operation day 04

================================================

**继续优化nginx与处理相关问题**

**3，配置nginx的数据缓存，一台服务器的相同数据可能会被同一个客户反复访问，为了不重复让服务器给客户传递相同数据，达到节约资源、节省时间的目的，我们可以进行以下优化配置。**

修改配置文件 在默认的location的下面添加一个新的location

location ~\* \.(jpg|txt|html|png)$ { //当用户访问的是这几种类型的文件

expires 30d; //都会缓存在客户机上30天

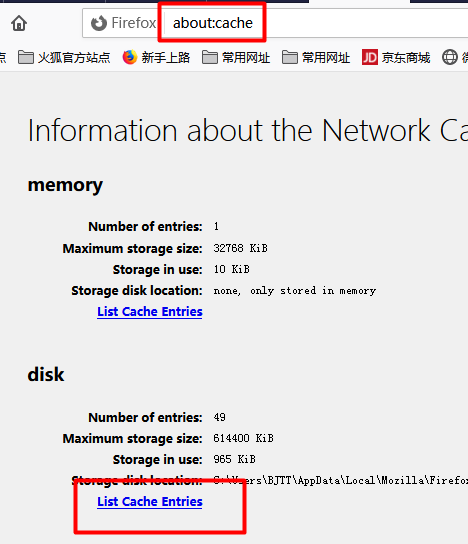
}

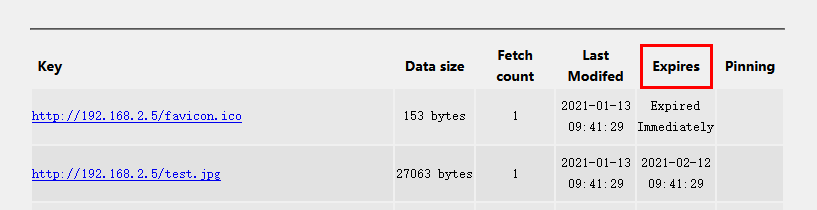
然后使用火狐浏览器，先清空历史记录，然后地址栏输入about:cache

查看disk文件的列表，找到被访问文件看最后倒数第2列(Expires)信息显示多

久超时，再访问192.168.2.5/test.jpg后再刷新网页，可以看到该文件的

过期时间是一个月之后。





**4, 优化nginx支持超长地址栏，默认情况下nginx无法支持长地址栏，会**

**报414错误**

打开配置文件,在默认的虚拟主机上方添加两行

client\_header\_buffer\_size 200k; //第一行表示,用户访问网站的头部信

息(包含地址栏)长度支持200k大小

large\_client\_header\_buffers 4 200k; //第二行表示,如果200k不够,再

给4个200k

server {

listen 80;

server\_name localhost;

sbin/nginx -s reload //重加载配置

然后到lnmp\_soft目录找到buffer.sh 脚本并运行, 看不到414报错即可

./buffer.sh //执行测试脚本，可以支持超长地址栏并看到页面内容，而

不是414报错

---------------------------------

buffer.sh脚本在执行时会产生超长地址

#!/bin/bash

URL=http://192.168.2.5/index.html?

for i in {1..5000}

do

URL=${URL}v$i=$i

URL=http://192.168.2.5/index.htmlv1=1 //产生的过程，第1次循环

URL=http://192.168.2.5/index.htmlv1=1v2=2 //产生的过程，第2次循环

URL=http://192.168.2.5/index.htmlv1=1v2=2v3=3 //产生的过程，第3次循环….

。。。。。。。

。。。。。。。

done

curl $URL

--------------------------------------

**5，nginx并发访问的优化**

**并发：多数用户同时对网站发起访问，并发量支持的越高，说明网站性能越强。**

**默认情况下nginx并发仅仅支持1024个，需要修改配置才能增加**

yum -y install httpd-tools //安装支持压力测试命令的软件包

ab -c 200 -n 200 <http://192.168.2.5/> //使用压力测试工具模拟200人，

每人访问网站1次，一共是200次，可以看到100%的提示，说明测试成功

ab -c 2000 -n 2000 <http://192.168.2.5/> //增加到2000人时失败

**打开nginx配置文件修改第3行，第13行**

worker\_processes 2; //开启的nginx进程数量，通常是随cpu的核心数一致

worker\_connections 50000; //每个nginx进程支持的并发访问量

**另外，除了nginx本身对并发量有限制，linux系统本身对文件的访问也有限制，默认情况下linux系统仅允许一个文件同时被打开1024次，普通情况下够用，但是作为网站服务器时，网站页面被N多用户同时访问时相当于同时打开，仅仅支持1024显然不够。**

ulimit -n //检查系统对文件打开数量的值，默认1024

vim /etc/security/limits.conf //修改53、54行，之后重启服务器，这里的soft是软件限制，hard是硬件限制，nofile表示被修改的内容是文件可以被打开的次数，一共定义为10万，这样就可以放开限制了

\* soft nofile 100000

\* hard nofile 100000

之后web1也安装httpd-tools,也按照上述配置修改配置文件，然后

重启web1，等proxy和web1都重启好了之后，

setenforce 0 //都关闭selinux，和proxy的防火墙

[root@proxy nginx]# sbin/nginx //proxy再开启nginx服务

ulimit -n //再测试文件的访问数量是否为10万

[root@web1 ~]# ab -c 2000 -n 2000 <http://192.168.2.5/> //最后用

web1测试proxy的并发量可以突破2000则成功

-----------------------------------------------------------------------------------------

**解决集群主机过多而导致用户重复登陆网站的问题**

**在一个集群中，如果网站需要用户输入用户名和密码登陆之后才能继续访问，那么当用户登陆其中一台集群主机之后随着继续访问页面，请求可能被代理服务器轮询到另外一台服务器上，那么对于另外一台服务器来说用户并没有登陆，想查看登陆之后的页面还需要再次登陆，这样集群主机越多需要客户重复登陆的次数就越多，想要解决该问题就要从Session与Cookies入手**

**Session：存储在服务器端，保存用户名、登陆状态等信息。**

**Cookies：由服务器下发给客户端，保存在客户端的一个文件里。**

**1，在web1与web2主机部署lnmp环境：**

[root@proxy ~]# scp lnmp\_soft.tar.gz 192.168.2.100: //从proxy主机拷贝软件包到web1的家目录

[root@web1 ~]# yum -y install gcc pcre-devel openssl-devel

[root@web1 ~]# tar -xf lnmp\_soft.tar.gz

[root@web1 ~]# cd lnmp\_soft/

[root@web1 lnmp\_soft]# tar -xf nginx-1.17.6.tar.gz

[root@web1 lnmp\_soft]# cd nginx-1.17.6/

[root@web1 nginx-1.17.6]# ./configure

make

make install

yum -y install mariadb mariadb-server mariadb-devel

yum -y install php php-mysql php-fpm

systemctl start mariadb

systemctl start php-fpm

ss -ntulp | grep :3306 //检查数据库

ss -ntulp | grep :9000 //检查php-fpm

cd /usr/local/nginx //进入nginx安装目录

vim conf/nginx.conf //修改配置文件,实现动静分离,修改65~71行为以下状态

location ~ \.php$ {

root html;

fastcgi\_pass 127.0.0.1:9000;

fastcgi\_index index.php;

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

include fastcgi.conf;

}

[root@web1 nginx]# sbin/nginx

[root@web1 nginx]# cd ~/lnmp\_soft/php\_scripts/

cp test.php /usr/local/nginx/html //拷贝测试页面

<http://192.168.2.100/test.php>

**web2也配置LNMP环境，并实现nginx动静分离**

**总体思路：**

1. **拷贝lnmp\_soft.tar.gz软件包到web2**
2. **释放里面的nginx软件包，安装nginx软件依赖包并编译安装**
3. **安装LNMP其他软件包，如：数据库、php**

**4，修改nginx配置文件，实现动静分离**

**提醒，关闭防火墙，开启LNMP环境依赖的服务**

**2，再测试可以登录的网页**

回到web1主机：

cd /root/lnmp\_soft/php\_scripts

tar -xf php-memcached-demo.tar.gz //释放带登录功能的网页

cp -r php-memcached-demo/\* /usr/local/nginx/html/ //拷贝测试页面到nginx中

cd /usr/local/nginx

ls html //查看测试页面，可以看到2个文件，其

中有index.php登录界面和home.php登录后的页面

使用火狐浏览器查看192.168.2.100/home.php 直接无法访问,必须先登录

ls /var/lib/php/session/ //登录之后查看服务器session目录可以看

到session文件,有该文件,用户再次访问home页面无需重复登录,如果

将该目录下的session文件删除,则用户需要重新登录,用户那边也可以清

空浏览器的历史记录(其中包含Cookies),也需要再次登录.

web1与web2都要关闭防火墙！！

yum -y reinstall php //如果不小心删除session目录,需要重装php软件包

给页面增加红色标记,web2要改成web2

[root@web1 nginx]# vim html/index.php //在第13行加web1

<h1>web1 PHP Login Session Example</h1>

[root@web1 nginx]# vim html/home.php //在第17行加web1

web1 Welcome : <?php echo $\_SESS …….

**3，在proxy主机配置集群，注意该主机不能有动静分离的配置**

upstream web { //在第34行,也就是默认的虚拟主机上面添加集群

server 192.168.2.100:80;

server 192.168.2.200:80;

}

location / {

proxy\_pass <http://web>; //然后在默认的虚拟主机中的location里调用集群

sbin/nginx -s reload //重新加载配置文件

http://192.168.2.5/index.php //之后测试效果，不断刷新页面，会看到web1与web2

的登录界面(需要提前在web1与web2的index.php与home.php页面进行标记)

删除web1与web2的session文件

rm /var/lib/php/session/\*

删除火狐浏览器的历史记录(主要是cookies文件)

然后尝试登录<http://192.168.2.5/index.php，发现还是需要2>次才能成功

----------------------------------------------------------------------

**上述实验由于web1与web2都是在各自的/var/lib/php/session目录中存储session，所以造成客户需要重复登录，为了统一session存储的位置（该存储方式通常被称为session共享），需要安装专门的数据库工具**

**memcache 可以利用内存存储数据的高性能数据库服务**

**1，在proxy安装memcache数据库**

[root@proxy nginx]# yum -y install memcached //安装数据库工具

[root@proxy nginx]# systemctl start memcached //启动服务

yum -y install telnet //安装远程登录工具，为了测试memcache

**2，连接测试，验证memcache是否正常：**

telnet 127.0.0.1 11211 //使用远程工具连接到本机的11211端口

set abc 0 200 3 //创建(覆盖)变量abc 0是不压缩，保存200秒，存3个字符

get abc //获取变量

add abc 0 200 3 //仅仅创建变量，如果已经有abc变量，则失败

replace abc 0 200 3 //仅仅覆盖变量，如果变量不存在，则失败

delete abc //删除变量

flush\_all //删除所有数据

quit //退出

**3，在web1与web2主机修改session存储的位置，实现session共享**

[root@web1 session]# vim /etc/php-fpm.d/www.conf //修改最后

两行为以下状态

php\_value[session.save\_handler] = memcache //这里改成memcache，表示session的存储不在是本地的普通文件，而是去找memcache

php\_value[session.save\_path] = tcp://192.168.2.5:11211 //这里的路径修改为安装了memcache服务的服务器地址与端口

[root@web1 session]# yum -y install php-pecl-memcache //安装php与memcached服务关联的软件包

[root@web1 session]# systemctl restart php-fpm //重启fpm服务，让上述配置生效

关闭所有服务器的selinux

然后去web2做与web1相同的配置

最后清空浏览器的历史记录，再访问http://192.168.2.5/index.php仅仅登录一次即

可成功