MySQL学习笔记（Day046-047：LoadBalance）

MySQL学习

一. LoadBalance – LVS

> LVS官网介绍

1.1. LVS介绍

LVS全称为 Linux Virtual Server，LVS本质上是一个内核模块（*ip_vs.so*），因为LVS的负载调度技术是在Linux内核中实现。

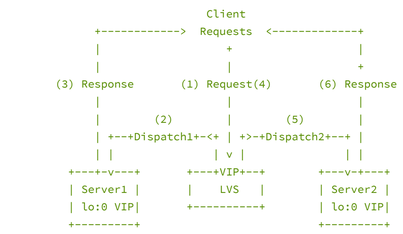
```
# 如果以下命令什么都没有显示，可能是你的lvs模块还没有加载，
# 在运行keepaliaved之后就会出现。
Shell> lsmod | grep ^ip_vs
ip_vs                 125220  0

Shell> modprobe -l | grep ipvs
kernel/net/netfilter/ipvs/ip_vs.ko
kernel/net/netfilter/ipvs/ip_vs_rr.ko
kernel/net/netfilter/ipvs/ip_vs_wrr.ko
kernel/net/netfilter/ipvs/ip_vs_lc.ko
kernel/net/netfilter/ipvs/ip_vs_wlc.ko
kernel/net/netfilter/ipvs/ip_vs_lblc.ko
kernel/net/netfilter/ipvs/ip_vs_lblcr.ko
kernel/net/netfilter/ipvs/ip_vs_dh.ko
kernel/net/netfilter/ipvs/ip_vs_sh.ko
kernel/net/netfilter/ipvs/ip_vs_sed.ko
kernel/net/netfilter/ipvs/ip_vs_nq.ko
kernel/net/netfilter/ipvs/ip_vs_ftp.ko
kernel/net/netfilter/ipvs/ip_vs_pe_sip.ko
```

LVS的部署模式大致分成3种：

1. **NAT**
2. **DR（Director Server）**
3. **Tunnel**

我们常用的部署模式为 DR模式 ，且 DR模式 的 性能 也是三者中 最好 的。

1.2. LVS–DR部署模式



如上图所示，LVS 、 Server1 和 Server2 需要在同一个VLAN之中，且在同一个IP段内。

1.3. LVS–DR模式的数据流转

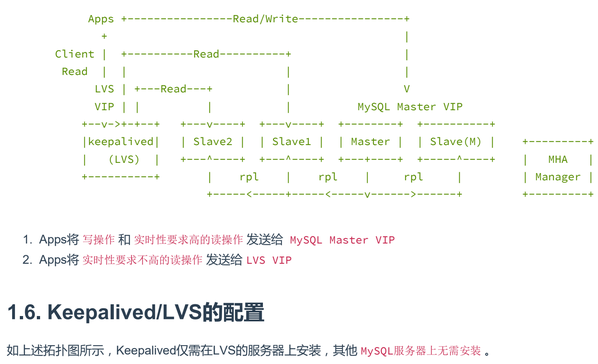
- (1)：Client-1 发送 请求（Request）给 LVS 的 虚拟IP ；
- (2)：LVS 调度给Server1 ，Server1对 请求（Request）进行处理；
- (3)：Server1 处理完请求 任务（*比如MySQL的查询*）后， 响应（Response）给 Client-1 ；
- (4)：此时 Client-2 发送 请求（Request）给LVS的 虚拟IP ；
- (5)：LVS 调度给Server2 ，Server2对 请求（Request）进行处理；
- (6)：Server2 处理完请求 任务（*比如MySQL的查询*）后， 响应（Response）给 Client-2 ；

1.4. LVS–DR模式的原理

DR模式实现的三个关键点：

1. **MAC地址替换**（*ARP欺骗*）
   - 将发送给LVS 请求 包的 目标MAC地址 替换 为某一个（*由调度器根据算法调度分配*）Server的MAC地址 ；
   - 然后将这个 修改过目标MAC地址的请求包 发送给对应的Server；
2. **虚拟IP**
   - 虚拟IP存在于LVS上，用于接受用户的请求；
   - 同时虚拟IP还绑定在 server 的 lo:0 上（*loopback口的别名*）
     - 假如 没有 绑定虚拟IP在 lo:0 上：
       1. 结合 MAC地址替换 ，此时数据包调度给了某一个server；
       2. server收到这个数据包后，首先 检查目标MAC是否是自己 的，由于之前做了MAC地址替换，所以检查的结果为真；
       3. 然后 检查目标IP ，由于 虚拟IP没有绑定在lo:0 上，即检查结果为假，则server就会丢弃该数据包。
3. **lo:0接口的ARP抑制**
   - 由于server上绑定虚拟IP在lo:0上，那此时看来，同一个网络（VLAN）中存在多个同样的IP，那不就IP地址冲突了么？
     - IP是否冲突主要看在一个网络（VLAN）内询问某一个IP地址是否有多台主机应答（*ARP广播*）；
   - 在server的 lo:0 上 启用ARP抑制 ，这样一来当网络中有设备 询问虚拟IP地址 时， server将不做应答 ， 此时只有LVS进行应答 ，确保不会引起冲突（*即 IP-MAC 的映射关系在该网络中唯一*）

1.5. LVS与MHA集群部署拓扑



1. Apps将 写操作 和 实时性要求高的读操作 发送给 MySQL Master VIP
2. Apps将 实时性要求不高的读操作 发送给 LVS VIP

1.6. Keepalived/LVS的配置

如上述拓扑图所示，Keepalived仅需在LVS的服务器上安装，其他 MySQL服务器上无需安装 。

1.6.1. Keepalived

由于LVS本质上是一个内核模块，所以我们需要借助一些 用户层工具 去配置管理，主要有以下两个工具：

1. **keepalived**
2. **ipvsadm**

以上两种工具都是用来 配置LVS 的，大致区别如下

1. ipvsadm是命令行工具 ，每次 重启 都需要 重新配置 （或者写入开机脚本中），但是配置参数十分灵活；
2. keepalived 是一个服务（*Daemon*），通过编辑配置文件（ /etc/keepalived/keepalived.conf ），可以很方便的对LVS的参数做调整，且系统重启后不会丢失;
3. Keepalived 还有 多机热备 的功能（ *VRRP协议* ），可以将多台服务器组成Master-Slave集群，通过 虚拟IP 的方式对外提供服务，使得LVS不会成为单点故障；
4. Keepalived 还可以配置后端server检测，当发现服务不可用时，将该server从转发列表中剔除，具有健康检查机制；

我们这里选择使用Keepalived对LVS进行配置。

1.6.2. 安装keepalived

```
Shell> yum install keepalived
```

1.6.3. 配置keepalived

1. 单台 或者 Master 的配置文件如下

```
! Configuration File for keepalived
## /etc/keepalived/keepalived.conf
##
## 全局配置部分，这里可以配置邮件告警
## 比较重要的参数是router_id，拥有同一个router_id的keepalived被认为是一组的
#
global_defs {
   #notification_email {
   #  acassen@firewall.loc
   #  failover@firewall.loc
   #  sysadmin@firewall.loc
   #}
   #notification_email_from Alexandre.Cassen@firewall.loc
   #smtp_server 192.168.200.1
   #smtp_connect_timeout 30

   router_id LVS_MYSQL
}

##
## VRRP组播协议部分参数配置：
## 比如：
##      发送的主机号：virtual_router_id
##      优先级：priority
##      通告间隔： advert_int
##      初始状态：state
##      是否抢占：nopreempt
##      认证：authentication
##      虚拟IP：virtual_ipaddress
##
vrrp_instance VI_1 {
    state BACKUP
    interface eth0
    virtual_router_id 60
    priority 100
    nopreempt
    advert_int 5
    authentication {
        auth_type PASS
        auth_pass 1234
    }
    virtual_ipaddress {
        172.18.14.100
    }
}

##
## LVS的转发配置
##

virtual_server 172.18.14.100 3306 {
    delay_loop 6
    lb_algo rr
    lb_kind DR
    # 为了兼容老的内核
    # nat_mask 255.255.255.0
    # 保持连接的时间，0：不保持
    persistence_timeout 0
    protocol TCP

    ##  转发主机1 -- Slave1
    real_server 172.18.14.71 3306 {
        weight 1
        TCP_CHECK {
            connect_timeout 3
            nb_get_retry 3
            delay_before_retry 3
            connect_port 3306
        }
    }

    ##  转发主机2 -- Slave2
    real_server 172.18.14.72 3306 {
        weight 1
        TCP_CHECK {
            connect_timeout 3
            nb_get_retry 3
            delay_before_retry 3
            connect_port 3306
        }
    }
}
```

2. 如果你有备机，组成双机热备，则可以根据上述配置进行修改，修改如下：

```
! Configuration File for keepalived

global_defs {
   #notification_email {
   #  acassen@firewall.loc
   #  failover@firewall.loc
   #  sysadmin@firewall.loc
   #}
   #notification_email_from Alexandre.Cassen@firewall.loc
   #smtp_server 192.168.200.1
   #smtp_connect_timeout 30
   ## 必须同Master【一致】
   router_id LVS_MYSQL
}

##
## VRRP组播协议部分参数配置：
## 比如：
##      发送的主机号：virtual_router_id
##      优先级：priority
##      通告间隔： advert_int
##      初始状态：state
##      是否抢占：nopreempt
##      认证：authentication
##      虚拟IP：virtual_ipaddress
##
vrrp_instance VI_1 {
    state BACKUP
    interface eth0
    ## virtual_router_id必须和Master 【不一致】
    virtual_router_id 70
    ## priority 必须比 Master的值 【小】
    priority 90
    nopreempt
    advert_int 5
    ## authentication设置必须 【一致】
    authentication {
        auth_type PASS
        auth_pass 1234
    }
    ## 虚拟IP必须 【一致】
    virtual_ipaddress {
        172.18.14.100
    }
}

######### 下面的配置必须和 Master 【一致】 #######
##
## LVS的转发配置
##

virtual_server 172.18.14.100 3306 {
    delay_loop 6
    lb_algo rr
    lb_kind DR
    # 为了兼容老的内核
    # nat_mask 255.255.255.0
    persistence_timeout 0
    protocol TCP

    ##  转发主机1 -- Slave1
    real_server 172.18.14.71 3306 {
        weight 1
        TCP_CHECK {
            connect_timeout 3
            nb_get_retry 3
            delay_before_retry 3
            connect_port 3306
        }
    }

    ##  转发主机2 -- Slave2
    real_server 172.18.14.72 3306 {
        weight 1
        TCP_CHECK {
            connect_timeout 3
            nb_get_retry 3
            delay_before_retry 3
            connect_port 3306
        }
    }
}
```

重要参数解释：

1. **`state BACKUP`** ：这个是keepalived的初始化状态， `Keepalived的Master` 和 `Keepalived的Slave` 上都配置成 `BACKUP` 表示两台主机都默认不认为自己的是 `Master` ，都 `默认不会去启用虚拟IP` ；
2. **`priority`** ：这个参数在 `state` 都配置成 `BACKUP` 的时候就很重要了，因为都是 `BACKUP` ，所以只能通过 `priority` 的大小来决定谁是 `Master` ， `priority的值大的那台服务器` ，就是 `Master` ；
3. **`nopreempt`** ： 这个参数表示如果 `Keepalived的Master` 宕机了，原来 `Keepalived的Slave` 变成了New Master（虚拟IP在New Master上了），而此时如果 `Keepalived的Master` 又恢复了， `不会抢占` New Master的状态和虚拟IP地址；

上面三个参数配合，可以减少VIP飘动的次数，避免网络来回震动。

> **注意，如果有一个Keepalived配置了 `state MASTER` ，则会忽略 `nopreempt` 参数，会 `强制夺回` Master的状态和虚拟IP**

1.6.4. keepalived的启动

```
Shell> service keepalived start
# 然后可以通过 查看/var/log/messages或者journalctl  -xe （centos 7），观察keepalived的启动过程，检测realserver的状态，以及最终的过程。
```

1.6.5. LVS的虚拟IP

在Keepalived启动完成后，就可以产看一下LVS的虚拟IP地址了，但是通过 `ifconfig` 是查看不到的，需要通过 `ip addr` 才能进行查看：
ifconfig和ip addr的区别

```
Shell> ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 16436 qdisc noqueue state UNKNOWN
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP qlen 1000
    link/ether 52:54:a0:39:76:fb brd ff:ff:ff:ff:ff:ff
    inet 172.18.14.68/24 brd 172.18.14.255 scope global eth0
    inet 172.18.14.100/32 scope global eth0 # 这个就是我们之前配置的虚拟IP
    inet6 fe80::5054:a0ff:fe39:76fb/64 scope link
       valid_lft forever preferred_lft forever
```

1.6.6. server绑定虚拟IP

将以下脚本保存成 lvs_realserver.sh ，将该 脚本配合虚拟IP 加入到 开机脚本 中，以便开机可以自动运行。

```
#!/bin/bash
# Filename: lvs_realserver.sh
#    Usage: sh lvs_realserver.sh 172.18.14.100 start
#
# echo "sh /usr/local/bin/lvs_realserver.sh 172.18.14.100 start" >> /etc/rc.local
#

VIP1=$1

case "$2" in
    start)
        echo " start LVS of REALServer"
        /sbin/ifconfig lo:0 $VIP1 broadcast $VIP1 netmask 255.255.255.255 up
        echo "1" >/proc/sys/net/ipv4/conf/lo/arp_ignore
        echo "2" >/proc/sys/net/ipv4/conf/lo/arp_announce
        echo "1" >/proc/sys/net/ipv4/conf/all/arp_ignore
        echo "2" >/proc/sys/net/ipv4/conf/all/arp_announce
        ;;
    stop)
        /sbin/ifconfig lo:0 down
        echo "close LVS Directorserver"
        echo "0" >/proc/sys/net/ipv4/conf/lo/arp_ignore
        echo "0" >/proc/sys/net/ipv4/conf/lo/arp_announce
        echo "0" >/proc/sys/net/ipv4/conf/all/arp_ignore
        echo "0" >/proc/sys/net/ipv4/conf/all/arp_announce
        ;;
    *)
        echo "Usage: $0 {start|stop}"
        exit 1
        ;;
esac

exit 0
```

上述脚本中实现了 将虚拟IP绑定在lo:0 上，同时做了 ARP抑制 。

- 由于 slave1 和 slave2 上是 LVS 的两个 realserver ，所以需要运行 lvs_realserver.sh
- Slave(M)也可以加入LVS的 realserver

1.7. LVS与MHA集群测试

找一台client访问LVS的虚拟IP，测试结果如下：

```
Shell> mysql -u root -p123 -h 172.18.14.100 -e "select @@hostname;"
mysql: [Warning] Using a password on the command line interface can be insecure.
+------------+
| @@hostname |
+------------+
| Slave1     |
+------------+
Shell> mysql -u root -p123 -h 172.18.14.100 -e "select @@hostname;"
mysql: [Warning] Using a password on the command line interface can be insecure.
+------------+
| @@hostname |
+------------+
| Slave2     |
+------------+
Shell> mysql -u root -p123 -h 172.18.14.100 -e "select @@hostname;"
mysql: [Warning] Using a password on the command line interface can be insecure.
+------------+
| @@hostname |
+------------+
| Slave1     |
+------------+
Shell> mysql -u root -p123 -h 172.18.14.100 -e "select @@hostname;"
mysql: [Warning] Using a password on the command line interface can be insecure.
+------------+
| @@hostname |
+------------+
| Slave2     |
+------------+
```

> 注意 persistence_timeout 参数的值如果 不为0 ，则会有 会话保持 效果（一段时间内访问都是同一台机器）。

1.8. 关于persistence_timeout的问题

当这个值被设置为 大于0 的时候，就会有 会话保持 的效果，但是这个参数容易让人 误以为是会话保持多久的时间 。

persistence_timeout 参数其实是LVS内部 connection倒计时的一个初始值 ，需要配合设置其他参数才能最终确认一个会话保持的时间。

1. connection 由 client-ip:port 和 realserver-ip:port 组成；
2. 当一个 client-ip 第一次连接到LVS的时候，除了产生原来访问服务的 标记为ESTABLISHED 的connection，还会额外产生一个标记为 None 的connection，这个 None 的connection 初始的倒计时时间 就是我们设置的 persistence_timeout
3. 当 同一个client-ip 再次访问LVS时，发现存在 标记为None 的connection（*同一个 client-ip* ）时，则又会产生一个 标记为ESTABLISHED 的connection，该connection的超时时间可以通过 ipvsadm -l --timeout 进行查看（*安装：yum install ipvsadm* ）

    ```
    Shell> ipvsadm -l --timeout
    Timeout (tcp tcpfin udp): 900 120 300  # TCP默认超时是15分钟
    ```

4. 当 None 的connection倒计时为0的时候，会查看是否有 同一个client-ip 的connection存在，如果存在，则从 60 开始倒计时（这个60是个固定值，和你设置的 persistence_timeout 没有关系），直到没有 ESTABLISHED 的connection时，标记为 None 的connection自动删除。

所以从LVS的策略看，我们设置的 persistence_timeout 太小的话，并没什么用，只有 初始 检测时，是受我们设置的值影响，如果 Timeout 设置的太大，之后的倒计时，都是60秒。

如果希望会话保持时间和我们设置的 persistence_timeout 一致或者接近，需要修改默认的 Timeout 让其 小于persistence_timeout ：

```
Shell> ipvsadm --set 13 5 13
# 然后需要将persistence_timeout 修改成15
```

> 注意：以上 ipvsadm 的修改在重启后均会丢失，需要放入开机脚本中。
>
> **如果在超时时间范围内，有大量访问（*None的connection永远检测有ESTABLISHED*），其实还是负载在一台后端server上**

二. LoadBalance – HAProxy

HAProxy是 七层代理 ，在使用HAProxy后，在MySQL上 看不到Apps的源IP地址 ，看到的是HAProxy地址，而 MySQL的权限访问设置是和IP地址有关 ，这样就导致了MySQL无法 针对应用 进行区分权限了，所以使用的时候要注意。

2.1. HAProxy的安装

```
Shell> yum install haproxy
```

2.2. HAProxy的配置

2.2.1. haproxy.cfg

将以下配置文件保存为 /etc/haproxy/haproxy.cfg

```
# 全局配置参数
global
    log 127.0.0.1 local0 notice
    user haproxy
    group haproxy

# 一些默认参数
defaults
    log global
    retries 3
    option  dontlognull
    option redispatch
    maxconn 2000
    timeout connect 3000
    timeout server 5000
    timeout client 5000

# 这个是我们定义的负载均衡的配置
listen mysql-lb1
    # 绑定的IP和端口
    bind 172.18.14.68:3306
    # 模式是TCP
    mode tcp
    # 通过mysql连接去检测mysql是否可以访问
    option mysql-check user haproxy_check
    # 负载均衡算法是 轮询
    balance roundrobin
    # 下面两个就是后端被访问的server
    server mysql_1 172.18.14.71:3306 weight 1 check
    server mysql_2 172.18.14.72:3306 weight 1 check

# 自带的监控服务器的配置
# 监控服务的端口是 8888
listen stats *:8888
        # 监控模式是http
        mode http
        option httpclose
        balance roundrobin
        stats uri /
        stats realm Haproxy\ Statistics
        # 监控的用户名和密码
        stats auth myadmin:myadmin
```

2.2.2. 添加haproxy_check用户

将以下SQL语句在Master端执行，通过复制功能，传递到Slave上。

```
drop user haproxy_check@'172.18.14.68';
create user haproxy_check@'172.18.14.68';
grant usage on *.* to haproxy_check@'172.18.14.68';
```

2.2.3. 配置日志

注意：**该方法仅在 CentOS 6.X 上使用**， CentOS 7.x 安装 HAProxy 后可用 systemctl status haproxy 进行查看。

将以下文件保存为 /etc/rsyslog.d/49-haproxy.conf

```
# Create an additional socket in haproxy's chroot in order to allow logging via
# /dev/log to chroot'ed HAProxy processes
$AddUnixListenSocket /var/lib/haproxy/dev/log
$ModLoad imudp
$UDPServerRun 514
local3.*     /var/log/haproxy.log

# Send HAProxy messages to a dedicated logfile
if $programname startswith 'haproxy' then /var/log/haproxy.log
&~
```

然后重启 rsyslog 服务

```
Shell> service  rsyslog restart
Shutting down system logger:                               [  OK  ]
Starting system logger:                                    [  OK  ]
```

2.3. 启动HAProxy

```
Shell> service haproxy start
Shell> netstat -tunlp | grep haproxy
tcp        0      0 0.0.0.0:8888                0.0.0.0:*                   LISTEN      1622/haproxy
# 监控端口
tcp        0      0 172.18.14.68:3306           0.0.0.0:*                   LISTEN      1622/haproxy
# 自定义的服务端口
udp        0      0 0.0.0.0:42626               0.0.0.0:*                               1622/haproxy
# 用于发送日志
```

2.4. HAProxy测试

```
Shell> mysql -u root -p123 -h 172.18.14.68 -e "select @@hostname;"
mysql: [Warning] Using a password on the command line interface can be insecure.
+------------+
| @@hostname |
+------------+
| Slave1     |
+------------+
Shell> mysql -u root -p123 -h 172.18.14.68 -e "select @@hostname;"
mysql: [Warning] Using a password on the command line interface can be insecure.
+------------+
| @@hostname |
+------------+
| Slave2     |
+------------+
Shell> mysql -u root -p123 -h 172.18.14.68 -e "select @@hostname;"
mysql: [Warning] Using a password on the command line interface can be insecure.
+------------+
| @@hostname |
+------------+
| Slave1     |
+------------+
Shell> mysql -u root -p123 -h 172.18.14.68 -e "select @@hostname;"
mysql: [Warning] Using a password on the command line interface can be insecure.
+------------+
| @@hostname |
+------------+
| Slave2     |
+------------+
```

某台从机上的显示如下：

```
mysql> show processlist;
+------+-------------+--------------------+------+---------+------+-----------------------------------------------------------------+------------------+
| Id   | User        | Host               | db   | Command | Time | State                                                           | Info             |
+------+-------------+--------------------+------+---------+------+-----------------------------------------------------------------+------------------+
|    1 | system user |                    | NULL | Connect | 7302 | Waiting for master to send event                                | NULL             |
|    2 | system user |                    | NULL | Connect | 4650 | Slave has read all relay log; waiting for more updates          | NULL             |
|    3 | system user |                    | NULL | Connect | 4649 | Waiting for an event from Coordinator                           | NULL             |
|    4 | system user |                    | NULL | Connect | 7302 | Waiting for an event from Coordinator                           | NULL             |
|    5 | system user |                    | NULL | Connect | 7302 | Waiting for an event from Coordinator                           | NULL             |
|    6 | system user |                    | NULL | Connect | 7302 | Waiting for an event from Coordinator                           | NULL             |
| 3344 | root        | localhost          | NULL | Query   |    0 | starting                                                        | show processlist |
| 3403 | root        | 172.18.14.68:54973 | NULL | Sleep   |    3 |                                                                 | NULL             |
+------+-------------+--------------------+------+---------+------+-----------------------------------------------------------------+------------------+
8 rows in set (0.00 sec)
```

> 从 Id 3403 可以看出来，HAProxy确实把 应用的源IP替换成了自己的IP ，在使用时注意权限问题。