主机;可以定义权重,自动检测后端主机健康状态;通过nginx实现负载均衡访问后端服务,两台后端服务提供一致内容;这里是测试 所以人为让两台服务提供不一样的内容以示区别;

2、**实验环境:**  
upstream:Centos7 模拟外网ip:172.16.3.152 内网Lan ip:192.168.56.254  
后端节点n1.pkey.cn:  
CentOS7 内网Lan ip:192.168.56.11  
后端节点n2.pkey.cn:  
CentOS7 内网Lan ip:192.168.56.12  
测试客户端:ubuntu 16.04 ip:172.16.3.140

**3、实现架构**  


**4、两台后台主机配置**

两台upstream后端主机

**#**yum install httpd -y

分别

**#**cat /var/www/html/index.html

<h1>Upstream Server 1</h1>

<h1>Upstream Server 2</h1>

分别重启两台httpd

**#**systemctl restart httpd

**5、nginx调度器配置**  
同样也是要配置在http区域,以下是配置

[root@upstream ~]*# cat /etc/nginx/nginx.conf*

http{

...

upstream websrvs { *#定义调度 group*

server 192.168.56.11:80;

server 192.168.56.12:80;

}

...省略...

}

具体网站负载配置文件

cat /etc/nginx/conf.d/pkey.cn.conf

server{

listen 80;

server\_name www.pkey.cn;

location /{

root /data/nginx/html;

proxy\_pass http://websrvs;*#引用upstream group*

}

}

测试 :  
以ubuntu测试机上的/etc/hosts中添加  
172.16.3.152 www.pkey.cn

san@ubuntu:~$ for i in {1..10};do curl http://www.pkey.cn;done

<h1>Upstream Server 1</h1>

<h1>Upstream Server 2</h1>

<h1>Upstream Server 1</h1>

<h1>Upstream Server 2</h1>

<h1>Upstream Server 1</h1>

<h1>Upstream Server 2</h1>

<h1>Upstream Server 1</h1>

<h1>Upstream Server 2</h1>

<h1>Upstream Server 1</h1>

<h1>Upstream Server 2</h1>

从测试结果看默认权重一样的,是轮循高度算法;

5、upstream 中重要的参数说明

upstream websrvs { *#group*

ip\_hash;

server 192.168.10.11:80 fail\_timeout=1 max\_fails=3;

server 192.168.10.12:80 weight=2;

server 127.0.0.1:80 backup;

keepalive 32;

}

#ip\_hash 基于访问客户端ip来始终调度到同一个后端服务器;类似于lvs中的sh算法  
#fail\_timeout =1 max\_fails=3 表示超时1s 最多3次 此台主机失效  
#weight=2 权重为2 默认权重为1  
#backup 表示此台主机在所有其他主机都岩机不可访问时 上线,一般用来做临时维护页  
#keepalive 表示调度代理与后端主机每个worker保持多少个连接数  
#down 可以人为的让某主机下线,不参与调度  
更多配置参见[官方文档](http://nginx.org/en/docs/http/ngx_http_upstream_module.html#ip_hash)另外有些功能是在企业版中才有的!

**总结:**

本文对nginx做代理服务器时使用proxy\_pass指令;使用phpamin项目实现动静分离;并分别使用缓存提交访问效率;  
结合http模块下的upstream模块来实现对后端服务器的调度,达到七层负载均衡作用;之所以说是七层负载均衡调度方案,因为最新的nginx版本(1.9.2之后)有单独的stream模块可实现类似四层调度负载方案;后期补上!以上不免有遗漏之处,如有错误或遗漏之处,欢迎指正交流~谢谢