

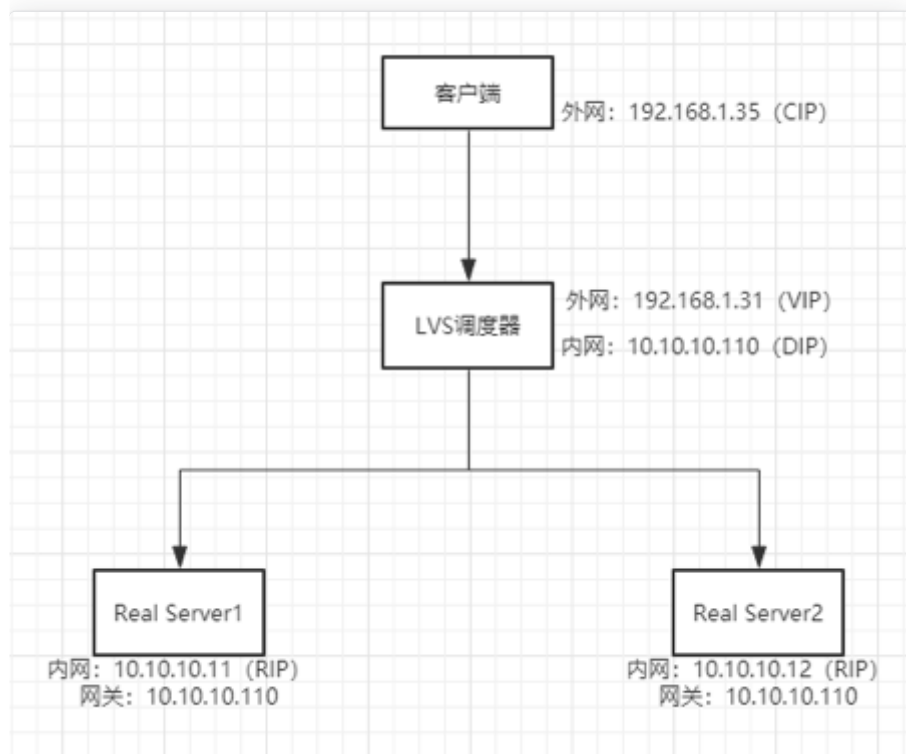


## LVSNAT模式实现

Post 2019-04-13 01:10 Read 558 Comment 0

### LVSNAT模式配置

NAT 模式架构图:



### 操作步骤



实验环境准备: (centos7平台)

hostname	IPaddress	说明
lvs-director	外网: 192.168.1.31 内网: 10.10.10.110	LVS 调度器 (Director)
web01	内网: 10.10.10.11	RS1 (后端真实服务器)
web02	内网: 10.10.10.12	RS2 (后端真实服务器)
client	外网: 192.168.1.35	用来测试 (可有可无)



## 所有服务器上配置

```
# systemctl stop firewalld //关闭防火墙
# sed -i 's/^SELINUX=.*SELINUX=disabled/' /etc/sysconfig/selinux //关闭selinux, 重启生效
# setenforce 0 //关闭selinux, 临时生效
# ntpdate 0.centos.pool.ntp.org //时间同步
注意: realserver的网关需要指向DIP
```

### 步骤一: 配置 realserver

在 realserver (web01和web02) 上安装 nginx, 并在不同的 web 服务器上建立不同的主页内容 (方便测试), 并启动。

#### 1) 在 web01 服务器配置

```
[root@web01 ~]# yum install nginx -y
[root@web01 ~]# echo "`hostname` ifconfig ens33 |sed -n 's#.*inet \(.*\)netmask.*#\1#p`" > /usr/share/nginx/html/index.html
[root@web01 ~]# systemctl start nginx
[root@web01 ~]# systemctl enable nginx
```

#### 2) 在 web02 服务器配置

```
[root@web02 ~]# yum install nginx -y
[root@web02 ~]# echo "`hostname` ifconfig ens33 |sed -n 's#.*inet \(.*\)netmask.*#\1#p`" > /usr/share/nginx/html/index.html
[root@web02 ~]# systemctl start nginx
[root@web02 ~]# systemctl enable nginx
```

### 步骤二: 调度器上配置



1) 在调度器上打开ip转发, 因为在上面架构图中, 调度器会用到两个IP段的转发

```
# vim /etc/sysctl.conf
net.ipv4.ip_forward = 1
# sysctl -p //使之生效
```

2) 在调度器 (director) 上安装软件包 (ipvsadm)

```
# yum install ipvsadm -y
# rpm -qa |grep ipvsadm //检查是否安装成功
```



3) 按照架构图来配置lvs进程调度

```
[root@lvs-director ~]# ipvsadm -A -t 192.168.1.31:80 -s rr //--A参数增加服务, -t 绑定的ip地址 (VIP), -s指定调度算法
[root@lvs-director ~]# ipvsadm -a -t 192.168.1.31:80 -r 10.10.10.11:80 -m // -a 指定增加真实服务器, -r 指定realserver; -m 表示使用 NAT 模式
[root@lvs-director ~]# ipvsadm -a -t 192.168.1.31:80 -r 10.10.10.12:80 -m
// 上面三条写的就是访问192.168.1.31的80端口会以rr算法调给10.10.10.11的80和10.10.10.12的80端口。
[root@lvs-director ~]# ipvsadm -ln //查看ipvsadm调度规则
```

IP Virtual Server version 1.2.1 (size=4096)

Prot LocalAddress:Port Scheduler Flags

-> RemoteAddress:Port Forward Weight ActiveConn InActConn

TCP 192.168.1.31:80 rr

-> 10.10.10.11:80 Masq 1 0 0

-> 10.10.10.12:80 Masq 1 0 0

步骤三: 测试

1) rr算法验证

在client 端进行访问验证

```
[root@client ~]# elinks 192.168.1.31
```





```
[root@client ~]# elinks 192.168.1.31
```

```
web02 10.10.10.12 http://192.168.1.31/
```

验证结果为 web1 页面和web2 页面轮循

验证一：健康检查



停掉web01上的nginx服务

```
[root@web01 ~]# systemctl stop nginx
```

```
[root@node01 ~]# elinks 192.168.1.31
```

```
web02 10.10.10.12 http://192.168.1.31/
```

验证结果只有 web02 了，说明健康检查ok

验证二：验证调度MySQL服务

1、在后端 realserver 装两个 mysql ，并启动服务，分别创建两个不同名的库，方便测试

```
# yum install mariadb mariadb-server
```

```
# systemctl start mariadb
```

```
MariaDB [(none)]> create database mariadb1; //web1上操作
```

```
MariaDB [(none)]> create database mariadb2; //web2上操作
```

2、两个 mysql 授权，这里授权的 IP 应该为客户端的 IP 192.168.1.35 (通过 IP 包的原理分析得到)


```
MariaDB [(none)]> grant all on *.* to 'abc'@'192.168.1.35' identified by '123';
```

```
MariaDB [(none)]> flush privileges;
```

3、在 调度器上面添加调度规则

```
[root@lvs-director ~]# ipvsadm -A -t 192.168.1.31:3306 -s rr
```

```
[root@lvs-director ~]# ipvsadm -a -t 192.168.1.31:3306 -r 10.10.10.11:3306 -m
```



```
[root@lvs-director ~]# ipvsadm -a -t 192.168.1.31:3306 -r 10.10.10.12:3306 -m
```

4、客户端 192.168.1.35 上使用 (mysql -h 192.168.1.31 -u 授权用户名 -p授权密码) 来测试, 通过 show databases; 查看库中有 mariadb1 爱是 mariadb2, 就知道连接的是哪个 mysql。



```
[root@client ~]# mysql -h 192.168.1.31 -u abc -p123
```

.....

```
MariaDB [(none)]> show databases;
```

```
+-----+
```

```
| Database |
```

```
+-----+
```

```
| information_schema |
```

```
| mariadb1 |
```

```
| mysql |
```

```
| performance_schema |
```

```
| test |
```

```
+-----+
```

```
5 rows in set (0.00 sec)
```



```
[root@client ~]# mysql -h 192.168.1.31 -u abc -p123
```

.....

```
MariaDB [(none)]> show databases;
```

```
+-----+
```

```
| Database |
```

```
+-----+
```

```
| information_schema |
```

```
| mariadb2 |
```

```
| mysql |
```

```
| performance_schema |
```

```
| test |
```

```
+-----+
```

```
5 rows in set (0.00 sec)
```





## 2) 会话保持验证

会话保持：永久粘贴和持续性的比较

验证一：永久性会话粘贴（类似 nginx 的 ip\_hash）

### 1、在调度器上面配置调度规则



```
[root@lvs-director ~]# ipvsadm -A -t 192.168.1.31:80 -s sh
[root@lvs-director ~]# ipvsadm -a -t 192.168.1.31:80 -r 10.10.10.12:80 -m
[root@lvs-director ~]# ipvsadm -a -t 192.168.1.31:80 -r 10.10.10.11:80 -m
[root@lvs-director ~]# ipvsadm -ln
IP Virtual Server version 1.2.1 (size=4096)
Prot LocalAddress:Port Scheduler Flags
  -> RemoteAddress:Port          Forward Weight ActiveConn InActConn
TCP  192.168.1.31:80  sh
    -> 10.10.10.11:80             Masq    1      0          0
    -> 10.10.10.12:80             Masq    1      0          0
```



### 2、测试结果如下，如果客户端第一次访问的是 web02，那么永远访问的是 web02（web02挂了还是访问 web02，这里的健康检查就无效了）



```
[root@client ~]# curl 192.168.1.31
web02 10.10.10.12
[root@client ~]# curl 192.168.1.31
web02 10.10.10.12
[root@web02 ~]# systemctl stop nginx
[root@client ~]# curl 192.168.1.31
curl: (7) Failed connect to 192.168.1.31:80; 拒绝连接
[root@client ~]# curl 192.168.1.31
curl: (7) Failed connect to 192.168.1.31:80; 拒绝连接
```



## 验证二：通过持续性 `persistent` 实现非永久性的会话粘贴

1、启动上面停掉的web02 的服务，并将调度器上面的规则更改为 `persistent` 持续性的会话粘贴 `-E` 参数 修改已有的规则



```
[root@lvs-director ~]# ipvsadm -E -t 192.168.1.31:80 -r ~ -p 10
[root@lvs-director ~]# ipvsadm -ln
IP Virtual Server version 1.2.1 (size=4096)
Prot LocalAddress:Port Scheduler Flags
  -> RemoteAddress:Port          Forward Weight ActiveConn InActConn
TCP  192.168.1.31:80 rr persistent 10
  -> 10.10.10.11:80                Masq    1      0      0
  -> 10.10.10.12:80                Masq    1      0      0
```



2、验证，在客户端上面访问，然后查看调度器上面的调度信息。



```
[root@client ~]# curl 192.168.1.31
web01 10.10.10.11
[root@client ~]# curl 192.168.1.31
web01 10.10.10.11
[root@lvs-director ~]# ipvsadm -lnc
IPVS connection entries
pro expire state      source          virtual         destination
TCP 00:07 NONE      192.168.1.35:0  192.168.1.31:80 10.10.10.11:80
TCP 01:55 TIME_WAIT 192.168.1.35:59132 192.168.1.31:80 10.10.10.11:80
TCP 01:56 TIME_WAIT 192.168.1.35:59134 192.168.1.31:80 10.10.10.11:80
```



通过上面的命令看到客户端 `192.168.1.35` 访问了两次`192.168.1.31:80`，被调度到了`10.10.10.11:80`

说明：当一个 `client` 访问 `vip` 的时候，`ipvs` 或记录一条状态为 `NONE` 的信息，`NONE` 状态前面的 `expire` 值是 `persistence_timeout` 的值（这里设置为10，所以从10开始计算），然后根据时钟主键变小，在以下记录存在时间，同一 `client ip` 连接上来，都会被分配到同一个后盾。`TIME_WAIT` 的值就是 `tcp tcpfin udp` 的超时时间，



一个客户端 ip 可能有几个 TIME\_WAIT , 它们公用一个 NONE, 当 NONE 的值为 0 时, 如果 TIME\_WAIT 还存在, 那么 NONE 的值会重新变成 60秒, 再减少, 知道这个客户端 IP 的所有 TIME\_WAIT 消失以后, NONE 才会消失, 只要 NONE 存在, 同一 client 的访问, 都会分配到统一的 real server。也就是说, 客户端 192.168.1.35 第一次访问 被调度给 10.10.10.11:80, 要等  $2*60+10=130$ 秒左右的时间再做第二次访问才可能被调度给另一个 real server。

### 3) wrr算法验证

wrr 加权轮循, 先将上面创建的规则删除, 然后再添加规则 (也可以直接修改 -E)



```
[root@lvs-director ~]# ipvsadm -D -t 192.168.1.31:80
[root@lvs-director ~]# ipvsadm -A -t 192.168.1.31:80 -s wrr
[root@lvs-director ~]# ipvsadm -a -t 192.168.1.31:80 -r 10.10.10.11:80 -m -w 2
[root@lvs-director ~]# ipvsadm -a -t 192.168.1.31:80 -r 10.10.10.12:80 -m -w 1
[root@lvs-director ~]# ipvsadm -ln
IP Virtual Server version 1.2.1 (size=4096)
Prot LocalAddress:Port Scheduler Flags
  -> RemoteAddress:Port          Forward Weight ActiveConn InActConn
TCP  192.168.1.31:80 wrr
  -> 10.10.10.11:80                Masq    2      0          0
  -> 10.10.10.12:80                Masq    1      0          0
```



验证, 在客户端机上测试结果如下, 可以发现按照配置的规则2:1的调度到 web01 和web02

```
[root@client ~]# curl 192.168.1.31
web01 10.10.10.11
[root@client ~]# curl 192.168.1.31
web01 10.10.10.11
[root@client ~]# curl 192.168.1.31
web02 10.10.10.12
```

### 4) lc算法验证



通过实时的链接数来判定，链接数少的会被分配更多的请求，此处就不测试了，因为这个要大规模连接才能看出效果来。配置方法和上面一样，只是更改 `-s` 后面的算法，示例：

```
[root@lvs-director ~]# ipvsadm -A -t 192.168.1.31:80 -s lc
[root@lvs-director ~]# ipvsadm -a -t 192.168.1.31:80 -r 10.10.10.11:80 -m
[root@lvs-director ~]# ipvsadm -a -t 192.168.1.31:80 -r 10.10.10.12:80 -m
```



## 5) wlc算法验证

wlc 加权最小连接

```
[root@lvs-director ~]# ipvsadm -A -t 192.168.1.31:80 -s wlc
[root@lvs-director ~]# ipvsadm -a -t 192.168.1.31:80 -r 10.10.10.11:80 -m
[root@lvs-director ~]# ipvsadm -a -t 192.168.1.31:80 -r 10.10.10.12:80 -m
[root@lvs-director ~]# ipvsadm -ln
IP Virtual Server version 1.2.1 (size=4096)
Prot LocalAddress:Port Scheduler Flags
  -> RemoteAddress:Port          Forward Weight ActiveConn InActConn
TCP  192.168.1.31:80 wlc
  -> 10.10.10.11:80               Masq    1      0          0
  -> 10.10.10.12:80               Masq    1      0          0
```

等等 还有别的算法都是一样的配置。

## 将 ipvsadm 规则加入开机启动

因为ipvsadm 命令添加的规则，重启后就没有了，此时我们就需要保存下来，添加到开机启动

```
[root@lvs-director ~]# rpm -ql ipvsadm
/etc/sysconfig/ipvsadm-config
/usr/lib/systemd/system/ipvsadm.service
/usr/sbin/ipvsadm
```



```
/usr/sbin/ipvsadm-restore
/usr/sbin/ipvsadm-save
/usr/share/doc/ipvsadm-1.27
/usr/share/doc/ipvsadm-1.27/README
/usr/share/man/man8/ipvsadm-restore.8.gz
/usr/share/man/man8/ipvsadm-save.8.gz
/usr/share/man/man8/ipvsadm.8.gz
```



通过查看ipvsadm 软件包安装清单，可以看到有启动服务脚本，内容如下：

```
[root@lvs-director ~]# cat /usr/lib/systemd/system/ipvsadm.service
[Unit]
Description=Initialise the Linux Virtual Server
After=syslog.target network.target

[Service]
Type=oneshot
ExecStart=/bin/bash -c "exec /sbin/ipvsadm-restore < /etc/sysconfig/ipvsadm"
ExecStop=/bin/bash -c "exec /sbin/ipvsadm-save -n > /etc/sysconfig/ipvsadm"
ExecStop=/sbin/ipvsadm -C
RemainAfterExit=yes

[Install]
WantedBy=multi-user.target
```

可以看出停止时候也是将规则保存在 `/etc/sysconfig/ipvsadm`文件中，由于该文件不存在，所以第一次需要手动保存创建文件。然后再添加到开机启动

```
[root@lvs-director ~]# ipvsadm-save -n > /etc/sysconfig/ipvsadm
[root@lvs-director ~]# systemctl enable ipvsadm
```

