角色	IP 地址	主机名	ServerID
Primary Master	192.168.200.107	server01	1
Secondary Master	192.168.200.108	server02	2
Slave1	192.168.200.109	server03	3
Slave2	192.168.200.104	server04	4
Manager	192.168.200.106	server05	-
LVS Matser	192.168.200.110	localhost	-
LVS Backup	192.168.200.105	localhost	-

```
[root@localhost ~]# hostname server01
[root@localhost ~]# bash
[root@localhost ~]# hostname server02
[root@localhost ~]# bash
[root@localhost ~]# hostname server03
[root@localhost ~]# bash
[root@localhost ~]# hostname server04
[root@localhost ~]# bash
[root@localhost ~]# hostname server05
[root@localhost ~]# bash
```

三、安装 MHA node

1.配置epel源

```
[root@web yum.repos.d]#rpm -ivh https://mirrors.aliyun.com/epel/epel-release-latest-7.noarch.rpm
[root@web yum.repos.d]# ls
backup  Centos-7.repo  epel.repo  epel-testing.repo
```

```
[root@server05 ~]# vim /etc/hosts
192.168.200.107 server01
192.168.200.108 server02
192.168.200.109 server03
192.168.200.104 server04
192.168.200.106 server05
[root@server01 ~]#scp /etc/hosts 192.168.200.108:/etc/
[root@server01 ~]#scp /etc/hosts 192.168.200.109:/etc/
[root@server01 ~]#scp /etc/hosts 192.168.200.104:/etc/
[root@server01 ~]#scp /etc/hosts 192.168.200.106:/etc/
[root@server01 ~]#iptables -F
[root@server01 ~]#systemctl stop firewalld
[root@server01 ~]#setenforce 0
```

2.所有主机装安装 MHA node 及相关 perl 依赖包

```
[root@server01 ~]#yum install -y perl-DBD-MySQL.x86_64 perl-DBI.x86_64 perl-CPAN perl-ExtUtils-CBuilder perl-ExtUtils-MakeMaker
```

注意：安装后建议检查一下所需软件包是否全部安装

```
[root@server01 ~]#rpm -q perl-DBD-MySQL.x86_64 perl-DBI.x86_64 perl-CPAN perl-ExtUtils-CBuilder perl-ExtUtils-MakeMaker
```

3.所有主机装上安装 MHA Node

注意：不要同时用全部会话同步，输入rz，容易卡死

```
[root@server01 ~]# rz
[root@server01 ~]# tar xf mha4mysql-node-0.56.tar.gz
[root@server01 ~]# cd mha4mysql-node-0.56/
[root@server01 mha4mysql-node-0.56]# perl Makefile.PL
[root@server01 mha4mysql-node-0.56]# make && make install
```

4.MHA Node 安装完 后会在 /usr/local/bin

```
[root@server01 mha4mysql-node-0.56]# ls -l /usr/local/bin/
总用量 40
-r-xr-xr-x 1 root root 16346 4月  29 01:09 apply_diff_relay_logs
-r-xr-xr-x 1 root root  4807 4月  29 01:09 filter_mysqlbinlog
-r-xr-xr-x 1 root root  7401 4月  29 01:09 purge_relay_logs
-r-xr-xr-x 1 root root  7395 4月  29 01:09 save_binary_logs
```

四、安装 MHA Manger

注意：安装 MHA Manger 之前也需要安装 MHA Node

1.首先安装 MHA Manger 依赖的 perl 模块（我这里使用 yum 安装）

```
[root@server01 ~]# yum install -y perl perl-Log-Dispatch perl-Parallel-ForkManager perl-DBD-MySQL perl-DBI perl-Time-HiRes
[root@server01 ~]# yum -y install perl-Config-Tiny-2.14-7.el7.noarch.rpm      #失败
[root@server01 ~]# cpan install Config::Tiny        # 用这个下载
[root@server01 ~]#rpm -q perl cpan perl-Log-Dispatch perl-Parallel-ForkManager perl-DBD-MySQL perl-DBI perl-Time-HiRes perl-Config-Tiny   （cpan即可）
```

注意：之前时候 perl-Config-Tiny.noarch 没有安装成功，

后来用 cpan([root@server01 ~]# cpan install Config::Tiny)

2. 安装 MHA Manger

```
tar xf mha4mysql-manager-0.56.tar.gz
cd mha4mysql-manager-0.56/
perl Makefile.PL
make && make install
```

3.安装完成后会有以下脚本文件

```
ls -l /usr/local/bin/
```

五、配置 SSH 密钥对验证

服务器之间需要实现密钥对验证。关于配置密钥对验证可看下面步骤。但是有一点需要注意：不能禁止 password 登陆，否则会出现错误

1. 服务器先要生成一个密钥对

2. 把自己的公钥传给对方

Server05(192.168.200.106) 上:

```
ssh-keygen -t rsa
ssh-copy-id -i /root/.ssh/id_rsa.pub root@192.168.200.107
ssh-copy-id -i /root/.ssh/id_rsa.pub root@192.168.200.108
ssh-copy-id -i /root/.ssh/id_rsa.pub root@192.168.200.109
ssh-copy-id -i /root/.ssh/id_rsa.pub root@192.168.200.104
```

注意：

Server05 需要连接每个主机测试，因为第一次连接入的时候需要输入yes ，

影响后期故障切换时，对于每的个主机的 SSH 控制

```
ssh server01
ssh server02
ssh server03
ssh server04
```

Primary Master(192.168.200.107):

```
ssh-keygen -t rsa
ssh-copy-id -i /root/.ssh/id_rsa.pub root@192.168.200.108
ssh-copy-id -i /root/.ssh/id_rsa.pub root@192.168.200.109
ssh-copy-id -i /root/.ssh/id_rsa.pub root@192.168.200.104

ssh server02
ssh server03
ssh server04
```

Primary Master(192.168.200.108):

```
ssh-keygen -t rsa
ssh-copy-id -i /root/.ssh/id_rsa.pub root@192.168.200.107
ssh-copy-id -i /root/.ssh/id_rsa.pub root@192.168.200.109
ssh-copy-id -i /root/.ssh/id_rsa.pub root@192.168.200.104
ssh server01
ssh server03
ssh server04
```

Primary Master(192.168.200.109):

```
ssh-keygen -t rsa
ssh-copy-id -i /root/.ssh/id_rsa.pub root@192.168.200.107
ssh-copy-id -i /root/.ssh/id_rsa.pub root@192.168.200.108
ssh-copy-id -i /root/.ssh/id_rsa.pub root@192.168.200.104
ssh server01
ssh server02
ssh server04
```

Primary Master(192.168.200.104):

```
ssh-keygen -t rsa
ssh-copy-id -i /root/.ssh/id_rsa.pub root@192.168.200.107
ssh-copy-id -i /root/.ssh/id_rsa.pub root@192.168.200.108
ssh-copy-id -i /root/.ssh/id_rsa.pub root@192.168.200.109
ssh server01
ssh server02
ssh server03
```

六、安装 mysql

107-104　主机上的操作：

```
yum -y install mariadb mariadb-server mariadb-devel
systemctl start mariadb
```

```
netstat -lnpt | grep :3306
```

设置数据库初始密码（后续操作中使用）

```
mysqladmin -u root password 123456
```

七、 搭建主从复制环境

注意：

binlog-do-db 和 replicate-ignore-db 设置必须相同。

MHA 在启动时候会检测过滤规则，如果过滤规则不同，MHA 将不启动监控和故障转移功能

修改 mysql 主机的配置文件

Primary Master(192.168.200.107):

```
vim /etc/my.cnf
[mysqld]
server-id = 1
log-bin=master-bin
log-slave-updates=true
relay_log_purge=0

[root@server01 ~]# systemctl restart mariadb
```

Secondary Master(192.168.200.108):

```
vim /etc/my.cnf
[mysqld]
server-id=2
log-bin=master-bin
log-slave-updates=true
relay_log_purge=0

[root@server02 ~]#systemctl restart mariadb
```

slave1(192.168.200.109):

```
vim /etc/my.cnf
[mysqld]
server-id=3
log-bin=mysql-bin
relay-log=slave-relay-bin
log-slave-updates=true
relay_log_purge=0
```

```
[root@server03~]#systemctl restart mariadb
```

slave2(192.168.200.104):

```
vim /etc/my.cnf
[mysqld]
server-id=4
log-bin=mysql-bin
relay-log=slave-relay-bin
log-slave-updates=true
relay_log_purge=0

[root@server04 ~]#systemctl restart mariadb
```

如果Primary Master（192.168.200.107）上之前运行过，

存在旧的数据，需要先对其进行备份

```
mysqldump --master-data=2 --single-transaction -R --triggers -A > all.sql
```

参数解释:

--master-data= 2　　备份时刻记录master的Binlog位置和Position

--single transaction　　获取一致性快照

-R　　备份存储过程和函数

-triggres　　备份触发器

-A　　备份所有的库

```
[root@server01 ~]# mysql -uroot -p123456
```

mysql 服务器创建复制授权

```
grant replication slave on *.* to 'repl'@'192.168.200.%' identified by '123456';
flush privileges;
```

查看主库备份时的binlog名称和位置

```
MariaDB [(none)]> show master status;
```

主从复制:

```
stop slave;
CHANGE MASTER TO
MASTER_HOST='192.168.200.107',
MASTER_USER='repl',
MASTER_PASSWORD='123456',
MASTER_LOG_FILE='master-bin.000002',
MASTER_LOG_POS=474;
```

```
start slave;
show slave status\G
```

三台 slave 服务器设置 read_only 状态

```
set global read_only=1;
```

创建监控用户 （107-104 主机上的操作 ）

```
grant all privileges on *.* to 'root'@'192.168.200.%' identified by '123456';
flush privileges;
```

每个服务器要为自己的主机名授权：

```
grant all privileges on *.* to 'root'@'server04' identified by '123456';
flush privileges;
```

到这里整个 mysql 主从集群环境已经搭建完毕~

八、配置 MHA 环境

1.创建 MHA 的工作目录 及 相关配置文件

Server05(192.168.200.106)： 在软件包解压后的目录里面有样例配置文件

```
mkdir /etc/masterha
cp mha4mysql-manager-0.56/samples/conf/app1.cnf /etc/masterha
```

2.修改 app1.cnf 配置文件

/usr/local/bin/master_ip_failover 脚本需要根据自己环境修改 ip 和网卡名称等

```
vim /etc/masterha/app1.cnf
master_binlog_dir=/var/lib/mysql
master_ip_failover_script= /usr/local/bin/master_ip_failover
password=123456
user=root
ping_interval=1
remote_workdir=/tmp
repl_password=123456
repl_user=repl

[server1]
hostname=server01
port=3306

[server2]
candidate_master=1
```

```
check_repl_delay=0
hostname=server02
port=3306


[server3]
hostname=server03
port=3306

[server4]
hostname=server04
port=3306
```

3. 配置故障转移脚本

```
[root@server05 ~]# vim /usr/local/bin/master_ip_failover
#!/usr/bin/env perl
use strict;
use warnings FATAL => 'all';
use Getopt::Long;
my (
$command, $ssh_user, $orig_master_host, $orig_master_ip,
$orig_master_port, $new_master_host, $new_master_ip, $new_master_port,
);
my $vip = '192.168.200.100'; #  写入 VIP
my $key = "1"; #非 keepalived  方式切换脚本使用的
my $ssh_start_vip = "/sbin/ifconfig ens32:$key $vip";
my $ssh_stop_vip = "/sbin/ifconfig ens32:$key down"; # 那么这里写服务的开关命令
$ssh_user = "root";
GetOptions(
'command=s' => \$command,
'ssh_user=s' => \$ssh_user,
'orig_master_host=s' => \$orig_master_host,
'orig_master_ip=s' => \$orig_master_ip,
'orig_master_port=i' => \$orig_master_port,
'new_master_host=s' => \$new_master_host,
'new_master_ip=s' => \$new_master_ip,
'new_master_port=i' => \$new_master_port,
);
```

```
exit &main();
sub main {
print "\n\nIN SCRIPT TEST====$ssh_stop_vip==$ssh_start_vip===\n\n";
if ( $command eq "stop" || $command eq "stopssh" ) {
# $orig_master_host, $orig_master_ip, $orig_master_port are passed.
# If you manage master ip address at global catalog database,
# invalidate orig_master_ip here.
my $exit_code = 1;
#eval {
# print "Disabling the VIP on old master: $orig_master_host \n";
# &stop_vip();
# $exit_code = 0;
#};
eval {
print "Disabling the VIP on old master: $orig_master_host \n";
#my $ping=`ping -c 1 10.0.0.13 | grep "packet loss" | awk -F',' '{print $3}' | awk '{print
$1}'`;
#if ( $ping le "90.0%"&& $ping gt "0.0%" ){
#$exit_code = 0;
#}
#else {
&stop_vip();
# updating global catalog, etc
$exit_code = 0;
#}
};
if ($@) {
warn "Got Error: $@\n";
exit $exit_code;
}
exit $exit_code;
}
elsif ( $command eq "start" ) {
# all arguments are passed.
# If you manage master ip address at global catalog database,
# activate new_master_ip here.
# You can also grant write access (create user, set read_only=0, etc) here.
my $exit_code = 10;
eval {
```

```
print "Enabling the VIP - $vip on the new master - $new_master_host \n";
&start_vip();
$exit_code = 0;
};
if ($@) {
warn $@;
exit $exit_code;
}
exit $exit_code;
}
elsif ( $command eq "status" ) {
print "Checking the Status of the script.. OK \n";
`ssh $ssh_user\@$orig_master_ip \" $ssh_start_vip \"`;
exit 0;
}
else {
&usage();
exit 1;
}
}
# A simple system call that enable the VIP on the new master
sub start_vip() {
`ssh $ssh_user\@$new_master_host \" $ssh_start_vip \"`;
}
# A simple system call that disable the VIP on the old_master
sub stop_vip() {
`ssh $ssh_user\@$orig_master_host \" $ssh_stop_vip \"`;
}
sub usage {
print
"Usage: master_ip_failover --command=start|stop|stopssh|status --orig_master_host=host
--orig_master_ip=ip --orig_master_port=port --
new_master_host=host --new_master_ip=ip --new_master_port=port\n"; }
[root@server05 ~]# chmod +x /usr/local/bin/master_ip_failover
```

4.设置库 从库 relay log 的清除方式((108- 104)) :

```
mysql -uroot -p123456 -e 'set global relay_log_purge=0;'
```

注意:

MHA 在故障切换的过程中，从库的恢复过程依赖于 relay log 的相关信息，所以这里要

将 relay log 的自动清除设置为 OFF，采用手动清除 relay log 的方式。在默认情况下，从服务器上的中继日志会在 SQL 线程执行完毕后被自动删除。但是在 MHA 环境中，这些中继日志在恢复其他从服务器时可能会被用到，因此需要禁用中继日志的自动清除功能。定期清除中继日志需要考虑到复制延时的问题。在 ext3 的文件系统下，删除大的文件需要一定的时间，会导致严重的复制延时。为了避免复制延时，需要暂时为中继日志创建硬链接，因为在 linux 系统中通过硬链接删除大文件速度会很快。（在 mysql 数据库中，删除大表时，通常也采用建立硬链接的方式）

5.配置从库 （108- 104 ）relay_log 清除脚本加入计划任务 ---暂时不用配置

MHA 节点中包含了 pure_relay_logs 命令工具，它可以为中继日志创建硬链接，执行 SET GLOBAL relay_log_purge=1,等待几秒钟以便 SQL 线程切换到新的中继日志，再执行 SET GLOBAL relay_log_purge=0

```
vim purge_relay_log.sh
#!/bin/bash
user=root
passwd=123456 # 注意：数据库要有 密码, 填自己所设置的密码就可以 , 前面设置过
port=3306
log_dir='/tmp'
work_dir='/tmp'
purge='/usr/local/bin/purge_relay_logs'
if [ ! -d $log_dir ]
then
mkdir $log_dir -p
fi
$purge --user=$user --password=$passwd --disable_relay_log_purge --port=$port
--workdir=$work_dir >> $log_dir/purge_relay_logs.log 2>&1
chmod +x purge_relay_log.sh
crontab -e
0 4 * * * /bin/bash /root/purge_relay_log.sh
```

pure_relay_logs 脚本参数如下所示:

--user mysql 用户名
--password mysql 密码
--port 端口号
--workdir 指定创建 relay log 的硬链接的位置，默认是/var/tmp，由于系统不同分区创建硬链接文件会失败，故需要执行硬链接具体位置，成功执行脚本后，硬链接的中继日志文件被删除
--disable_relay_log_purge 默认情况下，如果 relay_log_purge=1，脚本会什么都不清理，自

动退出，通过设定这个参数，当 relay_log_purge=1 的情况下会将 relay_log_purge 设置为0。清理 relay log 之后，最后将参数设置为 OFF

6.手动清除 中继 日志

```
purge_relay_logs --user=root --password=123456 --disable_relay_log_purge --port=3306 --workdir=/tmp
```

7.检查 MHA ssh 通信状态

```
[root@server05 ~]# masterha_check_ssh --conf=/etc/masterha/app1.cnf
```

最后会返回 successfully 表示没有问题

8.检查整个集群的状态

```
[root@server05 ~]# masterha_check_repl --conf=/etc/masterha/app1.cnf
MySQL Replication Health is OK.
```

返回 OK 表示也没有问题

九、VIP 配置管理

Master vip 配置有两种方式,

一种是通过 keepalived 或者 heartbeat 类似的软件的方式管理 VIP 的浮动,
另一种为通过命令方式管理。

通过命令方式管理 VIP 地址：

打开在前面编辑过的文件/etc/masterha/app1.cnf，检查如下行是否正确，再检查集群状态。

```
[root@server05 ~]# grep -n 'master_ip_failover_script' /etc/masterha/app1.cnf
6:master_ip_failover_script= /usr/local/bin/master_ip_failover
```

Primary Master(192.168.200.107)

```
[root@server01 ~]# ip a | grep ens32
2: ens32: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    inet 192.168.200.107/24 brd 192.168.200.255 scope global noprefixroute ens32
    inet 192.168.200.100/24 brd 192.168.200.255 scope global secondary ens32:1
```

Server05(192.168.200.106)修改故障转移脚本

```
[root@server05 ~]# head -13 /usr/local/bin/master_ip_failover
#!/usr/bin/env perl
```

```
use strict;
use warnings FATAL => 'all';
use Getopt::Long;
my (
$command, $ssh_user, $orig_master_host, $orig_master_ip,
$orig_master_port, $new_master_host, $new_master_ip, $new_master_port,
);
my $vip = '192.168.200.100'; #  写入 VIP
my $key = "1"; #非 keepalived  方式切换脚本使用的
my $ssh_start_vip = "/sbin/ifconfig ens32:$key $vip"; # 若是使用 keepalived
my $ssh_stop_vip = "/sbin/ifconfig ens32:$key down"; # 那么这里写服务的 开关命令
```

/usr/local/bin/master_ip_failover 文件的内容意思是当主库发生故障时，会触发 MHA 切换，MHA manager 会停掉主库上的 ens32:1 接口，触发虚拟 ip 漂移到备选从库，从而完成切换。

Server05(192.168.200.106) 检查 manager 状态

```
masterha_check_status --conf=/etc/masterha/app1.cnf
```

注意：如果正常会显示"PING_OK"，否则会显示"NOT_RUNNING"，代表 MHA 监控没有开启。

Server05(192.168.200.106) 开启 manager 监控

```
[root@server05 ~]# nohup masterha_manager --conf=/etc/masterha/app1.cnf --
remove_dead_master_conf --ignore_last_failover< /dev/null
>/var/log/masterha/app1/manager.log 2>&1 &
```

启动参数介绍：
--remove_dead_master_conf
该参数代表当发生主从切换后，老的主库的 ip 将会从配置文件中移除。
--manger_log
日志存放位置
--ignore_last_failover
在缺省情况下，如果 MHA 检测到连续发生宕机，且两次宕机间隔不足 8 小时的话，则不会进行 Failover，之所以这样限制是为了避免 ping-pong 效应。
该参数代表忽略上次 MHA 触发切换产生的文件，默认情况下，MHA 发生切换后会在日志目录，也就是上面我设置的/data 产生 app1.failover.complete 文件，下次再次切换的时候如果发现该目录下存在该文件将不允许触发切换，除非在第一次切换后收到删除该文件，为了方便，这里设置为--ignore_last_failover。

Server05(192.168.200.106)查看 Server05 监控是否正常：

```
[root@monitor ~]# masterha_check_status --conf=/etc/masterha/app1.cnf
app1 (pid:35895) is running(0:PING_OK), master:server01
```

可以看见已经在监控了

Server05(192.168.200.106)查看启动日志

```
[root@server05 ~]# cat /var/log/masterha/app1/manager.log
Checking the Status of the script.. OK
Wed Apr 29 12:57:26 2020 - [info]  OK.
Wed Apr 29 12:57:26 2020 - [warning] shutdown_script is not defined.
Wed Apr 29 12:57:26 2020 - [info] Set master ping interval 1 seconds.
Wed Apr 29 12:57:26 2020 - [warning] secondary_check_script is not defined. It is highly
recommended setting
it to check master reachability from two or more routes.Wed Apr 29 12:57:26 2020 - [info]
Starting ping health check on server01(192.168.200.107:3306)..
Wed Apr 29 12:57:26 2020 - [info] Ping(SELECT) succeeded, waiting until MySQL doesn't
respond..
```

注意：其中"Ping(SELECT) succeeded, waiting until MySQL doesn't respond.."说明整个系统已经开始监控了。

关闭 MHA manager 监控，忽略操作

```
masterha_stop --conf=/etc/masterha/app1.cnf
```

发现已经将 VIP：192.168.200.100 绑定在网卡 ens32

```
[root@server01 ~]# ip a | grep ens3
2: ens32: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group
default qlen 1000
    inet 192.168.200.107/24 brd 192.168.200.255 scope global noprefixroute ens32
    inet 192.168.200.100/24 brd 192.168.200.255 scope global secondary ens32:1
```

Primary Master(192.168.200.107) 模拟主库故障

```
[root@server01 ~]# systemctl stop mariadb
[root@server01 ~]# netstat -lnpt | grep :3306
[root@server01 ~]# ip a | grep ens32
2: ens32: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group
default qlen 1000
    inet 192.168.200.107/24 brd 192.168.200.255 scope global noprefixroute ens32
```

slave1(192.168.200.109)状态：

```
MariaDB [(none)]> show slave status\G
MariaDB [(none)]> show slave status\G
*************************** 1. row ***************************
               Slave_IO_State: Waiting for master to send event
                  Master_Host: 192.168.200.108
                  Master_User: repl
                  Master_Port: 3306
                Connect_Retry: 60
              Master_Log_File: master-bin.000001
          Read_Master_Log_Pos: 1368
               Relay_Log_File: slave-relay-bin.000002
                Relay_Log_Pos: 530
        Relay_Master_Log_File: master-bin.000001
             Slave_IO_Running: Yes
            Slave_SQL_Running: Yes
```

slave2(192.168.200.104)状态：

```
MariaDB [(none)]> show slave status\G
MariaDB [(none)]>  show slave status\G
*************************** 1. row ***************************
               Slave_IO_State: Waiting for master to send event
                  Master_Host: 192.168.200.108
                  Master_User: repl
                  Master_Port: 3306
                Connect_Retry: 60
              Master_Log_File: master-bin.000001
          Read_Master_Log_Pos: 1368
               Relay_Log_File: slave-relay-bin.000002
                Relay_Log_Pos: 530
        Relay_Master_Log_File: master-bin.000001
             Slave_IO_Running: Yes
            Slave_SQL_Running: Yes
```

Server05(192.168.200.106) 监控已经自动关闭：

```
[root@server05 ~]#
[1]+  完成                  nohup masterha_manager --conf=/etc/masterha/app1.cnf --
remove_dead_master_conf --ignore_last_failover < /dev/null >
```

```
/var/log/masterha/app1/manager.log 2>&1
```

Server05(192.168.200.106) 查看监控配置文件已经发生了变化（server01 的配置已被删除）：

```
[root@server05 ~]# cat /etc/masterha/app1.cnf
[server default]
manager_log=/var/log/masterha/app1/manager.log
manager_workdir=/var/log/masterha/app1
master_binlog_dir=/var/lib/mysql
master_ip_failover_script=/usr/local/bin/master_ip_failover
password=123456
ping_interval=1
remote_workdir=/tmp
repl_password=123456
repl_user=repl
user=root

[server2]
candidate_master=1
check_repl_delay=0
hostname=server02
port=3306

[server3]
hostname=server03
port=3306

[server4]
hostname=server04
port=3306
```

Server05(192.168.200.106) 故障切换过程中的日志文件内容如下

```
[root@server05 ~]# tail -f /var/log/masterha/app1/manager.log
Selected server02 as a new master.
server02: OK: Applying all logs succeeded.
server02: OK: Activated master IP address.
server03: This host has the latest relay log events.
server04: This host has the latest relay log events.
Generating relay diff files from the latest slave succeeded.
server03: OK: Applying all logs succeeded. Slave started, replicating from server02.
```

```
server04: OK: Applying all logs succeeded. Slave started, replicating from server02.
server02: Resetting slave info succeeded.
Master failover to server02(192.168.200.108:3306) completed successfully.
```

故障 主库修复及 及 VIP 切回

```
Primary Master(192.168.200.107)
[root@server01 ~]# systemctl start mariadb
[root@server01 ~]# netstat -lnpt | grep :3306
```

Primary Master(192.168.200.107) 指向新的主库

```
[root@server01 ~]# mysql -u root -p123456
stop slave;
CHANGE MASTER TO
MASTER_HOST='192.168.200.108',
MASTER_USER='repl',
MASTER_PASSWORD='123456';
start slave;
show slave status\G
*************************** 1. row ***************************
               Slave_IO_State: Waiting for master to send event
                  Master_Host: 192.168.200.108
                  Master_User: repl
                  Master_Port: 3306
                Connect_Retry: 60
              Master_Log_File: master-bin.000001
          Read_Master_Log_Pos: 1368
               Relay_Log_File: mariadb-relay-bin.000002
                Relay_Log_Pos: 1206
        Relay_Master_Log_File: master-bin.000001
             Slave_IO_Running: Yes
            Slave_SQL_Running: Yes
```

Server05(192.168.200.106) 修改监控配置文件添加 server1 配置:

```
[root@server05 ~]# vim /etc/masterha/app1.cnf
[server01]
hostname=server01
port=3306
```

Server05(192.168.200.106) 检查集群状态:

```
[root@server05 ~]# masterha_check_repl --conf=/etc/masterha/app1.cnf
Wed Apr 29 13:21:20 2020 - [info] Alive Servers:
Wed Apr 29 13:21:20 2020 - [info]   server01(192.168.200.107:3306)
Wed Apr 29 13:21:20 2020 - [info]   server02(192.168.200.108:3306)
Wed Apr 29 13:21:20 2020 - [info]   server03(192.168.200.109:3306)
Wed Apr 29 13:21:20 2020 - [info]   server04(192.168.200.104:3306)
server02 (current master)
 +--server01
 +--server03
 +--server04
MySQL Replication Health is OK.
```

Server05(192.168.200.106) 开启监控

```
[root@server05 ~]# nohup masterha_manager --conf=/etc/masterha/app1.cnf--
remove_dead_master_conf --ignore_last_failover< /dev/null
>/var/log/masterha/app1/manager.log 2>&1 &
[1] 38442
```

第二次模拟故障：

Secondary Master(192.168.200.108) 关闭现有主库 mysql

```
[root@server02 ~]# ip a | grep ens32
2: ens32: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group
default qlen 1000
    inet 192.168.200.108/24 brd 192.168.200.255 scope global noprefixroute ens32
    inet 192.168.200.100/24 brd 192.168.200.255 scope global secondary ens32:1
[root@server02 ~]# systemctl stop mariadb
[root@server02 ~]# netstat -lnpt | grep :3306
```

Primary Master(192.168.200.107)

查看不到VIP时；重启网络服务是个好办法

systemctl restart network

或者手动清除中继日志

```
[root@server01 ~]# ip a | grep ens32
2: ens32: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group
default qlen 1000
    inet 192.168.200.107/24 brd 192.168.200.255 scope global noprefixroute ens32
    inet 192.168.200.100/24 brd 192.168.200.255 scope global secondary ens32:1
```

slave1(192.168.200.109)状态:

```
MariaDB [(none)]>  show slave status\G
*************************** 1. row ***************************
               Slave_IO_State: Waiting for master to send event
                  Master_Host: 192.168.200.107
                  Master_User: repl
                  Master_Port: 3306
                Connect_Retry: 60
              Master_Log_File: master-bin.000005
          Read_Master_Log_Pos: 245
               Relay_Log_File: slave-relay-bin.000002
                Relay_Log_Pos: 530
        Relay_Master_Log_File: master-bin.000005
             Slave_IO_Running: Yes
            Slave_SQL_Running: Yes
```

slave2(192.168.200.104)状态:

```
MariaDB [(none)]>  show slave status\G
*************************** 1. row ***************************
               Slave_IO_State: Waiting for master to send event
                  Master_Host: 192.168.200.107
                  Master_User: repl
                  Master_Port: 3306
                Connect_Retry: 60
              Master_Log_File: master-bin.000005
          Read_Master_Log_Pos: 245
               Relay_Log_File: slave-relay-bin.000002
                Relay_Log_Pos: 530
        Relay_Master_Log_File: master-bin.000005
             Slave_IO_Running: Yes
            Slave_SQL_Running: Yes
```

Server05(192.168.200.106) 配置文件变化（已经移除故障机 server2 配置）：

```
[root@server05 ~]# cat /etc/masterha/app1.cnf
[server default]
manager_log=/var/log/masterha/app1/manager.log
manager_workdir=/var/log/masterha/app1
master_binlog_dir=/var/lib/mysql
```

```
master_ip_failover_script=/usr/local/bin/master_ip_failover
password=123456
ping_interval=1
remote_workdir=/tmp
repl_password=123456
repl_user=repl
user=root
[server1]
hostname=server01
port=3306
[server3]
hostname=server03
port=3306
[server4]
hostname=server04
port=3306
```

Server05(192.168.200.106) 监控日志：

```
Selected server01 as a new master.
server01: OK: Applying all logs succeeded.
server01: OK: Activated master IP address.
server03: This host has the latest relay log events.
server04: This host has the latest relay log events.
Generating relay diff files from the latest slave succeeded.
server03: OK: Applying all logs succeeded. Slave started, replicating from server01.
server04: OK: Applying all logs succeeded. Slave started, replicating from server01.
server01: Resetting slave info succeeded.
Master failover to server01(192.168.200.107:3306) completed successfully.
```

修复 Secondary Master(192.168.200.108)主机

```
[root@server02 ~]# systemctl start mariadb
[root@server02 ~]# netstat -lnpt | grep :3306
```

Secondary Master(192.168.200.108) 指向新的主库

```
[root@server02 ~]# mysql -u root -p123456
stop slave;
CHANGE MASTER TO
MASTER_HOST='192.168.200.107',
MASTER_USER='repl',
MASTER_PASSWORD='123456';
start slave;
show slave status\G
MariaDB [(none)]> show slave status\G
*************************** 1. row ***************************
               Slave_IO_State: Waiting for master to send event
                  Master_Host: 192.168.200.107
                  Master_User: repl
                  Master_Port: 3306
                Connect_Retry: 60
              Master_Log_File: master-bin.000005
          Read_Master_Log_Pos: 245
               Relay_Log_File: mariadb-relay-bin.000006
                Relay_Log_Pos: 530
        Relay_Master_Log_File: master-bin.000005
             Slave_IO_Running: Yes
            Slave_SQL_Running: Yes
```

Server05(192.168.200.106)修改监控配置文件添加 server2 配置：

```
[root@server05 ~]# vim /etc/masterha/app1.cnf
[server2]
hostname=server02
candidate_master=1
port=3306
check_repl_delay=0
```

Server05(192.168.200.106)检查集群状态：

```
[root@server05 ~]# masterha_check_repl --conf=/etc/masterha/app1.cnf
server01 (current master)
+--server02
+--server03
+--server04
```

十、 配置读请求负载均衡

1.安装 LVS 和 Keepalived

192.168.200.110 和 192.168.200.105 主机：

```
yum -y install ipvsadm kernel-devel openssl-devel keepalived
```

2. master 主机设置

1.192.168.200.110　lvs master 主机配置:

```
[root@master ~]# vim /etc/keepalived/keepalived.conf
! Configuration File for keepalived
global_defs {
notification_email {
crushlinux@163.com
}
notification_email_from crushlinux@163.com
smtp_server 192.168.200.1
smtp_connect_timeout 30
router_id LVS_DEVEL
vrrp_skip_check_adv_addr
vrrp_strict
vrrp_garp_interval 0
vrrp_gna_interval 0
}
vrrp_instance VI_1 {
state MASTER
interface ens32
virtual_router_id 51
priority 100
advert_int 1
authentication {
auth_type PASS
auth_pass 1111
}
virtual_ipaddress {
192.168.200.200
}
}
virtual_server 192.168.200.200 3306 {
delay_loop 6
lb_algo wrr
lb_kind DR
protocol TCP
real_server 192.168.200.113 3306 {
weight 1
TCP_CHECK {
connect_timeout 10
nb_get_retry 3
delay_before_retry 3
connect_port 3306
}
}
real_server 192.168.200.114 3306 {
weight 1
TCP_CHECK {
connect_timeout 10
nb_get_retry 3
delay_before_retry 3
connect_port 3306
}
}
}

[root@master ~]# systemctl restart keepalived
```

2.检查 VIP 地址

```
[root@master ~]# ip add | grep ens32
2: ens32: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group
default qlen 1000
    inet 192.168.200.110/24 brd 192.168.200.255 scope global noprefixroute ens32
    inet 192.168.200.200/32 scope global ens32
```

3.查看负载策略

```
[root@master ~]#  ipvsadm -Ln
IP Virtual Server version 1.2.1 (size=4096)
Prot LocalAddress:Port Scheduler Flags
  -> RemoteAddress:Port           Forward Weight ActiveConn InActConn
TCP  192.168.200.200:3306 rr
  -> 192.168.200.104:3306         Route   1      0          0
  -> 192.168.200.109:3306         Route   1      0          0
TCP  10.10.10.2:1358 rr persistent 50
  -> 192.168.200.200:1358         Masq    1      0          0
TCP  10.10.10.3:1358 rr persistent 50
```

3.backup 主机设置

192.168.200.105 backup 主机配置

```
[root@master ~]# scp /etc/keepalived/keepalived.conf 192.168.200.105:/etc/keepalived/
[root@backup ~]# vim /etc/keepalived/keepalived.conf
! Configuration File for keepalived
global_defs {
notification_email {
crushlinux@163.com
}
notification_email_from crushlinux@163.com
smtp_server 192.168.200.1
smtp_connect_timeout 30
router_id LVS_DEVEL
vrrp_skip_check_adv_addr
vrrp_strict
vrrp_garp_interval 0
vrrp_gna_interval 0
}
vrrp_instance VI_1 {
state BACKUP
interface ens32
virtual_router_id 51
priority 99
advert_int 1
authentication {
auth_type PASS
auth_pass 1111
}
virtual_ipaddress {
192.168.200.200
}
}
virtual_server 192.168.200.200 3306 {
delay_loop 6
lb_algo wrr
lb_kind DR
protocol TCP
real_server 192.168.200.113 3306 {
weight 1
TCP_CHECK {
connect_timeout 10
nb_get_retry 3
delay_before_retry 3
connect_port 3306
}
}
real_server 192.168.200.114 3306 {
weight 1
TCP_CHECK {
connect_timeout 10
nb_get_retry 3
delay_before_retry 3
connect_port 3306
}
}
}
[root@backup ~]# systemctl restart keepalived
```

4.为数据库节点设置VIP 地址及配置ARP相关参数

1.slave1(192.168.200.109)和 slave2(192.168.200.104)主机配置 realserver 脚本:

slave1(192.168.200.109)

```
[root@server03 ~]# vim realserver.sh
#!/bin/bash
SNS_VIP=192.168.200.200
ifconfig lo:0 $SNS_VIP netmask 255.255.255.255 broadcast $SNS_VIP
/sbin/route add -host $SNS_VIP dev lo:0
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
sysctl -p >/dev/null 2>&1
```

2.realserver1 检测VIP 地址

```
[root@server03 ~]# bash realserver.sh
[root@server03 ~]# ip a | grep lo
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen
1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
    inet 192.168.200.200/32 brd 192.168.200.200 scope global lo:0
    inet 192.168.200.109/24 brd 192.168.200.255 scope global noprefixroute ens32
    inet 192.168.122.1/24 brd 192.168.122.255 scope global virbr0
```

3.远程复制到 realserver2 主机上

```
[root@server03 ~]# scp realserver.sh 192.168.200.114:/root/
```

4.realserver2 检测 VIP 地址

```
[root@server04 ~]# bash realserver.sh
[root@server04 ~]# ip a |grep lo
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen
1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
    inet 192.168.200.200/32 brd 192.168.200.200 scope global lo:0
    inet 192.168.200.104/24 brd 192.168.200.255 scope global noprefixroute ens32
    inet 192.168.122.1/24 brd 192.168.122.255 scope global virbr0
```

5.数据库读请求测试

```
iptables -F
setenforce 0
systemctl stop firewalld
```

1.Server05(192.168.200.115)担任 client 角色测试，连接 vip

```
[root@server05 ~]# mysql -uroot -p123456 -h192.168.200.200 -P3306
```

2.断开后在连接测试。。。。（多访问两遍）

```
[root@server05 ~]# mysql -uroot -p123456 -h192.168.200.200 -P3306
```

3.查看请求分流

192.168.200.110

```
[root@master ~]#  ipvsadm -Ln
IP Virtual Server version 1.2.1 (size=4096)
Prot LocalAddress:Port Scheduler Flags
  -> RemoteAddress:Port           Forward  Weight    ActiveConn           InActConn
TCP  192.168.200.200:3306 rr
  -> 192.168.200.104:3306         Route      1                           0
2
  -> 192.168.200.109:3306         Route       1                          0
1
[root@master ~]#  ipvsadm -Lnc
IPVS connection entries
```

```
pro expire state       source                virtual               destination
TCP 01:55  FIN_WAIT    192.168.200.106:56084 192.168.200.200:3306 192.168.200.104:3306
TCP 01:49  FIN_WAIT    192.168.200.106:56082 192.168.200.200:3306 192.168.200.109:3306
```

6.数据库连接故障处理

```
[root@localhost ~]# mysql -u root -p123456 -h192.168.200.113 -P3306
ERROR 1129 (HY000): Host '192.168.200.113' is blocked because of many connection
errors;unblock with 'mysqladmin flush-hosts'
[root@server03 ~]# mysqladmin -uroot -p123456 flush-hosts
```