MySQL之MHA【推荐高可用方案】

一、MHA概念

软件下载 https://github.com/yoshinorim

MHA（Master High Availability）目前在MySQL高可用方面是一个相对成熟的解决方案，它由日本DeNA公司youshimaton（现就职于Facebook公司）开发，是一套优秀的作为MySQL高可用性环境下故障切换和主从提升的高可用软件。

在MySQL故障切换过程中，MHA能做到在0~30秒之内自动完成数据库的故障切换操作，并且在进行故障切换的过程中，MHA能在最大程度上保证数据的一致性，以达到真正意义上的高可用。

MHA里有两个角色一个是MHA Node（数据节点）另一个是MHA Manager（管理节点）。

- **MHA Manager**可以单独部署在一台独立的机器上管理多个master-slave集群，也可以部署在一台slave节点上。
- **MHA Node**运行在每台MySQL服务器上，MHA Manager会定时探测集群中的master节点，当master出现故障时，它可以自动将最新数据的slave提升为新的master，然后将所有其他的slave重新指向新的master。

整个故障转移过程对应用程序完全透明

MHA可以与半同步复制结合起来。如果只有一个slave已经收到了最新的二进制日志，MHA可以将最新的二进制日志应用于其他所有的slave服务器上，因此可以保证所有节点的数据一致性。

```
1 注：从MySQL5.5开始，MySQL以插件的形式支持半同步复制。
```

二、同步，异步，半同步

异步复制（Asynchronous replication）

MySQL默认的复制即是异步的，主库在执行完客户端提交的事务后会立即将结果返给给客户端，并不关心从库是否已经接收并处理，这样就会有一个问题，主如果crash掉了，此时主上已经提交的事务可能并没有传到从上，如果此时，强行将从提升为主，可能导致新主上的数据不完整。

全同步复制（Fully synchronous replication）

指当主库执行完一个事务，所有的从库都执行了该事务才返回给客户端。因为需要等待所有从库执行完该事务才能返回，所以全同步复制的性能必然会收到严重的影响。

半同步复制（Semisynchronous replication）

介于异步复制和全同步复制之间，主库在执行完客户端提交的事务后不是立刻返回给客户端，而是等待至少一个从库接收到并写到relay log中才返回给客户端。相对于异步复制，半同步复制提高了数据的安全性，同时它也造成了一定程度的延迟，这个延迟最少是一个TCP/IP往返的时间

三、工作原理

相较于其它HA软件，MHA的目的在于维持MySQL Replication中Master库的高可用性，其最大特点是**可以修复多个Slave之间的差异日志，最终使所有Slave保持数据一致**，然后从中选择一个充当新的Master，并将其它Slave指向它。

- -从宕机崩溃的master保存二进制日志事件(binlogevents)。
- -识别含有最新更新的slave。
- -应用差异的中继日志(relay log)到其它slave。
- -应用从master保存的二进制日志事件(binlogevents)。
- -提升一个slave为新master。 -使其它的slave连接新的master进行复制。

四、架构拓扑

目前MHA主要支持一主多从的架构，要搭建MHA,要求一个复制集群中必须最少有三台数据库服务器，一主二从，即一台充当master，一台充当备用master，另外一台充当从库，因为至少需要三台服务器。

拓扑：

IP	**角色**
192.168.10.11	**master1（主）**
192.168.10.12	**master2（备主）**
192.168.10.13	**slave & manager（从和管理节点）** **manager可以部署在一台slave节点上**

部署MHA

一、基础环境准备

1、关闭防火墙

在配置好IP地址后检查selinux，iptables设置，关闭 selinux ，iptables 服务以便后期主从同步不出错（注：时间要同步）

2、 在四台机器都配置epel源

```
curl -o /etc/yum.repos.d/CentOS-Base.repo http://mirrors.aliyun.com/repo/Centos-7.repo
curl -o /etc/yum.repos.d/epel.repo http://mirrors.aliyun.com/repo/epel-7.repo
```

3、建立ssh无交互登录环境

```
#### master1主机
ssh-keygen -t rsa
for i in 11 12 13
do ssh-copy-id -i ~/.ssh/id_rsa.pub root@192.168.10.$i
done
```

```
#### master2主机
ssh-keygen -t rsa
for i in 11 12 13
do ssh-copy-id -i ~/.ssh/id_rsa.pub root@192.168.10.$i
done
```

```
#### slave主机
ssh-keygen -t rsa
for i in 11 12 13
do ssh-copy-id -i ~/.ssh/id_rsa.pub root@192.168.10.$i
done
```

测试ssh无交互登录

```
for i in 11 12 13
do ssh root@192.168.10.$i hostname
done
```

4、配置hosts环境

```
# vim /etc/hosts
```

```
127.0.0.1 localhost localhost.localdomain localhost4 localhost4.localdomain4
::1 localhost localhost.localdomain localhost6 localhost6.localdomain6
192.168.10.11  master1
192.168.10.12  master2
192.168.10.13  slave
```

二、配置mysql半同步复制

为了尽可能的减少主库硬件损坏宕机造成的数据丢失，因此在配置MHA的同时建议配置成MySQL的半同步复制。

注：mysql半同步插件是由谷歌提供，具体位置/usr/local/mysql/lib/plugin/下，一个是master用的semisync_master.so，一个是slave用的semisync_slave.so，下面我们就来具体配置一下。

如果不清楚Plugin的目录，用如下查找：

```
mysql> show variables like '%plugin_dir%';
+---------------+-----------------------------+
| Variable_name | Value |
+---------------+-----------------------------+
| plugin_dir | /usr/local/mysql/lib/plugin/ |
+---------------+-----------------------------+
```

1、分别在主从节点上安装相关的插件（ master, Candicate master,slave ）

在MySQL上安装插件需要数据库支持动态载入。检查是否支持，用如下检测：

```
mysql> show variables like '%have_dynamic%';
+----------------------+-------+
| Variable_name | Value |
+----------------------+-------+
| have_dynamic_loading | YES |
+----------------------+-------+
```

所有mysql数据库服务器，安装半同步插件(semisync_master.so,semisync_slave.so)

```
mysql> INSTALL PLUGIN rpl_semi_sync_master SONAME 'semisync_master.so';
mysql> INSTALL PLUGIN rpl_semi_sync_slave SONAME 'semisync_slave.so';
```

其他mysql主机采用同样的方法安装 检查Plugin是否已正确安装：

```
mysql> show plugins;
```

查看半同步相关信息

```
mysql> show variables like '%rpl_semi_sync%';
+-------------------------------------------+------------+
| Variable_name | Value |
+-------------------------------------------+------------+
| rpl_semi_sync_master_enabled | OFF |
```

```
| rpl_semi_sync_master_timeout | 10000 |
| rpl_semi_sync_master_trace_level | 32 |
| rpl_semi_sync_master_wait_for_slave_count | 1 |
| rpl_semi_sync_master_wait_no_slave | ON |
| rpl_semi_sync_master_wait_point | AFTER_SYNC |
| rpl_semi_sync_slave_enabled | OFF |
| rpl_semi_sync_slave_trace_level | 32 |
+-------------------------------------------+-----------+
# 上图可以看到半同复制插件已经安装，只是还没有启用，所以是off
```

2、修改my.cnf文件，配置主从同步：

注：若主MYSQL服务器已经存在，只是后期才搭建从MYSQL服务器，在置配数据同步前应先将主MYSQL服务器的要同步的数据库拷贝到从MYSQL服务器上（如先在主MYSQL上备份数据库，再用备份在从MYSQL服务器上恢复）

master1 主机：

```
server-id = 1
log-bin=mysql-bin
binlog_format=mixed
log-bin-index=mysql-bin.index
rpl_semi_sync_master_enabled=1
rpl_semi_sync_master_timeout=1000
rpl_semi_sync_slave_enabled=1
relay_log_purge=0
relay-log = relay-bin
relay-log-index = slave-relay-bin.index

# 参数注释
# rpl_semi_sync_master_enabled=1 1表是启用，0表示关闭
# rpl_semi_sync_master_timeout=10000：毫秒单位 ，该参数主服务器等待确认消息10秒后，不再等待，变为异步方式。
```

master2主机：

```
server-id = 2
log-bin=mysql-bin
binlog_format=mixed
log-bin-index=mysql-bin.index
relay_log_purge=0
relay-log = relay-bin
relay-log-index = slave-relay-bin.index
rpl_semi_sync_master_enabled=1
rpl_semi_sync_master_timeout=10000
rpl_semi_sync_slave_enabled=1
```

```
# 参数注释
# relay_log_purge=0
# 禁止 SQL 线程在执行完一个 relay log 后自动将其删除
# 对于MHA场景下，对于某些滞后从库的恢复依赖于其他从库的relay log，因此采取禁用自动删除功能
```

Slave主机：

```
server-id = 3
log-bin = mysql-bin
relay-log = relay-bin
relay-log-index = slave-relay-bin.index
read_only = 1
rpl_semi_sync_slave_enabled=1
```

查看半同步相关信息

```
mysql> show variables like '%rpl_semi_sync%';
+-------------------------------------------+------------+
| Variable_name | Value |
+-------------------------------------------+------------+
| rpl_semi_sync_master_enabled | ON |
| rpl_semi_sync_master_timeout | 1000 |
| rpl_semi_sync_master_trace_level | 32 |
| rpl_semi_sync_master_wait_for_slave_count | 1 |
| rpl_semi_sync_master_wait_no_slave | ON |
| rpl_semi_sync_master_wait_point | AFTER_SYNC |
| rpl_semi_sync_slave_enabled | ON |
| rpl_semi_sync_slave_trace_level | 32 |
+-------------------------------------------+-----------+
```

查看半同步状态：

```
mysql> show status like '%rpl_semi_sync%';
+--------------------------------------------+-------+
| Variable_name | Value |
+--------------------------------------------+-------+
| Rpl_semi_sync_master_clients | 0 |
| Rpl_semi_sync_master_net_avg_wait_time | 0 |
| Rpl_semi_sync_master_net_wait_time | 0 |
| Rpl_semi_sync_master_net_waits | 0 |
| Rpl_semi_sync_master_no_times | 0 |
| Rpl_semi_sync_master_no_tx | 0 |
| Rpl_semi_sync_master_status | ON |
```

```
| Rpl_semi_sync_master_timefunc_failures | 0 |
| Rpl_semi_sync_master_tx_avg_wait_time | 0 |
| Rpl_semi_sync_master_tx_wait_time | 0 |
| Rpl_semi_sync_master_tx_waits | 0 |
| Rpl_semi_sync_master_wait_pos_backtraverse| 0 |
| Rpl_semi_sync_master_wait_sessions | 0 |
| Rpl_semi_sync_master_yes_tx | 0 |
| Rpl_semi_sync_slave_status | OFF |
+--------------------------------------------+------+
```

有几个状态参数值得关注的：

- rpl_semi_sync_master_status ：显示主服务是异步复制模式还是半同步复制模式
- rpl_semi_sync_master_clients ：显示有多少个从服务器配置为半同步复制模式
- rpl_semi_sync_master_yes_tx ：显示从服务器确认成功提交的数量
- rpl_semi_sync_master_no_tx ：显示从服务器确认不成功提交的数量
- rpl_semi_sync_master_tx_avg_wait_time ：事务因开启 semi_sync ，平均需要额外等待的时间
- rpl_semi_sync_master_net_avg_wait_time ：事务进入等待队列后，到网络平均等待时间

master1主机：

```
grant replication slave on *.* to mharep@'192.168.10.%' identified by '123';
grant all privileges on *.* to manager@'192.168.10.%' identified by '123';
show master status;
# 第一条grant命令是创建一个用于主从复制的帐号，在master和candicate master的主机上创建即可。
# 第二条grant命令是创建MHA管理账号，所有mysql服务器上都需要执行。
# MHA会在配置文件里要求能远程登录到数据库，所以要进行必要的赋权
```

master2主机：

```
grant replication slave on *.* to mharep@'192.168.10.%' identified by '123';
grant all privileges on *.* to manager@'192.168.10.%' identified by '123';

change master to
master_host='192.168.10.11',
master_port=3306,
```

```
master_user='mharep',
master_password='123',
master_log_file='mysql-bin.000001',
 master_log_pos=744;

 start slave;
 show slave status\G
```

Slave主机：

```
 grant all privileges on *.* to manager@'192.168.10.%' identified by
'123';

 change master to
 master_host='192.168.43.10',
 master_port=3306,
 master_user='mharep',
 master_password='123',
 master_log_file='mysql-bin.000001',
 master_log_pos=744;

 start slave;
 show slave status\G
```

查看master1服务器的半同步状态：

```
 mysql> show status like '%rpl_semi_sync%';
```

三、配置mysql-mha

mha包括manager节点和data节点，data节点包括原有的MySQL复制结构中的主机，至少3台，即1主2从，当master failover后，还能保证主从结构；

- master1，master2，slavev：只需安装node包。
- manager需要安装node包和manager包。

1、 在所有主机上安装mha所依赖的软件包（需要系统自带的yum源并联网）

```
 yum -y install perl-DBD-MySQL perl-Config-Tiny perl-Log-Dispatch perl-Par
allel-ForkManager perl-Config-IniFiles ncftp perl-Params-Validate perl-CPAN
perl-Test-Mock-LWP.noarch perl-LWP-Authen-Negotiate.noarch perl-devel perl-
ExtUtils-CBuilder perl-ExtUtils-MakeMaker
```

2、 以下操作管理节点需要两个都安装， 在3台数据库节点只要安装MHA的node节点：

在所有数据库节点上安装mha4mysql-node-0.56.tar.gz

```
tar zxf mha4mysql-node-0.58.tar.gz
cd mha4mysql-node-0.58
perl Makefile.PL
make && make install
```

其他两个数据节点也安装mha4mysql-node-0.56.tar.gz（过程略）

在管理节点需要两个都安装：mha4mysqlnode-0.56.tar.gz和mha4mysql-manager-0.56.tar.gz

```
tar zxf mha4mysql-node-0.58.tar.gz
cd mha4mysql-node-0.58
perl Makefile.PL
make && make install

tar zxf mha4mysql-manager-0.58.tar.gz
cd mha4mysql-manager-0.58/
perl Makefile.PL
make && make install
```

根据提示输入：

```
mkdir /etc/masterha
mkdir -p /masterha/app1
mkdir /scripts
cp samples/conf/* /etc/masterha/
cp samples/scripts/* /scripts/
```

3、 配置mha

与绝大多数Linux应用程序类似，MHA的正确使用依赖于合理的配置文件。MHA的配置文件与mysql的my.cnf文件配置相似，采取的是param=value的方式来配置，配置文件位于管理节点，通常包括每一个mysql server的主机名，mysql用户名，密码，工作目录等等。

编辑/etc/masterha/app1.conf，内容如下：

```
vim /etc/masterha/app1.cnf
```

```
[server default]
manager_workdir=/masterha/app1
manager_log=/masterha/app1/manager.log
user=manager
password=123
```

```
ssh_user=root
repl_user=mharep
repl_password=123
ping_interval=1


[server1]
hostname=192.168.10.11
port=3306
master_binlog_dir=/usr/local/mysql/data
candidate_master=1

[server2]
hostname=192.168.10.12
port=3306
master_binlog_dir=/usr/local/mysql/data
candidate_master=1

[server3]
hostname=192.168.10.13
port=3306
master_binlog_dir=/usr/local/mysql/data
no_master=1
```

```
#### 清空masterha_default.cnf
>/etc/masterha/masterha_default.cnf
```

配关配置项的解释：

- manager_workdir=/masterha/app1 //设置manager的工作目录
- manager_log=/masterha/app1/manager.log //设置manager的日志
- user=manager //设置监控用户manager
- password=123456 //监控用户manager的密码
- ssh_user=root //ssh连接用户 repl_user=mharep //主从复制用户
- repl_password=123.abc //主从复制用户密码 ping_interval=1 //设置监控主库，发送ping包的时间间隔，默认是3秒，尝试三次没有回应的时候自动进行
- railover master_binlog_dir=/usr/local/mysql/data //设置master 保存binlog的位置，以便MHA可以找到master的日志，我这里的也就是mysql的数据目录
- candidate_master=1 //设置为候选master，如果设置该参数以后，发生主从切换以后将会将此从库提升为主库。

SSH 有效性验证：

```
masterha_check_ssh --global_conf=/etc/masterha/masterha_default.cnf --conf=/etc/masterha/app1.cnf
```

集群复制的有效性验证： mysql必须都启动

```
ln -s /usr/local/mysql/bin/* /usr/local/bin/ # 四台主机都要做链接
masterha_check_repl --global_conf=/etc/masterha/masterha_default.cnf --conf=/etc/masterha/app1.cnf
```

启动 manager：

```
nohup masterha_manager --conf=/etc/masterha/app1.cnf &>/tmp/mha_manager.log &
```

状态检查：

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

故障转移验证：

先停掉 master1，候选master2库（Slave）会自动failover为Master. MHA自动停止

重启master1，设为master2的从

```
grep 'CHANGE' /masterha/app1/manager.log
```

删除manager的 masterha/app1/app1.failover.complete文件

```
rm -f /masterha/app1/app1.failover.complete
```

启动MHA

```
nohup masterha_manager --conf=/etc/masterha/app1.cnf &>/tmp/mha_manager.log &
```

检查状态：

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

检查日志：

```
tail -f /masterha/app1/manager.log
```

四、配置VIP：

vip配置可以采用两种方式，一种通过keepalived的方式管理虚拟ip的浮动；另外一种通过脚本方式启动虚拟ip的方式（即不需要keepalived或者heartbeat类似的软件）

1、keepalived方式管理虚拟ip【推荐使用下面的通过脚本实现VIP切换】

keepalived配置方法如下： 下载软件进行并进行安装（两台master，准确的说一台是master，另外一台是备选master，在没有切换以前是slave）

1）编译keepalived

```
wget https://www.keepalived.org/software/keepalived-2.1.2.tar.gz
yum install kernel-devel openssl-devel popt-devel
tar zxf keepalived-2.1.2.tar.gz
cd keepalived-2.1.2/
 ./configure --prefix=/ && make && make install
systemctl enable keepalived.service
```

2）关闭防火墙或创建规则

```
firewall-cmd --direct --permanent --add-rule ipv4 filter OUTPUT 0 --in-interface ens33 --destination 224.0.0.18 --protocol vrrp -j ACCEPT
firewall-cmd --direct --permanent --add-rule ipv4 filter INPUT 0 --in-interface ens33 --destination 224.0.0.18 --protocol vrrp -j ACCEPT
firewall-cmd --reload
```

3）修改Keepalived的配置文件（在master上配置）

```
! Configuration File for keepalived
global_defs {
router_id mysql-ha1
}
vrrp_instance VI_1 {
state BACKUP
interface ens33
virtual_router_id 51
priority 100
 nopreempt # 非抢占模式
 advert_int 1
 authentication {
  auth_type PASS
  auth_pass 1111
 }
 virtual_ipaddress {
 192.168.10.100
 }
```

```
}
```

4）在候选master上配置

```
! Configuration File for keepalived
global_defs {
router_id mysql-ha2
}
vrrp_instance VI_1 {
state BACKUP
interface ens33
virtual_router_id 51
priority 50 # 修改优先级
 nopreempt # 非抢占模式
 advert_int 1
 authentication {
  auth_type PASS
 auth_pass 1111
 }
 virtual_ipaddress {
 192.168.10.100
 }
 }
 ## 非抢占模式（nopreempt）；backup->backup
```

5）启动keepalived服务，在master上启动并查看日志

```
systemctl start keepalived.service
ps -ef |grep keep
ip a |grep 100
```

2、MHA引入keepalived（MySQL服务进程挂掉时通过MHA 停止keepalived）

要想把keepalived服务引入MHA，我们只需要修改切换时触发的脚本文件master_ip_failover即可，在该脚本中添加在master发生宕机时对keepalived的处理。

1）编辑脚本/scripts/master_ip_failover，修改后如下（创建新文件然后覆盖原文件，添加x权限）

```
vim /scripts/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 = '192.168.10.100';
my $ssh_start_vip = "systemctl start keepalived.service";
my $ssh_stop_vip = "systemctl stop keepalived.service";
GetOptions(
'command=s' => \$command,
'ssh_user=s' => \$ssh_user,
'orig_master_host=s' => \$orig_master_host,
'orig_master_ip=s' => \$orig_master_ip,
'orig_master_port=i' => \$orig_master_port,
'new_master_host=s' => \$new_master_host,
'new_master_ip=s' => \$new_master_ip,
'new_master_port=i' => \$new_master_port,
);
exit &main();
sub main {
print "\n\nIN SCRIPT TEST====$ssh_stop_vip==$ssh_start_vip===\n\n";
if ( $command eq "stop" || $command eq "stopssh" ) {
my $exit_code = 1;
eval {
print "Disabling the VIP on old master: $orig_master_host \n";
&stop_vip();
$exit_code = 0;
};
if ($@) {
warn "Got Error: $@\n";
exit $exit_code;
}
exit $exit_code;
}
elsif ( $command eq "start" ) {
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\@cluster1 \" $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 \" $ssh_start_vip \"`;
}
# A simple system call that disable the VIP on the old_master
sub stop_vip() {
return 0 unless ($ssh_user);
`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";
}
```

2) 停止MHA

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

3) 在配置文件/etc/masterha/app1.cnf 中启用下面的参数(在[server default下面添加])

```
master_ip_failover_script=/scripts/master_ip_failover
# 当主库数据库发生故障时，会触发MHA切换
```

```
# MHA Manager会停掉主库上的keepalived服务，触发虚拟ip漂移到备选从库，从而完成切换
```

4）启动MHA

```
nohup masterha_manager --conf=/etc/masterha/app1.cnf &>/tmp/mha_manager.log &
masterha_check_status --conf=/etc/masterha/app1.cnf
masterha_check_repl --conf=/etc/masterha/app1.cnf
```

5）测试：

在master上停止mysqld服务 到slave(192.168.10.13)查看slave的状态：

查看VIP绑定

6）重构：

原主库修复成一个新的slave（grep查看mha的日志信息）

```
CHANGE MASTER TO
MASTER_HOST='192.168.10.12',
MASTER_PORT=3306,
MASTER_LOG_FILE='mysql-bin.000003',
MASTER_LOG_POS=154,
MASTER_USER='mharep',
MASTER_PASSWORD='123'
```

启动mha manager:

```
rm -fr /masterha/app1/app1.failover.complete
nohup masterha_manager --conf=/etc/masterha/app1.cnf --ignore_fail_on_start &>/tmp/mha_manager.log &
```

2、通过脚本实现VIP切换

通过脚本的方式管理VIP。这里是修改/scripts/master_ip_failover，也可以使用其他的语言完成，比如php语言。使用php脚本编写的failover这里就不介绍了。

1）手动在master服务器上绑定一个vip

```
ifconfig ens33:0 192.168.10.100/24
```

2）在mha-manager上修改/scripts/ master_ip_failover，内容如下

```
# 覆盖原脚本
vim /scripts/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 = '192.168.10.100';
my $key = '0';
my $ssh_start_vip = "/sbin/ifconfig ens33:$key $vip";
my $ssh_stop_vip = "/sbin/ifconfig ens33:$key down";
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" ) {
my $exit_code = 1;
eval {
print "Disabling the VIP on old master: $orig_master_host \n";
&stop_vip();
$exit_code = 0;
};
if ($@) {
warn "Got Error: $@\n";
exit $exit_code;
}
exit $exit_code;
}
elsif ( $command eq "start" ) {
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\@cluster1 \" $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 \" $ssh_start_vip \"`;
}
# A simple system call that disable the VIP on the old_master
sub stop_vip() {
return 0 unless ($ssh_user);
`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";
}
```

```
# 给脚本执行权限
chmod +x /scripts/master_ip_failover
```

3）在配置文件/etc/masterha/app1.cnf 中启用下面的参数(在[server default下面添加])

```
grep "master_ip_failover_script" /etc/masterha/app1.cnf
master_ip_failover_script=/scripts/master_ip_failover
```

4）启动MHA

```
nohup masterha_manager --conf=/etc/masterha/app1.cnf &>/tmp/mha_manager.log &
masterha_check_status --conf=/etc/masterha/app1.cnf
masterha_check_repl --conf=/etc/masterha/app1.cnf
```

5）测试：

在master上停止mysqld服务 到slave(192.168.10.13)查看slave的状态：

查看VIP绑定

```
mysql -uroot -p123 -h 192.168.10.100
```