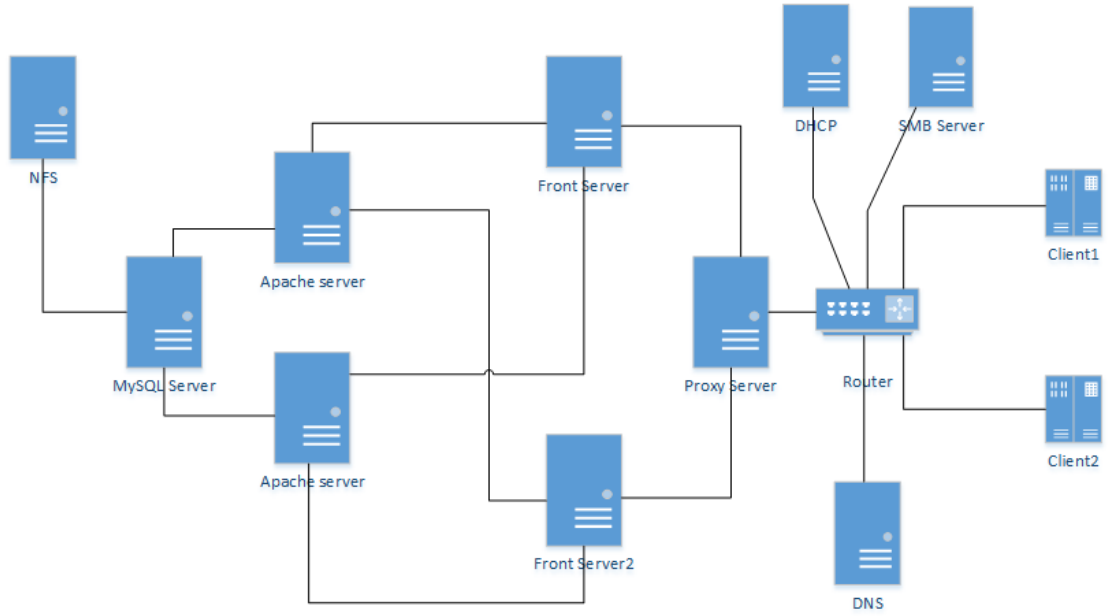
技术文档

网络拓扑图：

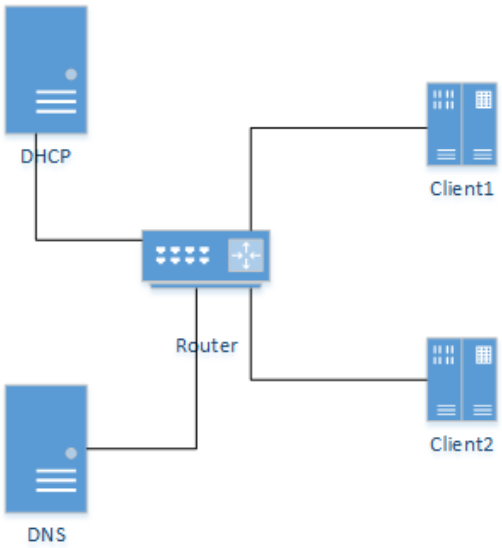


项目完成以下目的：

- 负载均衡的 Web 服务器
- 提供动态 IP 和域名
- 网络访问限制和缓存实现
- 高可用性数据库和存储
- 文件共享和备份

如拓扑图所示，在此项目中，我们设置了 10 台服务器。
我们使用两个前端服务器，它们设置 Haproxy 来平衡不同的 Apache 服务器。
使用 DHCP 服务器分配 IP 地址。
使用 DNS 服务器解析域名。
使用代理服务器设置缓存和限制网站访问。
使用 Apache 服务器托管不同的网站，例如 wordpress 和 bookstack。
使用 MySql Server 管理 mysql 数据库。
使用 NFS 服务器存储并备份来自 MySql Server 的导入数据。

动态 IP 和域名提供



DNS:
在这一部分，DNS 服务器使用 Bind 提供域名的前向和后向解析。
在 DNS 配置文件中，尤其是在前向解析文件“ahrs.com.zone”中，如下所示。

```
$TTL 1D

@       IN SOA  ahrs.com. rname.invalid. (
                                        0       ; serial
                                        1D      ; refresh
                                        1H      ; retry
                                        1W      ; expire
                                        3H )    ; minimum
@               IN NS master.ahrs.com.
master.ahrs.com. IN A 172.16.250.100


www.ahrs.com. IN A 172.16.250.1
www.ahrs.com. IN A 172.16.250.9
www.ahrs.lan. IN CNAME www.ahrs.com.
*       IN A    8.8.8.8
*       IN A    8.8.4.4
*       IN A    1.1.1.1
```

forward resolution file -ahrs.com.zone

我们添加了 DNS 服务器名称和 DNS 服务器 ip 地址，为了满足在 Apache 服务器中设置虚拟主机的需要，我们添加了一个新的主机名以供以后在 Apache 服务器中使用。
同时，由于域名未知，我们添加了一些 DNS 地址进行转发。
为了达到更好的网络负载均衡。 我们使用 DNS-Round Robin 在 DNS 服务器配置中提供这两个 Haproxy 服务器。 我们将多个“A”记录添加到前向解析文件中的同一主机上。 DNS 服务器根据它们记录的记录，将客户端请求随机分配给不同的 IP 地址。 然后，我们在 DNS 服务器上实现了简单的负载平衡。

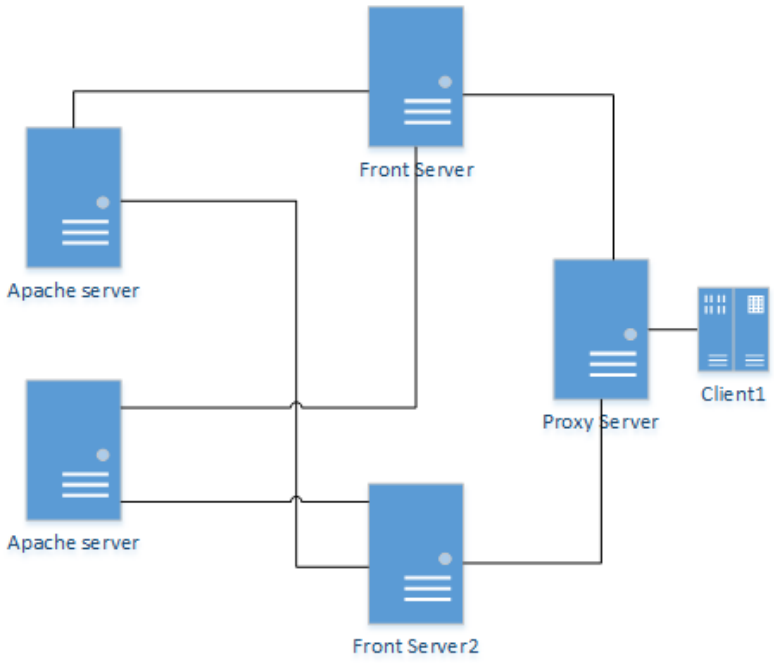
DHCP:
在这一部分，DHCP 服务器使用 dhcp 软件来管理不同的 IP 地址。

在此项目中，我们需要将所有服务器都设置在固定 ip 网段中，因此首先要添加一个新的网络适配器以满足固定 ip 范围。
在 DHCP 服务器配置文件“dhcpd.conf”中，如下所示。

```
#
# DHCP Server Configuration file.
#   see /usr/share/doc/dhcp*/dhcpd.conf.example
#   see dhcpd.conf(5) man page

subnet 172.16.250.0 netmask 255.255.255.0{
        range 172.16.250.0 172.16.250.255;
        option domain-name-servers 172.16.250.1;
        option domain-name "ahrs.lan";
        option routers 172.16.250.1;
        option broadcast-address 172.168.250.255;
        default-lease-time 259200;
        max-lease-time 518400;
        ddns-update-style none;
}
```

dhcpd.conf

负载均衡且可用的 Web 服务器



Haproxy:
为了平衡所有 Apache 服务器之间的流量，我们在项目中使用 Haproxy 软件。
为了实现此功能，我们在两个 Haproxy 服务器，也就是两个 Front Server 中添加了两个 apache 服务器的 IP 地址，然后当客户端要访问 Web 服务器时，它将从 DNS 服务器获取 Haproxy 服务器的 IP 地址。 这意味着当客户访问网址时，他会访问不同的 Apache 服务器。这就是 Haproxy 服务器正常工作的结果。

Haproxy 的配置文件如下：

```
#---------------------------------------------------------------------
frontend  main *:80
    acl url_static       path_beg       -i /static /images /javascript /stylesheets
    acl url_static       path_end       -i .jpg .gif .png .css .js

    use_backend static          if url_static
    default_backend             app

#---------------------------------------------------------------------
# static backend for serving up images, stylesheets and such
#---------------------------------------------------------------------
backend static
    balance     roundrobin
    server      static 127.0.0.1:80 check

#---------------------------------------------------------------------
# round robin balancing between the various backends
#---------------------------------------------------------------------
backend app
    balance     roundrobin
    server  app1 127.0.0.1:5001 check
    server  app2 127.0.0.1:5002 check
    server  app3 127.0.0.1:5003 check
    server  app4 127.0.0.1:5004 check
    server  app5 172.16.250.5:80 check
    server  app6 172.16.250.6:80 check
```

Haproxy.cfg

Apache:

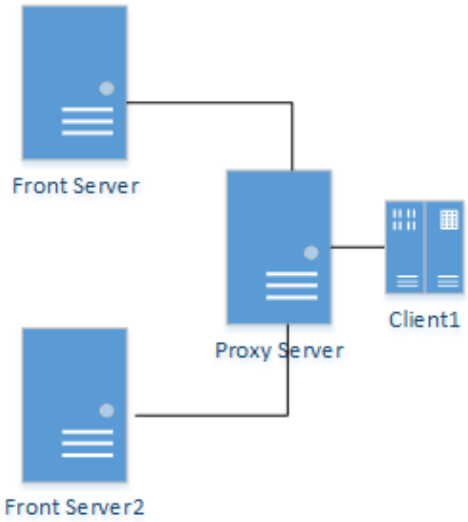
在本项目中，我们要求在同一 Apache 服务器中设置不同的网站，根据这个需求，我们采用虚拟主机技术，在每个 Apache 服务器中设置 “virtual.conf” 文件。更重要的是，在该文件中写入的域名必须与 DNS 服务器上设置的主机名相同。

配置信息如下：

```
NameVirtualHost *:80

#<Directory "var/www/www">
#       Options FollwSymLinks
#       AllowOverride None
#       Order allow,deny
#       Allow from all
#</Directory>

####
<VirtualHost *:80>
        ServerName www.ahrs.lan
        DocumentRoot /var/www/html
</VirtualHost>

<VirtualHost *:80>
        ServerName www.ahrs.com
        DocumentRoot /var/www/www
</VirtualHost>
```

virtual.conf

网络访问限制和缓存实现



为了实现网络缓存和网站访问限制，我们使用 Squid 软件来实现。当客户端对 Internet 发出数据请求时，代理服务器可帮助用户从目标网站获取数据。访问完成后，数据将存储在缓存中。当客户端再次请求时，代理服务器首先检查缓存，如果它们具有相同的数据，则将数据发送到客户端而无需通过 Internet，这就是代理服务器可以减少带宽的原因。同时，Squid 软件可帮助我们阻止对某些网站的访问。

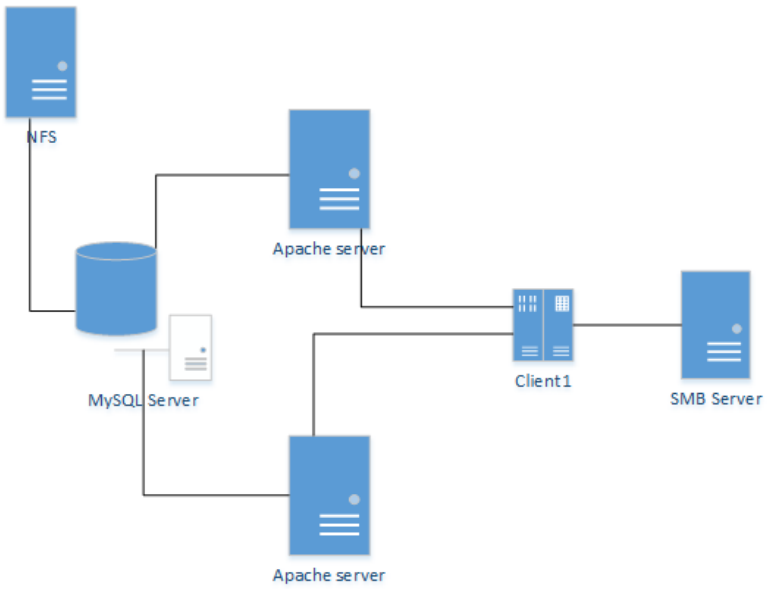
所有这些配置都在 “squid.conf” 文件中设置，配置信息如下：

```
http_access deny !Safe_ports

# Deny CONNECT to other than secure SSL ports
http_access deny CONNECT !SSL_ports

# Only allow cachemgr access from localhost
http_access allow localhost manager
http_access deny manager

# We strongly recommend the following be uncommented to protect innocent
# web applications running on the proxy server who think the only
# one who can access services on "localhost" is a local user
#http_access deny to_localhost

#
# INSERT YOUR OWN RULE(S) HERE TO ALLOW ACCESS FROM YOUR CLIENTS
#acl ahrs src 172.16.250.0-172.16.250.255/24
acl spyware dstdomain "etc/squid/spyware-domains.txt"
acl warez dstdomain "etc/squid/warez-domains.txt"
http_access deny spyware
http_access deny warez
#http_access allow ahrs
#

# Example rule allowing access from your local networks.
# Adapt localnet in the ACL section to list your (internal) IP networks
# from where browsing should be allowed
http_access allow localnet
http_access allow localhost

# And finally deny all other access to this proxy
http_access allow all

# Squid normally listens to port 3128
http_port 3128
```

squid.conf

高可用性数据库和存储

拓扑图：



首先，我们使用磁盘冗余技术在 MySQL Server 和 NFS Server 上设置 RAID10 磁盘阵列，以防止由于磁盘损坏而导致数据丢失。
我们的数据库每天 24 小时备份并存储在 NFS Server 中，通过脚本来实现定期删除一周前存储的备份，以减少磁盘消耗。
我们配置 OpensSH-Server，设置主机的配置文件，通过 SSH 配置来允许并访问 MySQL Server。

然后在两台服务器上生成密钥，并将 MySQLserver 密钥复制到 SSH 中
配置信息如下：

```
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa):
/root/.ssh/id_rsa already exists.
Overwrite (y/n)? y
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /root/.ssh/id_rsa.
Your public key has been saved in /root/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:R0yuq7Pi8dywCBV/qVfUJvOcINHS5hWYu7IGy30tK5A root@BACK
The key's randomart image is:
+---[RSA 2048]----+
|        .o.o..   |
|        .=*..    |
|    .   .+Ooo    |
|     o   *oB .   |
|    . ..S o.+    |
|     .  Eo.+.    |
|  . ..o=oo .     |
|   ..=+*= o .    |
|   .o.**.o.o     |
+----[SHA256]-----+
[root@BACK ~]# _
```

自动备份脚本：

```
BACKUP=/home/backup/mysql
DATETIME=$(date +%Y-%m-%d_%H%M%S)
echo “===BACKUP START===”
echo “Backup files ate stored in ${BACKUP}/$DATETIME.tar.gz”
HOST=localhost
DB_USER='root'
DB_PW='123ZCWzcw<<<'
[ ! -d “${BACKUP}/$DATETIME” ] && mkdir -p "${BACKUP}/$DATETIME"
DATEBASE=mysql
mysqldump -u${DB_USER} -p${DB_PW} --single-transaction --host=$HOST -q -R --databases
cd $BACKUP
tar -zcvf $DATETIME.tar.gz $DATETIME
scp $DATETIME.tar.gz root@192.168.43.129:/home/mysqlbackup
rm -rf ${BACKUP}/$DATETIME
find $BACKUP -mtime +30 -name "*.tar.gz" -exec rm -rf {} \;
echo “===backuo succeed===”
```

同时，我们还在数据库服务器上配置了 PhpMyAdmin 来进行数据库控制，通过安装 PhpMyAdmin 预包装 EPEL 和 REMi，建立了到 Apache 服务器的连接。除此之外，在客户端开发人员可以通过远程连接的方式来实现数据库管理。

文件共享和备份

NFS：

NFS 是网络文件系统的缩写。它的最大功能是允许不同的机器和不同的操作系统通过网络彼此共享单个文件。我们在文件服务器和数据库服务器之间使用 NFS 协议来完成文件共享的功能。

这里我们调用（RPC）服务。RPC 的主要功能是指定与每个 NFS 功能相对应的端口号，并将其报告给客户端，以便客户端可以连接到正确的端口。

同时为了提高 NFS 的稳定性，我们使用 TCP 协议来进行挂载（NFS 默认使用 UDP 协议）。挂载信息如下：

```
[root@localhost ~]# mount -t nfs 192.168.1.200:/data/zhr /zhr -o proto=tcp -o nolock
[root@localhost ~]# df -h
Filesystem               Size  Used Avail Use% Mounted on
devtmpfs                 1.9G     0  1.9G   0% /dev
tmpfs                    1.9G     0  1.9G   0% /dev/shm
tmpfs                    1.9G   13M  1.9G   1% /run
tmpfs                    1.9G     0  1.9G   0% /sys/fs/cgroup
/dev/mapper/centos-root   17G  6.6G   11G  39% /
/dev/sda1               1014M  165M  850M  17% /boot
tmpfs                    378M     0  378M   0% /run/user/0
192.168.1.200:/data/zhr   17G  3.7G   14G  22% /zhr
```

SMB:

在本项目中，我们同时使用了 SMB 来实现文件共享。

SMB（服务器消息块）是一个协议名称，可用于 Web 连接以及客户端和服务器之间的通信。当新员工加入网络中，需要提供了一个脚本来自动生成密码，来访问存储上属于他的文件夹。根据这个需求，采用脚本实现。

用户生成脚本如下：

```
string=abcdefghijklmnopqrstuvwxyzABCDEFGHIGKLMNOPQRSTUVWXYZ1234567890

for i in {i,2,3,4,5,6,7}
do
   num=$[RANDOM%62]
   x=${string:num:1}
   password1=${password1}$x
done
char=\!\@\#\$\%\^\&\*\(\)\_\+\>
for j in {1}
do
   num=$[RANDOM%26]
   y=${char:num:1}
   password2=${password2}$y
done

useradd $1
echo ${password1}${password2} | passwd --stdin $1
echo "Adding user $1"
echo "Password is ${password1}${password2}"
smbpasswd -a $1
```

在现有的几个用户中，我们为不同的用户配置了不同的 SMB 文件夹，并配置了它们所属的组。来保证该组中的用户只能访问该组中的文件，而不能访问其他组的文件。

配置信息如下：

```
[germaine]
        comment = germaine
        path=/home/germaine
        read only = no
        write list = germaine
        valid users = germaine
        public = no
```

michel 的访问和权限结果：

```
-bash-4.2$ smbclient //172.16.250.252/hr -U michel
Enter SAMBA\michel's password:
Try "help" to get a list of possible commands.
smb: \> ls
  .                                   D        0  Tue Nov 20 20:31:30 2020
  ..                                  D        0  Tue Nov 20 20:31:30 2020
  TeamData                            D        0  Tue Nov 20 20:31:30 2020

                17811456 blocks of size 1024. 14416240 blocks available
smb: \> quit
-bash-4.2$ smbclient //172.16.250.252/hr -U germaine
Enter SAMBA\germaine's password:
tree connect failed: NT_STATUS_ACCESS_DENIED
```