MySQL 高可用：MHA 实现 MySQL 高可用

原创 屿辰Linux SRE运维派 2025年03月03日 18:03 北京

1 MySQL 集群

1.6 MySQL 高可用

1.6.1 MySQL 高可用解决方案

MySQL 官方和社区里推出了很多高可用的解决方案，不同方案的高可用率大体如下，仅供参考（数据引用自 Percona）

Method	Level of Availability
Simple replication	98% -- 99.9%
Master-Master/MMM	99%
SAN	99.5% -- 99.9%
DRBD，MHA，Tungsten Replicator	99.9%
NDBCluster，Galera Cluster	99.999%

MMM：Multi-Master Replication Manager for MySQL，Mysql 主主复制管理器是一套灵活的脚本程序，基于perl实现，用来对mysql replication 进行监控和故障迁移，并能管理 mysql Master-Master 复制的配置（同一时间只有一个节点是可写的）。

```
http://www.mysql-mmm.org
https://code.google.com/archive/p/mysql-master-master/downloads
```

MHA：Master High Availability，对主节点进行监控，可实现自动故障转移至其它从节点；通过提升某一从节点为新的主节点，基于主从复制实现，还需要客户端配合实现，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，出于机器成本的考虑，淘宝进行了改造，目前淘宝TMHA已经支持一主一从。

```
https://code.google.com/archive/p/mysql-master-ha/
https://github.com/yoshinorim/mha4mysql-manager/wiki/Downloads
https://github.com/yoshinorim/mha4mysql-manager/releases
https://github.com/yoshinorim/mha4mysql-node/releases/tag/v0.5
```

以下技术可以达到金融级的高可用性要求：

Galera Cluster：wsrep(MySQL extended with the Write Set Replication) 通过 wsrep 协议在全局实现复制；任何一节点都可读写，不需要主从复制，实现多主读写。

GR（Group Replication)： MySQL官方提供的组复制技术(MySQL 5.7.17引入的技术)，基于原生复制技术 Paxos 算法，实现了多主更新，复制组由多个 server 成员构成，组中的每个 server 可独立地执行事务，但所有读写事务只在冲突检测成功后才会提交。

这3个节点互相通信，当有事件发生，都会向其他节点传播该事件，然后协商，如果大多数节点都同意这次的事件，那么该事件将通过，否则该事件将失败或回滚。这些节点可以是单主模型的(singleprimary)，也可以是多主模型的(multi-primary)。单主模型只有一个主节点可以接受写操作，主节点故障时可以自动选举主节点。多主模型下，所有节点都可以接受写操作，所以没有 master-slave 的概念。

1.6.2 MHA Master High Availability

1.6.2.1 MHA 工作原理和架构

```
https://github.com/yoshinorim/mha4mysql-manager/wiki #官方文档
```

MHA 集群架构

一个 MHA-manager 节点可以管理多个 MySQL 集群

MHA 工作原理

- MHA利用 SELECT 1 As Value 指令判断 master 服务器的健康性，一旦 master 宕机，MHA 从宕机崩溃的 master 保存二进制日志事件（binlog events）
- 识别含有最新更新的 slave
- 应用差异的中继日志（relay log）到其他的 slave
- 应用从 master 保存的二进制日志事件（binlog events）到所有 slave 节点
- 提升一个 slave 为新的 master
- 使其他的 slave 连接新的 master 进行复制
- 故障服务器自动被剔除集群(masterha_conf_host)，将配置信息去掉
- 旧的 Master的 VIP 漂移到新的 master上，用户应用就可以访问新的 Master
- MHA 是一次性的高可用性解决方案，Manager 会自动退出

选举新的 Master

- 如果设定权重(candidate_master=1)，按照权重强制指定新主，但是默认情况下如果一个 slave 落后 master 二进制日志超过 100M 的relay logs，即使有权重，也会失效，如果设置 check_repl_delay=0，即使落后很多日志，也强制选择其为新主
- 如果从库数据之间有差异，最接近于 Master 的 slave 成为新主
- 如果所有从库数据都一致,按照配置文件顺序最前面的当新主

数据恢复

- 当主服务器的 SSH 还能连接，从库对比主库 position 或者 GTID 号，将二进制日志保存至各个从节点并且应用(执行save_binary_logs 实现)
- 当主服务器的 SSH 不能连接，对比从库之间的 relaylog 的差异（执行 apply_diff_relay_logs[实现])

注意：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

MHA 软件

MHA 软件由两部分组成，Manager工具包和 Node 工具包

Manager工具包主要包括以下几个工具

```
masterha_check_ssh #检查MHA的SSH配置状况
masterha_check_repl #检查MySQL复制状况
masterha_manger #启动MHA
masterha_check_status #检测当前MHA运行状态
masterha_master_monitor #检测master是否宕机
masterha_master_switch #故障转移（自动或手动）
masterha_conf_host #添加或删除配置的server信息
masterha_stop --conf=app1.cnf #停止MHA
masterha_secondary_check #两个或多个网络线路检查MySQL主服务器的可用
```

Node工具包：这些工具通常由MHA Manager的脚本触发，无需人为操作，主要包括以下几个工具

```
save_binary_logs #保存和复制master的二进制日志
apply_diff_relay_logs #识别差异的中继日志事件并将其差异的事件应用于其他的slave
filter_mysqlbinlog #去除不必要的ROLLBACK事件（MHA已不再使用此工具）
purge_relay_logs #清除中继日志（不会阻塞SQL线程）
```

MHA自定义扩展

```
secondary_check_script #通过多条网络路由检测master的可用性
master_ip_ailover_script #更新Application使用的masterip
shutdown_script #强制关闭master节点
report_script #发送报告
init_conf_load_script #加载初始配置参数
master_ip_online_change_script #更新master节点ip地址
```

MHA配置文件

```
global配置，为各application提供默认配置，默认文件路径 /etc/masterha_default.cn
application配置：为每个主从复制集群
```

1.6.2.2 MHA 实现 MySQL 高可用

主机清单

主机IP	主机名	角色	操作系统	MySQL版本

10.0.0.198	mha-manager	MHA Manager	Centos7	
10.0.0.177	master	master	Rocky8	8.0.30
10.0.0.183	slave-1	slave-1	Rocky8	8.0.30
10.0.0.186	slave-2	slave-2	Rocky8	8.0.30

软件下载

```
https://github.com/yoshinorim/mha4mysql-manager/releases/tag/v0.58 #man
https://github.com/yoshinorim/mha4mysql-node/releases/tag/v0.58 #node下
#mha4mysql-manager-0.58-0.el7.centos.noarch.rpm 只支持CentOS7上安装，不支持
#mha4mysql-manager-0.56-0.el6.noarch.rpm 不支持CentOS 8，只支持CentOS7及以
```

在 mha-manager 节点上安装 manager 包和 node 包

```
[root@mha-manager ~]# wget https://github.com/yoshinorim/mha4mysql-man
[root@mha-manager ~]# wget https://github.com/yoshinorim/mha4mysqlnode

[root@mha-manager ~]# ll mha4mysql-*
-rw-r--r--. 1 root root 81024 Dec 7 2021 mha4mysql-manager-0.58-0.el7.
-rw-r--r--. 1 root root 36328 Dec 7 2021 mha4mysql-node-0.58-0.el7.cen

#通过yum安装，自行解决依赖
[root@mha-manager ~]# yum install epel-release
[root@mha-manager ~]# yum install mha4mysql-* -y

#查看，可执行程序都是 perl 脚本
[root@mha-manager ~]# file /usr/bin/masterha_*
/usr/bin/masterha_check_repl: Perl script, ASCII text executable
/usr/bin/masterha_check_ssh: Perl script, ASCII text executable
/usr/bin/masterha_check_status: Perl script, ASCII text executable
/usr/bin/masterha_conf_host: Perl script, ASCII text executable
/usr/bin/masterha_manager: Perl script, ASCII text executable
/usr/bin/masterha_master_monitor: Perl script, ASCII text executable
/usr/bin/masterha_master_switch: Perl script, ASCII text executable
/usr/bin/masterha_secondary_check: Perl script, ASCII text executable
/usr/bin/masterha_stop: Perl script, ASCII text executable
```

在所有 mysql 节点上安装 node 包

```
[root@mha-manager ~]# scp mha4mysql-node-0.58-0.el7.centos.noarch.rpm
```

```
10.0.0.177:

[root@mha-manager ~]# scp mha4mysql-node-0.58-0.el7.centos.noarch.rpm
10.0.0.183:

[root@mha-manager ~]# scp mha4mysql-node-0.58-0.el7.centos.noarch.rpm
10.0.0.186:

#分别安装
[root@master ~]# yum install -y mha4mysql-node-0.58-0.el7.centos.noarch
```

在所有

```
#生成密钥对，并在当前主机完成C/S校验
[root@mha-manager ~]# ssh-keygen
[root@mha-manager ~]# ssh-copy-id 127.1

#分发
[root@mha-manager ~]# rsync -av .ssh 10.0.0.177:/root/
[root@mha-manager ~]# rsync -av .ssh 10.0.0.183:/root/
[root@mha-manager ~]# rsync -av .ssh 10.0.0.186:/root/

#测试ssh连接
......
```

在 mha-manager 节点创建相关配置文件

```
[root@mha-manager ~]# mkdir /etc/mastermha
[root@mha-manager ~]# vim /etc/mastermha/app1.cnf

#默认设置
[server default]
user=mhauser #mha-manager节点连接远程mysql使用的账户，需要有管理员的权限
password=123456 #mha-manager节点连接远程mysql使用的账户密码
manager_workdir=/data/mastermha/app1/ #mha-manager对于当前集群的工作目录
manager_log=/data/mastermha/app1/manager.log #mha-manager对于当前集群的日
remote_workdir=/data/mastermha/app1/ #mysql 节点mha工作目录，会自动创建
ssh_user=root #各节点间的SSH连接账号，提前做好基于key 的登录验证，用于访问二进制日
repl_user=repluser #mysql节点主从复制用户名
repl_password=123456 #mysql节点主从复制密码
ping_interval=1 #mha-manager节点对于master节点的心跳检测时间间隔
master_ip_failover_script=/usr/local/bin/master_ip_failover #切换VIP的p
report_script=/usr/local/bin/sendmail.sh #发送告警信息脚本
```

```
check_repl_delay=0 #默认值为1，表示如果 slave 中从库落后
主库 relay log 超过 100M，主库不会选择这个从库为新的 master，因为这个从库进行恢复
时间。通过设置参数 check_repl_delay=0，mha 触发主从切换时会忽略复制的延时，对于设
candidate_master=1 的从库非常有用，这样确保这个从库一定能成为最新的 master
master_binlog_dir=/data/mysql/logbin/ #指定二进制日志存放的目录，mha4mysql-

[server1]
hostname=10.0.0.177
candidate_master=1 #优先候选master，即使不是集群中事件最新的slave，也会优先当ma

[server2]
hostname=10.0.0.183
candidate_master=1 #优先候选master，即使不是集群中事件最新的slave，也会优先当ma

[server3]
hostname=10.0.0.186
```

提升 slave 节点为 master 节点的策略

- 如果所有 slave 节点日志都是一致的，则默认会以配置文件的顺序选择一个 slave 节点提升为 master 节点。
- 如果 slave 节点上的日志不一致，则会选择数据量最接近 master 节点的 slave 节点，将其提升为 master 节点。
- 如果对某 slave 节点设定了权重（candidate_master=1），权重节点会优先选择。但是此节点日志量落后于 master 节点超过100M，也不会被选择。可以配合 check_repl_delay=0，关闭日志量的检查，强制选择候选节点。

配置相关脚本

```
#告警消息脚本
[root@mha-manager ~]# vim /usr/local/bin/sendmail.sh
#!/bin/bash
echo 'MHA is failover!' | mail -s 'MHA Warning' alex@josedu.com

[root@mha-manager ~]# chmod a+x /usr/local/bin/sendmail.sh

#需要安装邮件服务
[root@mha-manager ~]# yum install mailx postfix

#邮件服务配置
[root@mha-manager ~]# vim /etc/mail.rc
#加在最下面
set from=170xxxx325@qq.com #发件箱
set smtp=smtp.qq.com #smtp服务器
```

```
set smtp-auth-user=170xxxx325@qq.com #发件人
set smtp-auth-password=geahkxxxxvregjac #授权码
set smtp-auth=login
set ssl-verifv=ignore

[root@mha-manager ~]# systemctl restart postfix.service

#测试告警邮件
[root@mha-manager ~]# sendmail.sh
```

```
[root@mha-manager ~]# vim /usr/local/bin/master_ip_failover
#!/usr/bin/env perl
# Copyright (C) 2011 DeNA Co.,Ltd.
#
# This program is free software; you can redistribute it and/or modify
# it under the terms of the GNU General Public License as published by
# the Free Software Foundation; either version 2 of the License, or
# (at your option) any later version.
#
# This program is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
# GNU General Public License for more details.
#
# You should have received a copy of the GNU General Public License
# along with this program; if not, write to the Free Software
# Foundation, Inc.,
# 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA
## Note: This is a sample script and is not complete. Modify the scrip

use strict;
use warnings FATAL => 'all';

use Getopt::Long;
use MHA::DBHelper;

my (
  $command, $ssh_user, $orig_master_host,
  $orig_master_ip, $orig_master_port, $new_master_host,
  $new_master_ip, $new_master_port, $new_master_user,
  $new_master_password
);

my $vip = '10.0.0.100/24'; #virtually IP，此IP会在不同的MySQL节点漂移
my $key = "1";
```

```
my $ssh_start_vip = "/sbin/ifconfig ens160:$key $vip"; #在网卡上添加IP，
my $ssh_stop_vip = "/sbin/ifconfig ens160:$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,
  'new_master_user=s' => \$new_master_user,
  'new_master_password=s' => \$new_master_password,
);

exit &main();

sub main {
  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 {
      # 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,
      my $exit_code = 10;
      eval {
        print "Enabling the VIP - $vip on the new master - $new_master
        &start_vip();
        &stop_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_host \" $ssh_start_vip \"`;
        exit 0;
      }
      else {
        &usage();
        exit 1;
      }
}

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@mha-manager ~]# chmod a+x /usr/local/bin/master_ip_failover
```

在 master 节点配置 VIP，此IP会在不同 MySQL 节点上漂移

```
[root@master ~]# ifconfig ens160:1 10.0.0.100/24
[root@master ~]# ifconfig ens160:1
ens160:1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
        inet 10.0.0.100 netmask 255.255.255.0 broadcast 10.0.0.255
        ether 00:0c:29:7e:ce:82 txqueuelen 1000 (Ethernet)
```

配置 MySQL 节点

```
#master 节点配置
[root@master ~]# yum install -y mysql-server
[root@master ~]# vim /etc/my.cnf
......
[mysqld]
server-id=177
log-bin=/data/mysql/logbin/mysql-bin

[root@master ~]# mkdir -pv /data/mysql/logbin/
[root@master ~]# chown -R mysql.mysql /data/mysql
[root@master ~]# systemctl start mysqld.service
```

```
#slave-1 节点配置
[root@slave-1 ~]# yum install -y mysql-server
[root@slave-1 ~]# vim /etc/my.cnf
......
[mysqld]
server-id=183
log-bin=/data/mysql/logbin/mysql-bin
read-only

[root@slave-1 ~]# mkdir -pv /data/mysql/logbin/
[root@slave-1 ~]# chown -R mysql.mysql /data/mysql
[root@slave-1 ~]# systemctl start mysqld.service
```

```
#slave-2 节点配置
[root@slave-2 ~]# yum install -y mysql-server
[root@slave-2 ~]# vim /etc/my.cnf
......
[mysqld]
server-id=186
log-bin=/data/mysql/logbin/mysql-bin
read-only

[root@slave-2 ~]# mkdir -pv /data/mysql/logbin/
[root@slave-2 ~]# chown -R mysql.mysql /data/mysql
[root@slave-2 ~]# systemctl start mysqld.service
```

```
#配置主从
#master 节点查看
mysql> show master logs;
+------------------+-----------+-----------+
```

```
| Log_name         | File_size | Encrypted |
+------------------+-----------+-----------+
| mysql-bin.000001 | 180       | No        |
| mysql-bin.000002 | 157       | No        |
+------------------+-----------+-----------+
2 rows in set (0.00 sec)

#创建主从同步账号并授权
mysql> create user repluser@'10.0.0.%' identified by '123456';
Query OK, 0 rows affected (0.00 sec)

mysql> grant replication slave on *.* to repluser@'10.0.0.%';
Query OK, 0 rows affected (0.00 sec)

#创建 mha-manager 使用的账号并授权
mysql> create user mhauser@'10.0.0.%' identified by '123456';
Query OK, 0 rows affected (0.00 sec)

mysql> grant all on *.* to mhauser@'10.0.0.%';
Query OK, 0 rows affected (0.00 sec)
#slave-1 节点配置主从

mysql> CHANGE MASTER TO
    -> MASTER_HOST='10.0.0.177',
    -> MASTER_USER='repluser',
    -> MASTER_PASSWORD='123456',
    -> MASTER_LOG_FILE='mysql-bin.000002',
    -> MASTER_LOG_POS=157;
Query OK, 0 rows affected, 8 warnings (0.01 sec)

mysql> start slave;
Query OK, 0 rows affected, 1 warning (0.01 sec)

mysql> show slave status\G

#slave-2 节点上配置主从，同上
```

```
#主从同步测试
#master 节点上导入数据并查看
[root@master ~]# mysql < hellodb_innodb.sql

mysql> show databases like '%hello%';
+--------------------+
| Database (%hello%) |
+--------------------+
```

```
| hellodb            |
+--------------------+
1 row in set (0.00 sec)

mysql> show tables from hellodb;
+------------------+
| Tables_in_hellodb |
+------------------+
| classes          |
| coc              |
| courses          |
| scores           |
| students         |
| teachers         |
| toc              |
+------------------+
7 rows in set (0.00 sec)

#slave-1 节点查看
[root@slave-1 ~]# mysql -e "show tables from hellodb"
+------------------+
| Tables_in_hellodb |
+------------------+
| classes          |
| coc              |
| courses          |
| scores           |
| students         |
| teachers         |
| toc              |
+------------------+

#slave-2 节点查看
[root@slave-2 ~]# mysql -e "show tables from hellodb"
+------------------+
| Tables_in_hellodb |
+------------------+
| classes          |
| coc              |
| courses          |
| scores           |
| students         |
| teachers         |
| toc              |
+------------------+
```

mha-manager 节点上检查环境

```
#配置和 SSH 连接检查
[root@mha-manager ~]# vim /etc/mastermha/app1.cnf

#主从复制检查，会在mysql节点自动创建 remote_workdir=/data/mastermha/app1/
[root@mha-manager ~]# masterha_check_repl --conf=/etc/mastermha/app1.cn

#查看当前mysql 集群状态
[root@mha-manager ~]# masterha_check_status --conf=/etc/mastermha/app1.
app1 is stopped(2:NOT_RUNNING).
```

在 mha-manager 节点上启动集群

```
#生产环境放在后台执行，并且与终端分离
nohup masterha_manager --conf=/etc/mastermha/app1.cnf --remove_dead_mas

#如果想停止后台的 manager，使用此命令
masterha_stop --conf=/etc/mastermha/app1.cnf

#启动
[root@mha-manager ~]# masterha_manager --conf=/etc/mastermha/app1.cnf -

#查看生成的文件
[root@mha-manager ~]# tree /data/mastermha/app1/
/data/mastermha/app1/
├── app1.master_status.health
└── manager.log
0 directories, 2 files

#查看日志
[root@mha-manager ~]# cat /data/mastermha/app1/manager.log
```

在 MySQL 的 master 节点上查看 mha-manager 发送的心跳查询

```
#开启 master 节点通用日志
mysql> set global general_log=1;
Query OK, 0 rows affected (0.00 sec)

#每秒检测一次活动状态，间隔时长是在配置文件中指定的
[root@master ~]# tail -f /var/lib/mysql/master.log
```

测试，停止 MySQL master 节点，VIP会转移

```
#master 节点上查看，当前网卡有VIP
[root@master ~]# ip a s ens160
2: ens160: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state U
    link/ether 00:0c:29:7e:ce:82 brd ff:ff:ff:ff:ff:ff
    inet 10.0.0.177/24 brd 10.0.0.255 scope global dynamic noprefixrou
        valid_lft 1192sec preferred_lft 1192sec
    inet 10.0.0.100/24 brd 10.0.0.255 scope global secondary ens160:1
        valid_lft forever preferred_lft forever
    inet6 fe80::20c:29ff:fe7e:ce82/64 scope link noprefixroute
        valid_lft forever preferred_lft forever

#停止 mysqld 服务
[root@master ~]# systemctl stop mysqld

#再次查看网卡，VIP已经转移了
[root@master ~]# ip a s ens160
2: ens160: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state U
    link/ether 00:0c:29:7e:ce:82 brd ff:ff:ff:ff:ff:ff
    inet 10.0.0.177/24 brd 10.0.0.255 scope global dynamic noprefixrou
        valid_lft 1099sec preferred_lft 1099sec
    inet6 fe80::20c:29ff:fe7e:ce82/64 scope link noprefixroute
        valid_lft forever preferred_lft forever

#查看 mha-manager 日志，提示提升了新的 master 节点，发送告警日志
[root@mha-manager ~]# cat /data/mastermha/app1/manager.log

#在slave-1节点查看，VIP已转移到当前主机，因为在 mha-manager 配置中设定了当前节点仍
[root@slave-1 ~]# ip a s ens160
2: ens160: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state U
    link/ether 00:0c:29:b1:3f:86 brd ff:ff:ff:ff:ff:ff
    inet 10.0.0.183/24 brd 10.0.0.255 scope global dynamic noprefixrou
        valid_lft 933sec preferred_lft 933sec
    inet 10.0.0.100/24 brd 10.0.0.255 scope global secondary ens160:1
        valid_lft forever preferred_lft forever
    inet6 fe80::20c:29ff:feb1:3f86/64 scope link noprefixroute
        valid_lft forever preferred_lft forever

#当前同步状态消失，只读状态失效
mysql> show slave status\G
Empty set, 1 warning (0.00 sec)

mysql> select @@read_only;
+-------------+
| @@read_only |
```

```
+-------------+
| 0           |
+-------------+
1 row in set (0.00 sec)

#插入数据
mysql> use hellodb;
Database changed

mysql> insert into teachers(name,age,gender)values('tom',20,'M');
Query OK, 1 row affected (0.00 sec)

#在 slave-2 节点上查看，master IP已经发生了变化
mysql> show slave status\G
*************************** 1. row ***************************
Slave_IO_State: Waiting for source to send event
Master_Host: 10.0.0.183
Master_User: repluser
Master_Port: 3306

#查看数据，验证是否能从新的 master 节点同步数据
mysql> use hellodb;
Database changed

mysql> select * from hellodb.teachers where name='tom';
+-----+------+-----+--------+
| TID | Name | Age | Gender |
+-----+------+-----+--------+
| 5   | tom  | 20  | M      |
+-----+------+-----+--------+
1 row in set (0.00 sec)

#此时收到了告警邮件
```

说明：

- 当 master 节点出当机，mha-manager 节点上的 masterha_manager 程序会退出
- 经过手动处理，原来的 master 节点上线后，应该将该节点设置成 slave 节点，让其从新的 master 节点处同步数据
- 对于前端用户来讲，此过程是无感知的，因为前端用户是通过连接VIP来操作数据库的，而VIP一直可用，只是转移到另一台机器而已
- MHA 只能解决一次 master 节点故障，VIP 只能漂移一次， 再次启动之前需要删除相关文件，否则无法工作

```
[root@mha-manager ~]# cat /etc/mastermha/app1.cnf
```

```
....
manager_workdir=/data/mastermha/app1/
manager_log=/data/mastermha/app1/manager.log
remote_workdir=/data/mastermha/app1/

#MHA 再次使用之前，需要先删除 manager_workdir 指向的目录和 remote_workdir 指向的
#manager_workdir 指向的目录在 mha-manager 节点上
#remote_workdir 指向的目录在 mysql 节点上
```

— END —

–点击下方卡片关注–

数据库 29　MySQL 29　高可用 2

数据库 · 目录

上一篇
MySQL 集群：ProxySQL 实现 MySQL 读写分离

下一篇
MySQL 高可用：Galera Cluster