MySQL学习笔记（Day045：mysqlrouter_MHA）

MySQL学习

MySQL学习笔记（Day045：mysqlrouter_MHA）
一. MySQL Router
1.1. MySQL Router的安装
1.2. MySQL Router的配置
1.3. MySQL Router的部署
1.4. MySQL Router的启动
1.5. MySQL Router的测试
二. MHA
2.1. MHA介绍
2.2. MHA Failover过程
2.3. MHA的安装
2.3.1. MHA Manager的安装
2.3.2. MHA Node的安装
2.4. SSH免密码登录
2.4.1. MHA Manager生成Key
2.4.2. 公钥分发
2.4.3. 修改hosts
2.4.4. 确认SSH打通
2.5. MHA环境说明
2.6. MySQL建立复制
2.7. MHA配置
2.7.1. MHA配置文件和说明
2.7.2. MHA脚本说明
2.7.3. Slave上的relay_log配置
2.7.3.1. pure_relay_logs的参数
2.7.3.2. 定时清理relay-log的脚本
2.7.4. SSH检测
2.7.5. 复制关系检测
2.8. 启动MHA
2.8.1. 添加VIP
2.8.2. 启动masterha_manager
2.9. 自动Failover测试
2.9.1. 停止Slave的IO线程
2.9.2. Master上产生大量binlog
2.9.3. 开启Slave的IO线程
2.9.4. 关闭Master的MySQL
2.9.5. 观察MHA Manager的日志
2.9.6. 查看最终复制关系
2.9.7. 查看masterha_manager的配置
2.10. 相关注意事项
2.10.1. Failover后MHA Manager自动退出
2.11. 手动Failover测试
2.11.1. 注意事项
2.11.2. 切换测试
2.11.3. 查看状态
2.12. MHA总结
2.12.1. 部署操作过程
2.12.2. Old Master恢复
2.12.2.1. 日志处理
2.12.2.2. 角色处理

一. MySQL Router

> 官方下载

1.1. MySQL Router的安装

```
[root@MyServer local]> ll | grep mysql-router
-rw-rw-r--.  1 root  root  6343846 Apr 10 14:38 mysql-router-2.0.3-linux-glibc2.12-x86-64bit.tar.gz

# 解压缩
[root@MyServer local]> tar zxvf mysql-router-2.0.3-linux-glibc2.12-x86-64bit.tar.gz

# 做软连接
[root@MyServer local]> ln -sf mysql-router-2.0.3-linux-glibc2.12-x86-64bit mysql-router

[root@MyServer local]> cd mysql-router

[root@MyServer mysql-router]> ll
total 20
drwxr-xr-x. 2 7161 wheel 4096 Feb 23 02:29 bin  # 二进制可执行文件
drwxr-xr-x. 4 7161 wheel 4096 Feb 23 02:29 include  # 头文件
drwxr-xr-x. 3 7161 wheel 4096 Feb 23 02:29 lib      # lib库
drwxrwxr-x. 2 7161 wheel 4096 Feb 23 02:29 run
drwxr-xr-x. 3 7161 wheel 4096 Feb 23 02:29 share  # 配置示例

# 创建日志目录
[root@MyServer mysql-router]> mkdir -p /var/log/mysqlrouter/

# 创建配置文件目录
[root@MyServer mysql-router]> mkdir -p /etc/mysqlrouter
[root@MyServer mysql-router]> touch /etc/mysqlrouter/mysqlrouter.ini
```

1.2. MySQL Router的配置

配置文件为 `/etc/mysqlrouter/mysqlrouter.ini`

```
[DEFAULT]
# 定义日志目录
logging_folder = /var/log/mysqlrouter

[logger]
# 定义日志等级
level = INFO

# 一个高可用的标签
[routing:failover]
bind_address = 0.0.0.0
bind_port = 7001
max_connections = 1024
# 目前就支持两种 ：  read-write  和 read-only
# read-write：用于高可用，用于可读可写
# read-only：用于负载均衡，只读
mode = read-write
# 实际转发的地址
# 第一个socket如果可用，就一直会使用第一个
# 如果第一个socket无法连接了，才会连接到第二个socket
destinations = 172.18.14.70:3306, 172.18.14.71:3306

# 一个用于复杂均衡的标签
[routing:balancing]
bind_address = 0.0.0.0
bind_port = 7002
max_connections = 1024
# 用于负载均衡的只读模式
mode = read-only
# 这里的两个socket是轮询用的
destinations =  172.18.14.72:3306, 172.18.14.73:3306
```

1.3. MySQL Router的部署

```
          +--------Master-------+  RW:Active
      +-----+ M:172.18.14.70:3306 <-------------+    +--------+
      |     +---------------------+             |    | MySQL  |
      |                                         |    | Router |
      | rpl +--------Slave1-------+  RW:Standby |    +--------+        +---------+
      +-----> S:172.18.14.71:3306 <-------------+----+   RW   <---RW---+         |
      |     +---------------------+             |    |  7001  |        |  Some   |
      |                                         |    +--------+        | Clients |
      | rpl +--------Slave2-------+  RO:Active    |   RO   <---RO---+         |
      +-----> S:172.18.14.72:3306 <-------------+----+  7002  |        +---------+
      |     +---------------------+             |    +--------+
      |                                         |   172.18.14.68
      | rpl +--------Slave3-------+  RO:Active  |
      +-----> S:172.18.14.73:3306 <-------------+
            +---------------------+
```

1. `Master、Slave1、Slave2、Slave3` 之间的复制关系是 `独立` 于 `MySQL Router` 的；
   - 也就是说，MySQL Router `不管MySQL服务器的主从关系` 以及数据是否一致，他只起到 `数据转发` 作用
2. `Slave1` 是候选节点，在 `Master能提供服务` 的时候，MySQL Router `不会` 把连接 `转发给Slave1` ；
3. 当Master宕机，MySQL Router把 `mode = read-write` 的连接转发给 `候选节点Slave1` 时，并不保证Slave1上的数据是回放完毕的（*因为他普不了主从复制*）；

1.4. MySQL Router的启动

官方二进制安装包中的启动脚本 `share/doc/mysqlrouter/sample_mysqlrouter.init` 是debian系的；
如果要在CentOS上运行，可在MySQL官网下载RedHat对应的mysql-router的RPM包，解压后，将其中的/etc/init.d/mysqlrouter 抽取出来，上传到CentOS中，然后修改部分变量的路径即可，本文档中的启动脚本修改如下：

```
#! /bin/bash
#
# mysqlrouter   This shell script takes care of starting and stopping
#       the MySQL Router
#
# chkconfig: 2345 66 34
# description: MySQL Router
# processname: mysqlrouter
# config: /etc/mysqlrouter/mysqlrouter.ini
# pidfile: /var/run/mysqlrouter/mysqlrouter.pid
#
# Copyright (c) 2015, Oracle and/or its affiliates. All rights reserved.
#
# 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; version 2 of the License.
#
# 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 St, Fifth Floor, Boston, MA  02110-1301  USA

#
# Maintainer: MySQL Release Engineering <mysql-build@oss.oracle.com>
#

# Source function library
. /etc/rc.d/init.d/functions

# Source networking configuration
. /etc/sysconfig/network

# add general install path
base_dir=/usr/local/mysql-router
# fix exec path
exec=${base_dir}/bin/mysqlrouter
prog=mysqlrouter
piddir=${base_dir}/run/mysqlrouter
pidfile=${piddir}/mysqlrouter.pid
logdir=/var/log/mysqlrouter
logfile=$logdir/mysqlrouter.log
lockfile=/var/lock/subsys/$prog

# add conf path
conf=/etc/mysqlrouter/mysqlrouter.ini

start () {
    [ -d $piddir ] || mkdir -p $piddir
    chown mysql:mysql $piddir
    [ -d $logdir ] || mkdir -p $logdir
    chown mysql:mysql $logdir
    [ -e $logfile ] || touch $logfile
    chown mysql:mysql $logfile
    export ROUTER_PID=$pidfile
    # add opt -c to resolv mysqlrouter.ini
    daemon --user mysql $exec -c $conf >/dev/null 2>&1 &
    ret=$?
    if [ $ret -eq "0" ]; then
    action $"Starting $prog: " /bin/true
    touch /var/lock/subsys/$prog
    else
    action $"Starting $prog: " /bin/false
    fi
    return $ret
}

stop () {
    [ -f /var/lock/subsys/$prog ] || return 0
    killproc mysqlrouter >/dev/null 2>&1
    ret=$?
    if [ $ret -eq "0" ]; then
    rm -f $pidfile
    rm -f /var/lock/subsys/$prog
    action $"Stopping $prog: " /bin/true
    else
    ation $"Stopping $prog: " /bin/false
    fi
}

restart ()  {
    stop
    start
}

condrestart () {
    [ -e /var/lock/subsys/$prog ] && restart || return 0
}

case "$1" in
    start)
    start
    ;;
    stop)
    stop
    ;;
    status)
    status -p "$pidfile" $prog
    ;;
    restart)
    restart
    ;;
    condrestart|try-restart)
    condrestart
    ;;
    reload)
    exit 3
    ;;
    force-reload)
    restart
    ;;
    *)
    echo $"Usage: $0 {start|stop|status|condrestart|try-restart|reload|force-reload}"
    exit 2
esac

exit $?
```

启动修改后的 mysqlrouter 的启动脚本

```
[root@MyServer ~]> /etc/init.d/mysqlrouter  start
Starting mysqlrouter:                                      [  OK  ]

[root@MyServer init.d]> netstat -tunlp | grep mysql
tcp        0      0 0.0.0.0:7001                0.0.0.0:*                   LISTEN      19674/mysqlrouter
tcp        0      0 0.0.0.0:7002                0.0.0.0:*                   LISTEN      19674/mysqlrouter

[root@MyServer init.d]> cat /var/log/mysqlrouter/mysqlrouter.log
2016-04-10 17:00:30 INFO    [7f385ba38700] [routing:failover] listening on 0.0.0.0:7001; read-write
2016-04-10 17:00:30 INFO    [7f385c439700] [routing:balancing] listening on 0.0.0.0:7002; read-only
```

1.5. MySQL Router的测试

```
[root@MyServer ~]> mysql -u root -p123 -h 172.18.14.68 -P 7002 -e "show variables like 'server_id'";
+---------------+-------+
| Variable_name | Value |
+---------------+-------+
| server_id     | 102   |
+---------------+-------+

[root@MyServer ~]> mysql -u root -p123 -h 172.18.14.68 -P 7002 -e "show variables like 'server_id'";
+---------------+-------+
| Variable_name | Value |
+---------------+-------+
| server_id     | 103   |
+---------------+-------+

[root@MyServer ~]> mysql -u root -p123 -h 172.18.14.68 -P 7002 -e "show variables like 'hostname'";
+---------------+--------+
| Variable_name | Value  |
+---------------+--------+
| hostname      | Slave2 |
+---------------+--------+

[root@MyServer ~]> mysql -u root -p123 -h 172.18.14.68 -P 7002 -e "show variables like 'hostname'";
+---------------+--------+
| Variable_name | Value  |
+---------------+--------+
| hostname      | Slave3 |
+---------------+--------+
```

以上的测试方式都是 短连接 ，每次执行完SQL语句后，便会释放连接，每一次查询都走 轮询 进行 负载均衡

如果使用 长连接 ，则会一只 保持 一个MySQL的连接，如下所示：

```
[root@MyServer ~]> mysql -u root -p123 -h 172.18.14.68 -P 7002
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 35
Server version: 5.7.11-log MySQL Community Server (GPL)

Copyright (c) 2000, 2013, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> select @@hostname;
+------------+
| @@hostname |
+------------+
| Slave2     |
+------------+
1 row in set (0.00 sec)

mysql> select @@hostname;
+------------+
| @@hostname |
+------------+
| Slave2     |
+------------+
1 row in set (0.00 sec)

mysql> select @@hostname;
+------------+
| @@hostname |
+------------+
| Slave2     |
+------------+
1 row in set (0.00 sec)
```

长连接的情况下，会 保持 一个连接不变，直到客户端超时或者主动释放。

> 通过这种方式，其实就实现了读写分离，但是需要依靠业务端进行代码修改才能完成。读的指向7002（RO），写的指向7001（RW）；
> 如果一个事物中既有读，又有写的时候，针对MySQL Router这个中间件来说，应该指向7001（RW）
>
> 针对其他中间件如mycat，DDB（网易）、TDDL（淘宝）来说，对外 只暴露一个端口 ， 对业务层是透明 的，但是实现起来较重，因为这类中间件需要对SQL语句进行解析（*CPU开销较重*）。且在一个 事物中有读有写 的情况下，最保险的做法还是应该发送给Master节点，避免数据不一致。

二. MHA

> 官方网站

2.1. MHA介绍

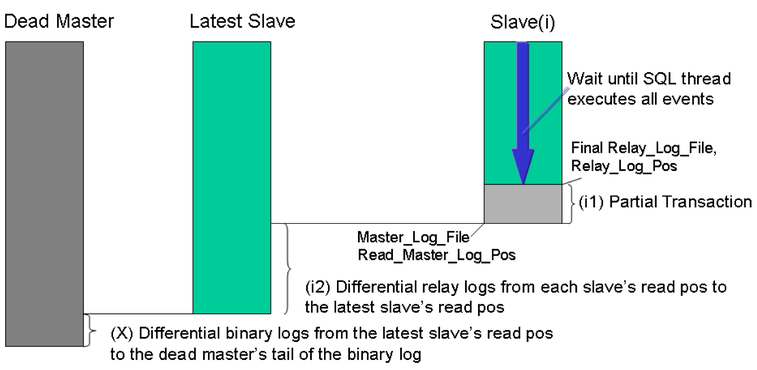
MHA是一套MySQL高可用 管理软件 ，除了检测Master宕机后，提升候选Slave为New Master之外（*漂虚拟IP*），还会 自动 让其他Slave与New Master 建立复制关系。

MHA Manager可以 单独部署 在一台独立的机器上，并管理 多个 master-slave集群

```
+-------------------------------------+ (app1.conf) +----------+
|                        VIP          |  heartbeat  |   MHA    |
|  +----------+   rpl    +--------+  <-------------+  Manager |
|  | Slave(M) +----------> Master |  | rpl manager +----------+
|  +----------+          +-^----^-+  |                  |
|                rpl      |    |    |                  |
|       +------------------+    |rpl |                  |
|       |                       |    |     (app2.conf) |
|  +----------+          +-------+-+  |             +----V----+
|  | Slave 1 |  ......  | Slave N |  |             |  Other  |
|  +----------+          +---------+  |             |  MySQL  |
|                                     |             |   rpl   |
|          MySQL Replication          |             +---------+
+-------------------------------------+
```

1. 上图中存在以下节点：
   - **MHA Manager**：MHA的管理节点，负责检测MySQL的状态，以及管理MySQL的复制关系；
   - **Master**：MySQL对外提供的服务（ *通过 VIP* ）的主节点；
   - **Slave(M)**：MySQL候选主节点，本质上为一个Slave节点，只是在Master宕机后，会被提升为 New Master ；
   - **Slave N**：MySQL从机节点，对外可以提供只读服务；
2. 当Master宕机后，MHA Manager会让 Slave(M) 提升为 New Master ，然后修改 Slave N 的复制关系（ *CHANGE MASTER* ），指向 New Master ；
   - MHA Manager检测Master是否宕机， 不会简单的只检查自身与Master之间的直连状态；
   - MHA Manager会 透过其他Slave节点去检测Master，进行二次探测 ，最大限度的避免脑裂的发生；
     - 比如说，检测到Master宕机了，MHA Manager会 透过Slave(M) 检测是否为宕机；
     - 如果检测后仍为宕机状态，会继续 透过Slave1...SlaveN 进行探测，只有在 透过所有节点 检测到Master宕机了，才真的认为Master宕机了；
     - **以上操作步骤的前提是，MHA Manager到所有MySQL节点的 SSH免密码登录要打通 ；**
3. 最后将提供对外服务的 VIP漂移到New Master 上，继续对外提供服务；
4. 一个MHA Manager可以 管理多个 MySQL的Master-Slave集群（*通过不同的app.conf*）

2.2. MHA Failover过程



- On slave(i),
  - Wait until the SQL thread executes events
  - Apply i1 -> i2 -> X
    - On the latest slave, i2 is empty

1. **从宕机崩溃的Master节点** 保存二进制日志事件 **（binlog events）；**
   - 此种情况为 MySQL挂了，但是服务器没挂 ，还是可以通过SSH连接过去；
   - 如果服务器彻底宕机了，该步骤略过；
2. **识别** 含有最新的更新 **的Slave节点；**
3. 应用差异的中继日志 **（relay-log）到其他的Slave；**
4. 应用 **从Master保存的** 二进制日志 **事件（binlog events）；**
   - 如果之前Master彻底宕机了，就没有保存的binlog，该步骤略过；
5. 提升一个 **Slave** 为新的 **Master（New）；**
6. **使其他的** Slave 连接新的 **Master（New）** **进行复制；**

> 从上面的步骤看， **MHA**无法完全保证数据不会丢 ；
>
> 1. MySQL 5.7 之前 数据不丢的前提是Master服务器还可以被MHA Manager进行SSH连接 ，通过 应用保存的binlog 的方式来保证。
> 2. MySQL 5.7 之后通过无损复制， 仅仅是减少了丢数据的可能性 ，假如此时的状态为 切成异步的状态 ，那就和之前一样了（可以设置超时的时间很大）；
> 3. 当Master恢复的时候，最后一部分数据是否需要Flashback，MHA也是不负责这个事情，需要人工介入。

2.3. MHA的安装

MHA由两个部分组成 MHA Manager 和 MHA Node

> MHA GoogleCode下载地址
> MHA-0.57 Google Drive下载地址

2.3.1. MHA Manager的安装

MHA Manager 的安装包（*当前最新版本为 mha4mysql-manager-0.57.tar.gz*）仅仅只需要在MHA Manager服务器上安装，过程如下所示：

```
# CentOS 7.0 MHA Manager
# 预先安装 Perl 依赖
Shell> yum install epel-release -y
Shell> yum install perl-DBI perl-DBD-MySQL perl-Config-Tiny perl-Log-Dispatch  perl-Parallel-ForkManager perl-Time-HiRes perl-Params-Validate perl-DateTime -y

Shell> tar zxvf mha4mysql-manager-0.57.tar.gz

Shell> cd mha4mysql-manager-0.57

Shell> perl Makefile.PL
*** Module::AutoInstall version 1.06
*** Checking for Perl dependencies...
[Core Features]
- DBI                   ...loaded. (1.627)  # 这些loaded的包都是通过之前yum预先安装好的
- DBD::mysql            ...loaded. (4.023)
- Time::HiRes           ...loaded. (1.9725)
- Config::Tiny          ...loaded. (2.14)
- Log::Dispatch         ...loaded. (2.41)
- Parallel::ForkManager ...loaded. (1.05)
- MHA::NodeConst        ...missing.         # 这个MHA::NodeConst缺失，下面会自动解决该依赖
==> Auto-install the 1 mandatory module(s) from CPAN? [y] y
*** Dependencies will be installed the next time you type 'make'.
*** Module::AutoInstall configuration finished.
Warning: prerequisite MHA::NodeConst 0 not found.
Writing Makefile for mha4mysql::manager
Writing MYMETA.yml and MYMETA.json

Shell> make
# ------省略部分输出------
*** Installing dependencies...  ## 这个一步可能会要比较长的时间
*** Installing MHA::NodeConst...
*** Could not find a version 0 or above for MHA::NodeConst; skipping.  # 这个包其实没有了，skip了，不用管
*** Module::AutoInstall installation finished.
# ------省略部分输出------

Shell> make install  # 安装到 /usr/local/bin中

Shell> ll /usr/local/bin/  | grep masterha
-r-xr-xr-x 1 root root  1995 Apr 12 05:40 masterha_check_repl      # 检查MySQL复制状态
-r-xr-xr-x 1 root root  1779 Apr 12 05:40 masterha_check_ssh       # 检查MHA的SSH状态
-r-xr-xr-x 1 root root  1865 Apr 12 05:40 masterha_check_status    # 检查当前MHA的状态
-r-xr-xr-x 1 root root  3201 Apr 12 05:40 masterha_conf_host       # 添加或删除配置的server信息
-r-xr-xr-x 1 root root  2517 Apr 12 05:40 masterha_manager         # 启动MHA
-r-xr-xr-x 1 root root  2165 Apr 12 05:40 masterha_master_monitor  # 检测master是否宕机
-r-xr-xr-x 1 root root  2373 Apr 12 05:40 masterha_master_switch   # 控制故障转移（手动或者自动）
-r-xr-xr-x 1 root root  5171 Apr 12 05:40 masterha_secondary_check # 通过slave检测master是否宕机（二次探测）
-r-xr-xr-x 1 root root  1739 Apr 12 05:40 masterha_stop            # MHA停止
```

创建必要的目录

```
##
## MHA Manager 端
##
# 配置文件存放
Shell> mkdir -p /etc/masterha

# 如果MHA Manager要管理多个MySQL Master-Slave组，还需要建立子目录
Shell> mkdir -p /var/log/masterha/app1   # 建议和MHA配置文件保持一致
```

2.3.2. MHA Node的安装

`MHA Node` 的安装包（*当前最新版本为 mha4mysql-node-0.57.tar.gz*）所有服务器均要安装（*包括MHA Manager*），过程如下所示：

```
# CentOS 7.0 ALL Server
# 预先安装 Perl 依赖
Shell> yum install perl-DBI perl-DBD-MySQL -y

Shell> tar zxvf mha4mysql-node-0.57.tar.gz

Shell> cd mha4mysql-node-0.57

Shell> make

Shell> make install

Shell> ll
total 44
-r-xr-xr-x 1 root root 16381 Apr 12 06:19 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
-r-xr-xr-x 1 root root  4807 Apr 12 06:19 filter_mysqlbinlog     # 去除不必要的ROLLBACK事件（脚本中指出该脚本已经过时，应该是被废弃了）
-r-xr-xr-x 1 root root  8261 Apr 12 06:19 purge_relay_logs       # 清除中继日志（不会阻塞SQL线程）
-r-xr-xr-x 1 root root  7525 Apr 12 06:19 save_binary_logs       # 保存和复制master的二进制日志
```

> **再次强调：`MHA Node` 在 所有节点 上都要安装**

2.4. SSH免密码登录

SSH免密码登录是为了能够让 `MHA Manager` 可以 远程登录 到各个MySQL Server，以及 各个`MySQL Server`之间可以 相互 远程登录 ，然后进行相关操作，避免输入密码。

2.4.1. MHA Manager生成Key

```
#
# MHA Manager 端
#
Shell> ssh-keygen -t rsa  # 生成RSA密钥对，如果原先有的可以跳过（大部分系统安装完后直接就有）

Shell> cd ~/.ssh/

Shell>ll
total 12
-rw------- 1 root root 1675 Feb  1 23:51 id_rsa        # 私钥，千万不能泄漏出去
-rw-r--r-- 1 root root  393 Feb  1 23:51 id_rsa.pub    # 公钥，下面就是要把这个公钥分发给所有MySQL Server
-rw-r--r-- 1 root root 1508 Feb 25 16:30 known_hosts
```

2.4.2. 公钥分发

1. **MHA Manager**
   将 `MHA Manager` 服务器的公钥，依次分发给 `Master`、 `Slave1` 和 `Slave2`

```
#
# MHA Manager 端
#
Shell> ssh-copy-id  -i ~/.ssh/id_rsa.pub root@172.18.14.70   # 172.18.14.70 是Master的IP
# 输入 接受密钥指纹的 yes
# 输入 172.18.14.70 的root 密码
# ssh root@172.18.14.70  ，可直接进入 Master节点

Shell> ssh-copy-id  -i ~/.ssh/id_rsa.pub root@172.18.14.71   # Slave1 节点

Shell> ssh-copy-id  -i ~/.ssh/id_rsa.pub root@172.18.14.72   # Slave2 节点
# 如果还有其他MySQL Server节点，依次执行 ssh-copy-id 操作
```

2. **MySQL Master**
   将 `Master` 的公钥依次分发给 `Slave1` 和 `Slave2`

```
Shell> ssh-copy-id  -i ~/.ssh/id_rsa.pub root@172.18.14.71
Shell> ssh-copy-id  -i ~/.ssh/id_rsa.pub root@172.18.14.72
# 如果还有其他MySQL Server节点，依次执行 ssh-copy-id 操作
```

3. **MySQL Slave1**
   将 `Slave1` 的公钥依次分发给 `Master` 和 `Slave2`

```
Shell> ssh-copy-id  -i ~/.ssh/id_rsa.pub root@172.18.14.70
Shell> ssh-copy-id  -i ~/.ssh/id_rsa.pub root@172.18.14.72
# 如果还有其他MySQL Server节点，依次执行 ssh-copy-id 操作
```

4. **MySQL Slave2**
   将 `Slave2` 的公钥依次分发给 `Master` 和 `Slave1`

```
Shell> ssh-copy-id  -i ~/.ssh/id_rsa.pub root@172.18.14.70
Shell> ssh-copy-id  -i ~/.ssh/id_rsa.pub root@172.18.14.71
# 如果还有其他MySQL Server节点，依次执行 ssh-copy-id 操作
```

注意：仅仅需要 `MySQL Server`节点之间 ，以及 `MHA Manager`到所有`MySQL Servers` 之间相互打通SSH免密码登录；MySQL Servers到MHA Manager并不需要

2.4.3. 修改hosts

为了方便使用主机名登录， 同时`MHA Manager`在`Failover`阶段会做`resolve name` 的操作，所以需要把下列信息加入到 所有的节点 中。

```
##
## 所有节点
##
Shell> vim /etc/hosts
# 添加如下信息
172.18.14.70 Master
172.18.14.71 Slave1
172.18.14.72 Slave2
```

2.4.4. 确认SSH打通

上述操作配置完成以后，在每个服务器上， 执行`ssh@IP` 和 `ssh@hostname` **的操作（*MySQL Server节点之间相互连接，以及 MHA Manager到所有 MySQL Server*），确认可以免密码登录成功。**

2.5. MHA环境说明

Role	IP	Hostname	Server_id	Desc
MHA Manager	172.18.14.80	MHA	N/A	MHA 管理服务器，用于监控
Master	172.18.14.70	Master	100	MySQL Master服务器，用于read-write
Candicate Master (*Slave*)	172.18.14.71	Slave1	101	MySQL Slave服务器，read-only，当Master宕机后，被提升为Master（New）
Slave	172.18.14.72	Slave2	102	MySQL Slave服务器，read-only

2.6. MySQL建立复制

由于之前的文档中已经演示了MySQL复制的建立，这里就不重复演示了。

为了最大限度的保证数据不丢失，强烈建议在 `MySQL 5.7` 下配置为 无损复制 。

复制关系建好以后，可以通过 `mysqlrplshow` 命令查看 复制拓扑

```
Shell> mysqlrplshow --master=root:123@172.18.14.70:3306 --discover-slaves-login=root:123 --verbose
WARNING: Using a password on the command line interface can be insecure.
# master on 172.18.14.70: ... connected.
# Finding slaves for master: 172.18.14.70:3306
WARNING: Unable to find aliases for hostname 'Master' reason: [Errno -2] Name or service not known
WARNING: IP lookup by address failed for 172.18.14.70,reason: Unknown host
# 这两个warning可以编辑 /etc/hosts 把IP和hostname 添加上即可消除警告

# Replication Topology Graph
172.18.14.70:3306 (MASTER)
   |
   +--- 172.18.14.71:3306 [IO: Yes, SQL: Yes] - (SLAVE)
   |
   +--- 172.18.14.72:3306 [IO: Yes, SQL: Yes] - (SLAVE)
```

> 注意： slave 中的 /etc/my.cnf 中需要配置 report-host=slave ip_addr ，否则上述命令执行失败

另外如果使用 mysqlreplicate 创建复制关系，则他会帮你在你的 Master 上创建 rpl@172.18.14.71 ， rpl@172.18.14.72 用户，但是并没有赋予 replication和slave 权限。
所以需要对这两个用户 先进行授权 ，否则在开启masterha_manager时，会有如下错误：

> [error][/usr/local/share/perl5/MHA/Server.pm, ln398] Slave1(172.18.14.71:3306): User rpl does not exist or does not have REPLICATION SLAVE privilege! Other slaves can not start replication from this host.

如果使用手工 CHANGE MASTER 操作，要确保 rpl用户 通过复制，已经 传递到Slave 上了，否则也会报上述错误。

2.7. MHA配置

2.7.1. MHA配置文件和说明

配置文件统一放在 /etc/masterha/ 中，针对当前这个MySQL复制组，我们定义为 app1.conf ，如果还有 其他MySQL复制组 ，可以 继续定义app2.conf / app3.conf等等

```
#
# MHA Manager 端  /etc/masterha/app1.conf
#
[server default]
# 这两个参数需要根据不同的集群进行修改
manager_workdir=/var/log/masterha/app1
manager_log=/var/log/masterha/app1/manager.log
# 按照master服务器存放binlog的实际路径进行修改，主要为了让MHA拉取binlog
master_binlog_dir=/data/mysql_data/5.7.11/
# 设置自动failover的脚本
master_ip_failover_script= /usr/local/bin/master_ip_failover
# 设置手动切换时候的脚本 （供(masterha_master_switch使用）
master_ip_online_change_script= /usr/local/bin/master_ip_online_change
log_level=debug
# 监控的用户
user=root
# 监控用户的密码
password=123
# 监控主库的时间间隔，默认是3秒，尝试三次没有回应的时候自动进行railover
ping_interval=3
# 检测方式是insert，MHA-0.56开始支持insert
# 会在Master中生成一个 infra 数据库
ping_type=INSERT
# 设置远端mysql在发生切换时binlog的保存位置
remote_workdir=/tmp
# 复制用的密码
repl_password=123
# 复制的用户
repl_user=rpl
# 告警脚本，可自行修改，这里没有使用
#report_script=/usr/local/send_report
# 通过从机进行二次探测的脚本， IP地址按照实际的情况进行修改
secondary_check_script=/usr/local/bin/masterha_secondary_check -s 172.18.14.71 -s 172.18.14.72 --user=root --master_host=172.18.14.70 --master_port=3306
# 设置故障发生后关闭故障主机的脚本（主要作用是关闭主机防止发生脑裂,这里没有使用，类似Fence功能）
#shutdown_script="/usr/local/bin/power_manager --command=stopssh2 --host=test-1 --ssh_user=root"
# 定义ssh的用户
ssh_user=root

[server1]
# 这个hostname也可以配置成IP地址，同 ip 参数一样
# 如果这里写名字，需要DNS配合，或者使用 /etc/hosts
hostname=Master
ip=172.18.14.70
port=3306
# candidate_master参数的意思为：设置为候选Master，如果发生主从切换，该主机会被提升为Master，即使这个服务器上的数据不是最新的（会用relay-log补全）
candidate_master=1

[server2]
hostname=Slave1
ip=172.18.14.71
port=3306
candidate_master=1
# check_repl_delay参数的意思为：默认情况下如果一个slave落后master 100M的relay logs的话，MHA将不会选择该slave作为一个新的master：
# 因为对于这个slave的恢复需要花费很长时间；
# 通过设置check_repl_delay=0,MHA触发切换在选择一个新的master的时候将会忽略复制延时；
# 这个参数对于设置了candidate_master=1的主机非常有用，因为这个候选主在切换的过程中一定是新的master
check_repl_delay=0

[server3]
hostname=Slave2
ip=172.18.14.72
port=3306
# no_master 表示该主机不会被提升为Master
no_master=1
```

> 官方文档 – 其他参数说明

2.7.2. MHA脚本说明

- **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 = '172.18.14.88/24';
# 注意修改系统对应的接口名称（CentOS比较变态）
my $eth = 'eth0';
my $key = '88';
my $ssh_start_vip = "/sbin/ifconfig $eth:$key $vip";
my $ssh_stop_vip = "/sbin/ifconfig $eth:$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";
        exit 0;
    }
    else {
        &usage();
        exit 1;
    }
}

sub start_vip() {
    `ssh $ssh_user\@$new_master_host \" $ssh_start_vip \"`;
}
sub stop_vip() {
     return 0  unless  ($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";
}
```

- **master_ip_online_change**
  从 master_ip_failover 修改而来

```
#!/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 (
  $command,              $orig_master_is_new_slave, $orig_master_host,
  $orig_master_ip,       $orig_master_port,         $orig_master_user,
  $orig_master_password, $orig_master_ssh_user,     $new_master_host,
  $new_master_ip,        $new_master_port,          $new_master_user,
  $new_master_password,  $new_master_ssh_user,
);


my $vip = '172.18.14.88/24';
# 注意修改系统对应的接口名称（CentOS比较变态）
my $eth = 'eth0';
my $key = '88';
my $ssh_start_vip = "/sbin/ifconfig $eth:$key $vip";
my $ssh_stop_vip = "/sbin/ifconfig $eth:$key down";
my $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,
    'orig_master_is_new_slave' => \$orig_master_is_new_slave,
    'orig_master_host=s'       => \$orig_master_host,
    'orig_master_ip=s'         => \$orig_master_ip,
    'orig_master_port=i'       => \$orig_master_port,
    'orig_master_user=s'       => \$orig_master_user,
    'orig_master_password=s'   => \$orig_master_password,
    'orig_master_ssh_user=s'   => \$orig_master_ssh_user,
    '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,
    'new_master_ssh_user=s'    => \$new_master_ssh_user,
);

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";
        exit 0;
    }
    else {
        &usage();
        exit 1;
    }
}

sub start_vip() {
    `ssh $ssh_user\@$new_master_host \" $ssh_start_vip \"`;
}
sub stop_vip() {
     return 0  unless  ($ssh_user);
    `ssh $ssh_user\@$orig_master_host \" $ssh_stop_vip \"`;
}

sub usage {
    print
    "Usage: master_ip_failover --command=start|stop|stopssh|status --ssh-user=user --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";
}
```

1. 修改上述两个脚本中的 `$vip` ，以符合自己实际的需求；
2. 将上述两个文件复制到 `MHA Manager` 节点的 `/usr/local/bin` 中（*同masterha_*在同一个目录*）；
3. 将上述两个文件赋予执行权限：
    - `chmod +x master_ip_failover`
    - `chmod +x master_ip_online_change`

2.7.3. Slave上的relay_log配置

从 `MHA Failover` 的过程中可以了解到， `MHA Manager` 在恢复（*补齐*）其他Slave数据时会用到 `relay-log` ，因此这些 `relay-log` 需要被保留。

而默认情况下，SQL线程在回放完毕后，MySQL会 主动删除`relay-log` ，需要 禁用 该功能，确保 `relay-log` 不被自动删除。

在 所有Slave节点 中配置如下参数，然后 重启mysql 服务即可。

```
# 所有的Slave节点
[mysqld]
# 关闭relay-log主动删除的功能
relay_log_purge = 0
```

但是这样做了以后又带来另外一个问题， `relay-log会大量堆积` ，导致磁盘空间紧张，所以需要 定时清空 过时的 `relay-log` 。

所幸的是MHA帮我们实现了这个功能， `MHA Node` 的安装包中有一个 `pure_relay_logs` 工具，提供 删除大量relay-log 的功能。

> 在Linux下删除体积较大的文件需要一定的时间，且消耗一定的资源，所以 `pure_relay_logs` 在删除 `relay-log` 之前，会做一次 硬连接 ，然后删除对应的 `relay-log` （*只 删除 指向 relay-log 的 指针* ），这样不会造成系统资源消耗，从而影响复制（*造成复制延时*），最后再 删除硬连接的`relay-log`（`_hardlink_`） 。

2.7.3.1. pure_relay_logs的参数

- `--user` ： 用户名
- `--password` ： 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 硬连接不能跨分区 ，所以请 确保这个目录和你的**`relay-log`**在同一个分区 ；
    - 当脚本执行成功后，硬链接会被删除
- `--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，该参数会将relay_log_purge设置为0。清理relay log之后，最后再设置 `relay_log_purge=0` 。
    - 但是还是要在 `/etc/my.cnf` 中显示配置 `relay_log_purge=0` ，避免重启服务后被还原。

2.7.3.2. 定时清理relay-log的脚本

将下面这个脚本以及定时任务放入 所有的Slave节点

```
#!/bin/bash
# Filename: /usr/local/bin/cron_purge_relay_logs.sh
user=root
passwd=123
port=3306
log_dir='/data/mysql_data/purge_relay_logs'
work_dir='/data/mysql_data/relay_log_hardlink'
purge='/usr/local/bin/purge_relay_logs'

if [[ ! -d ${work_dir} ]];then
   mkdir ${work_dir} -p
fi

if [[ ! -d ${log_dir} ]];then
   mkdir ${log_dir} -p
fi

${purge} --user=${user} --password=${passwd} --port=${port} --workdir=${work_dir} --disable_relay_log_purge >> ${log_dir}/purge_relay_logs.log 2>&1
```

增加执行权限

```
Shell> chmod +x /usr/local/bin/cron_purge_relay_logs.sh
```

添加到计划任务

```
Shell> crontab -e   # 编辑定时任务
# 输入如下
0 5 * * * /bin/bash /usr/local/bin/cron_purge_relay_logs.sh
# 保存退出

Shell> crontab -l    # 查看当前cron信息
0 5 * * * /bin/bash /usr/local/bin/cron_purge_relay_logs.sh
```

2.7.4. SSH检测

```
##
## MHA Manager端
##
Shell>  masterha_check_ssh  --conf=/etc/masterha/app1.conf
Wed Apr 13 00:52:45 2016 - [warning] Global configuration file /etc/masterha_default.cnf not found. Skipping.
Wed Apr 13 00:52:45 2016 - [info] Reading application default configuration from /etc/masterha/app1.conf..
Wed Apr 13 00:52:45 2016 - [info] Reading server configuration from /etc/masterha/app1.conf..
Wed Apr 13 00:52:45 2016 - [info] Starting SSH connection tests..
Wed Apr 13 00:52:46 2016 - [debug]
Wed Apr 13 00:52:45 2016 - [debug]  Connecting via SSH from root@Master(172.18.14.70:22) to root@Slave1(172.18.14.71:22)..
Wed Apr 13 00:52:45 2016 - [debug]   ok.
Wed Apr 13 00:52:45 2016 - [debug]  Connecting via SSH from root@Master(172.18.14.70:22) to root@Slave2(172.18.14.72:22)..
Wed Apr 13 00:52:45 2016 - [debug]   ok.
Wed Apr 13 00:52:46 2016 - [debug]
Wed Apr 13 00:52:46 2016 - [debug]  Connecting via SSH from root@Slave1(172.18.14.71:22) to root@Master(172.18.14.70:22)..
Warning: Permanently added '172.18.14.70' (ECDSA) to the list of known hosts.
Wed Apr 13 00:52:46 2016 - [debug]   ok.
Wed Apr 13 00:52:46 2016 - [debug]  Connecting via SSH from root@Slave1(172.18.14.71:22) to root@Slave2(172.18.14.72:22)..
Warning: Permanently added '172.18.14.72' (ECDSA) to the list of known hosts.
Wed Apr 13 00:52:46 2016 - [debug]   ok.
Wed Apr 13 00:52:47 2016 - [debug]
Wed Apr 13 00:52:46 2016 - [debug]  Connecting via SSH from root@Slave2(172.18.14.72:22) to root@Master(172.18.14.70:22)..
Warning: Permanently added '172.18.14.70' (ECDSA) to the list of known hosts.
Wed Apr 13 00:52:46 2016 - [debug]   ok.
Wed Apr 13 00:52:46 2016 - [debug]  Connecting via SSH from root@Slave2(172.18.14.72:22) to root@Slave1(172.18.14.71:22)..
Wed Apr 13 00:52:46 2016 - [debug]   ok.
Wed Apr 13 00:52:47 2016 - [info] All SSH connection tests passed successfully.
```

> 通过上面的输出信息可知，当前SSH免密码登录已经打通。

2.7.5. 复制关系检测

```
Shell> masterha_check_repl  --conf=/etc/masterha/app1.conf
Wed Apr 13 00:59:22 2016 - [warning] Global configuration file /etc/masterha_default.cnf not found. Skipping.
Wed Apr 13 00:59:22 2016 - [info] Reading application default configuration from /etc/masterha/app1.conf..
Wed Apr 13 00:59:22 2016 - [info] Reading server configuration from /etc/masterha/app1.conf..
Wed Apr 13 00:59:22 2016 - [info] MHA::MasterMonitor version 0.57.
Wed Apr 13 00:59:22 2016 - [debug] Connecting to servers..
Wed Apr 13 00:59:23 2016 - [debug]  Connected to: Master(172.18.14.70:3306), user=root
Wed Apr 13 00:59:23 2016 - [debug]  Number of slave worker threads on host Master(172.18.14.70:3306): 0
Wed Apr 13 00:59:23 2016 - [debug]  Connected to: Slave1(172.18.14.71:3306), user=root
Wed Apr 13 00:59:23 2016 - [debug]  Number of slave worker threads on host Slave1(172.18.14.71:3306): 4
Wed Apr 13 00:59:23 2016 - [debug]  Connected to: Slave2(172.18.14.72:3306), user=root
Wed Apr 13 00:59:23 2016 - [debug]  Number of slave worker threads on host Slave2(172.18.14.72:3306): 4
Wed Apr 13 00:59:23 2016 - [debug]  Comparing MySQL versions..
Wed Apr 13 00:59:23 2016 - [debug]   Comparing MySQL versions done.
Wed Apr 13 00:59:23 2016 - [debug] Connecting to servers done.
Wed Apr 13 00:59:23 2016 - [info] GTID failover mode = 1
Wed Apr 13 00:59:23 2016 - [info] Dead Servers:
Wed Apr 13 00:59:23 2016 - [info] Alive Servers:
Wed Apr 13 00:59:23 2016 - [info]   Master(172.18.14.70:3306)
Wed Apr 13 00:59:23 2016 - [info]   Slave1(172.18.14.71:3306)
Wed Apr 13 00:59:23 2016 - [info]   Slave2(172.18.14.72:3306)
Wed Apr 13 00:59:23 2016 - [info] Alive Slaves:
Wed Apr 13 00:59:23 2016 - [info]   Slave1(172.18.14.71:3306)  Version=5.7.11-log (oldest major version between slaves) log-bin:enabled
Wed Apr 13 00:59:23 2016 - [info]     GTID ON
Wed Apr 13 00:59:23 2016 - [debug]    Relay log info repository: TABLE
Wed Apr 13 00:59:23 2016 - [info]     Replicating from 172.18.14.70(172.18.14.70:3306)
Wed Apr 13 00:59:23 2016 - [info]     Primary candidate for the new Master (candidate_master is set)
Wed Apr 13 00:59:23 2016 - [info]   Slave2(172.18.14.72:3306)  Version=5.7.11-log (oldest major version between slaves) log-bin:enabled
Wed Apr 13 00:59:23 2016 - [info]     GTID ON
Wed Apr 13 00:59:23 2016 - [debug]    Relay log info repository: TABLE
Wed Apr 13 00:59:23 2016 - [info]     Replicating from 172.18.14.70(172.18.14.70:3306)
Wed Apr 13 00:59:23 2016 - [info]     Not candidate for the new Master (no_master is set)
Wed Apr 13 00:59:23 2016 - [info] Current Alive Master: Master(172.18.14.70:3306)
Wed Apr 13 00:59:23 2016 - [info] Checking slave configurations..
Wed Apr 13 00:59:23 2016 - [info]  read_only=1 is not set on slave Slave1(172.18.14.71:3306).
Wed Apr 13 00:59:23 2016 - [info]  read_only=1 is not set on slave Slave2(172.18.14.72:3306).
Wed Apr 13 00:59:23 2016 - [info] Checking replication filtering settings..
# 这里会检查 binlog_do_db和binlog_ignore_db的设置，如果不一致会报错
Wed Apr 13 00:59:23 2016 - [info]  binlog_do_db= , binlog_ignore_db=
Wed Apr 13 00:59:23 2016 - [info]  Replication filtering check ok.
Wed Apr 13 00:59:23 2016 - [info] GTID (with auto-pos) is supported. Skipping all SSH and Node package checking.
Wed Apr 13 00:59:23 2016 - [info] Checking SSH publickey authentication settings on the current master..
Wed Apr 13 00:59:23 2016 - [debug] SSH connection test to Master, option -o StrictHostKeyChecking=no -o PasswordAuthentication=no -o BatchMode=yes -o ConnectTimeout=5, timeout 5
Wed Apr 13 00:59:23 2016 - [info] HealthCheck: SSH to Master is reachable.
Wed Apr 13 00:59:23 2016 - [info]
Master(172.18.14.70:3306) (current master)
 +--Slave1(172.18.14.71:3306)
 +--Slave2(172.18.14.72:3306)

Wed Apr 13 00:59:23 2016 - [info] Checking replication health on Slave1..
Wed Apr 13 00:59:23 2016 - [info]  ok.
Wed Apr 13 00:59:23 2016 - [info] Checking replication health on Slave2..
Wed Apr 13 00:59:23 2016 - [info]  ok.
Wed Apr 13 00:59:23 2016 - [info] Checking master_ip_failover_script status:
Wed Apr 13 00:59:23 2016 - [info]   /usr/local/bin/master_ip_failover --command=status --ssh_user=root --orig_master_host=Master --orig_master_ip=172.18.14.70 --orig_master_port=3306


IN SCRIPT TEST====/sbin/ifconfig ens3:88 down==/sbin/ifconfig ens3:88 10.166.224.88/24===

Checking the Status of the script.. OK
Wed Apr 13 00:59:23 2016 - [info]  OK.
Wed Apr 13 00:59:23 2016 - [warning] shutdown_script is not defined.
Wed Apr 13 00:59:23 2016 - [debug]  Disconnected from Master(172.18.14.70:3306)
Wed Apr 13 00:59:23 2016 - [debug]  Disconnected from Slave1(172.18.14.71:3306)
Wed Apr 13 00:59:23 2016 - [debug]  Disconnected from Slave2(172.18.14.72:3306)
Wed Apr 13 00:59:23 2016 - [info] Got exit code 0 (Not master dead).

MySQL Replication Health is OK.
```

> 通过上面的输出信息可知，当前Master-Slave复制关系正常。

2.8. 启动MHA

2.8.1. 添加VIP

由于MHA的主要目标是 为了维护MySQL的复制关系，提供高可靠性的MySQL集群 ；至于Apps想如何访问这个集群，MHA其实没有规定的，留给我们自己决定，目前大概有如下几种访问方式：

1. **虚拟IP**
   - **方式一**： `Keepalived` ：Keepalived是Linux下的一个高可用组件，可以让两台或多台Linux主机组成一个高可用集群，对外提供一个虚拟IP进行访问（*也意味着同一时刻，只有一台主机能够被访问*）。
     - 我们这里可以在 `Master` 和 `Slave1` （ *`Candicate Master`* ）上安装keepalived，然后 检测mysql状态 （ *keepalived中可以定义检测脚本* ）；
       - 如果Master没有宕机，MySQL挂了，则Master上的Keepalived检测到MySQL挂了， 降低自己的优先级（低于`Slave1`） ，然后keepalived使用的 `VRRP` 协议（*心跳*）通过协商后，认为 `Slave1`的优先级更高 ，进而 `VIP` 转移到 `Slave1` ；
       - 如果Master直接宕机了，则 `Slave1`检测不到`Master`的存在 ，进而 `VIP` 也转移到 `Slave1` ；
     - keepalived会有可能产生脑裂，导致VIP在两个服务器上（*有两台服务器宣称自己的IP地址是VIP*），这样就会导致有的写操作落在Master上，而有的写操作落在Slave1上，从而到时整个集群的数据不一致；
       - 产生脑裂的原因有多种，大部分情况是网络问题；又或者是服务器太卡，导致心跳包没有及时处理；
   - **方式二**：使用脚本 `masterha_ip_failover` ；**由于这个脚本的start/stop操作只有在切换（Failover）期间才会执行，所以** 即便你设置了该脚本，**MHA**也不会在一开始（ ***masterha_manager启动*** ）的时候，帮你在**Master**上设置**VIP**，初次在**Master**上设置**VIP**需要人工操作，后期如果有**Failover**操作，**MHA**会执行脚本，帮你切换。
2. **智能DNS**
   此种模式下，MySQL集群本身只做高可用，维持复制关系，Apps通过内网DNS来访问MySQL。此种情况下，需要DNS端可以检测MySQL服务是否可用，从而决定推送Master还是Slave1的IP地址给Apps，类似Smart Client的效果；

我们这里 推荐使用 虚拟**IP**--方式二

```
##
## Master 端
##
# 在我的虚拟机中，网卡名称为ens3
Shell> ifconfig ens3:88 172.18.14.88/24
```

2.8.2. 启动masterha_manager

```
Shell> nohup masterha_manager --conf=/etc/masterha/app1.conf --remove_dead_master_conf --ignore_last_failover < /dev/null > /var/log/masterha/app1/manager.log 2>&1 &

Shell> masterha_check_status  --conf=/etc/masterha/app1.conf
app1 (pid:18702) is running(0:PING_OK), master:Master
# 状态正常，且当前主机是  Master
```

- **masterha_manager参数解释**
  - `--conf` ：当前MySQL集群的配置文件，可以有多个，应用于不同的集群
  - `--remove_dead_master_conf` ：当发生切换操作后（Failover），需要把之前的 `Dead Master` 从配置文件中删除。
    - 如果不删除，且没有恢复的话，此时 `masterha_manager` 重启后，会报错 `"there is a dead slave"`
  - `--ignore_last_failover` ：如果前一次Failover失败了，MHA不会去再一次去做Failover操作，除非人为的删除 `(manager_workdir)/(app_name).failover.error` ，或者增加此参数，MHA会继续进行Failover操作。

2.9. 自动Failover测试

> **强烈建议在做Failover测试之前，把MHA Manager停掉，把MySQL节点也都停掉（ 注意停止顺序 ），把数据库进行一次冷备份，方便以后测试**

测试步骤如下：

1. 停掉 `Slave的IO线程` ，模拟复制延时（Slave1保持不变）；
2. 使用sysbench对Master进行测试，生成测试数据（可以 产生大量的`binlog` ）；
3. 等待步骤2完成 后， 开启`Slave`上的`IO`线程 ，去追Master的binlog，同时立即操作第四步；
4. 关闭`Master`上的`MySQL` ，让MHA 产生 `Failover` 操作；
5. 观察最终状态；

> 注意： 步骤3 和 步骤4 不要调换，我多次测试发现会有问题；

测试之前的状态：

```
--
-- Master 端
--
mysql> show master status\G
*************************** 1. row ***************************
             File: bin.000004
         Position: 1062
     Binlog_Do_DB:
 Binlog_Ignore_DB:
Executed_Gtid_Set: 2d408b04-0154-11e6-83b6-5254f035dabc:1-137
1 row in set (0.00 sec)

--
-- Slave 1/2 端
--
mysql> show slave status\G
Slave_IO_State: Waiting for master to send event
                  Master_Host: 172.18.14.70
                  Master_User: rpl
                  Master_Port: 3306
                Connect_Retry: 60
              Master_Log_File: bin.000004
          Read_Master_Log_Pos: 1062
               Relay_Log_File: relay.000002
                Relay_Log_Pos: 1223
        Relay_Master_Log_File: bin.000004
             Slave_IO_Running: Yes
            Slave_SQL_Running: Yes
-- -------------省略其他输出--------------
           Retrieved_Gtid_Set: 2d408b04-0154-11e6-83b6-5254f035dabc:134-137
            Executed_Gtid_Set: 2d408b04-0154-11e6-83b6-5254f035dabc:1-137
-- -------------省略其他输出--------------
```

详细的测试过程如下：

2.9.1. 停止Slave的IO线程

```
--
-- Slave 1/2 端
--
mysql> stop slave io_thread;
Query OK, 0 rows affected (0.06 sec)
```

2.9.2. Master上产生大量binlog

通过sysbench模拟OLTP业务，产生大量的binlog。

```
--
-- Master端
--
mysql> create database sbtest;
Query OK, 1 row affected (0.01 sec)

mysql> commit;
Query OK, 0 rows affected (0.00 sec)
```

准备数据，这里在Master服务器上跑sysbench，仅仅为了产生binlog，而非为了测试性能。

```
##
## Master端
##
Shell> ./sysbench  --test=./tests/db/oltp.lua  --oltp-table-size=10000 --oltp-tables-count=1 --mysql-user=root --mysql-password=123 --mysql-host=172.18.14.88 --mysql-port=3306  --num-threads=4 --max-requests=0  --max-time=30 --report-interval=3 prepare
```

产生测试的binlig

```
##
## Master端
##
Shell> ./sysbench  --test=./tests/db/oltp.lua  --oltp-table-size=10000 --oltp-tables-count=1 --mysql-user=root --mysql-password=123 --mysql-host=172.18.14.88 --mysql-port=3306  --num-threads=4 --max-requests=0  --max-time=30 --report-interval=3 run
# --------省略输出--------
```

同时为了测试数据一致性，我们在sysbench跑完以后，再添加数据

```
--
-- Master 端
--
mysql> insert into burn_test.test_rpl select null;
Query OK, 1 row affected (0.00 sec)
Records: 1  Duplicates: 0  Warnings: 0

-- 以上命令运行11次，凑满20个数据

mysql> commit;  -- 由于开了autocommit=0 所以这个步骤一定要做
                -- 不然切换后，slave上没多的11个数据，要怀疑人生了
                -- 被坑惨了
Query OK, 0 rows affected (0.02 sec)

mysql> select * from burn_test.test_rpl;
+----+
| a  |
+----+
|  1 |
|  2 |
|  3 |
|  4 |
|  5 |
|  6 |
|  7 |
|  8 |
|  9 |
| 10 |
| 11 |
| 12 |
| 13 |
| 14 |
| 15 |
| 16 |
| 17 |
| 18 |
| 19 |
| 20 |
+----+
20 rows in set (0.01 sec)
```

2.9.3. 开启Slave的IO线程

```
--
-- Slave 1/2 端
--
mysql> start slave io_thread;
Query OK, 0 rows affected (0.00 sec)
```

2.9.4. 关闭Master的MySQL

```
##
## Master 端
##
Shell> ifconfig
ens3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 172.18.14.70  netmask 255.255.255.0  broadcast 172.18.14.255
        inet6 fe80::5054:f0ff:fe35:dabc  prefixlen 64  scopeid 0x20<link>
        ether 52:54:f0:35:da:bc  txqueuelen 1000  (Ethernet)
        RX packets 165824  bytes 24602903 (23.4 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 206430  bytes 205925027 (196.3 MiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

# 目前这个VIP还在Master上
ens3:88: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 172.18.14.88  netmask 255.255.255.0  broadcast 172.18.14.255
        ether 52:54:f0:35:da:bc  txqueuelen 1000  (Ethernet)

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 0  (Local Loopback)
        RX packets 147  bytes 13926 (13.5 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 147  bytes 13926 (13.5 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

# 停止MySQL，模拟宕机
Shell> service mysqld stop
Shutting down MySQL..... SUCCESS!
```

2.9.5. 观察MHA Manager的日志

```
Thu Apr 14 17:29:16 2016 - [warning] Got error on MySQL insert ping: 2006 (MySQL server has gone away)
# ----------------MHA 发现Master已经不通了------------------
# ----------启用二次探测的方式，进一步确认Master是否挂了---------
Thu Apr 14 17:29:16 2016 - [info] Executing secondary network check script: /usr/local/bin/masterha_secondary_check -s 172.18.14.71 -s 172.18.14.72 --user=root --master_host=172.18.14.70 --master_port=3306  --user=root  --master_host=Master  --master_ip=172.18.14.70  --master_port=3306 --master_user=root --master_password=123 --ping_type=INSERT
Thu Apr 14 17:29:16 2016 - [info] Executing SSH check script: exit 0
Thu Apr 14 17:29:16 2016 - [debug] SSH connection test to Master, option -o StrictHostKeyChecking=no -o PasswordAuthentication=no -o BatchMode=yes -o ConnectTimeout=5, timeout 5
Thu Apr 14 17:29:19 2016 - [warning] Got error on MySQL connect: 2003 (Can't connect to MySQL server on '172.18.14.70' (111))
Thu Apr 14 17:29:19 2016 - [warning] Connection failed 2 time(s)..
Thu Apr 14 17:29:21 2016 - [warning] HealthCheck: Got timeout on checking SSH connection to Master! at /usr/local/share/perl5/MHA/HealthCheck.pm line 342.
Thu Apr 14 17:29:22 2016 - [warning] Got error on MySQL connect: 2003 (Can't connect to MySQL server on '172.18.14.70' (111))
Thu Apr 14 17:29:22 2016 - [warning] Connection failed 3 time(s)..
Thu Apr 14 17:29:25 2016 - [warning] Got error on MySQL connect: 2003 (Can't connect to MySQL server on '172.18.14.70' (111))
Thu Apr 14 17:29:25 2016 - [warning] Connection failed 4 time(s)..
Monitoring server 172.18.14.71 is reachable, Master is not reachable from 172.18.14.71. OK. ## 通过 Slave1 检测，Master挂了
Monitoring server 172.18.14.72 is reachable, Master is not reachable from 172.18.14.72. OK. ## 通过 Slave2 检测，Master挂了
Thu Apr 14 17:29:36 2016 - [info] Master is not reachable from all other monitoring servers. Failover should start.
Thu Apr 14 17:29:36 2016 - [warning] Master is not reachable from health checker!
Thu Apr 14 17:29:36 2016 - [warning] Master Master(172.18.14.70:3306) is not reachable!
# ----------------------确认Master已经挂了----------------------
Thu Apr 14 17:29:36 2016 - [warning] SSH is NOT reachable.
Thu Apr 14 17:29:36 2016 - [info] Connecting to a master server failed. Reading configuration file /etc/masterha_default.cnf and /etc/masterha/app1.conf again, and trying to connect to all servers to check server status..
Thu Apr 14 17:29:36 2016 - [warning] Global configuration file /etc/masterha_default.cnf not found. Skipping.
# --------------------检查MHA配置----------------------
Thu Apr 14 17:29:36 2016 - [info] Reading application default configuration from /etc/masterha/app1.conf..
Thu Apr 14 17:29:36 2016 - [info] Reading server configuration from /etc/masterha/app1.conf..
Thu Apr 14 17:29:36 2016 - [debug] Skipping connecting to dead master Master(172.18.14.70:3306).
Thu Apr 14 17:29:36 2016 - [debug] Connecting to servers..
Thu Apr 14 17:29:36 2016 - [debug]  Connected to: Slave1(172.18.14.71:3306), user=root
Thu Apr 14 17:29:36 2016 - [debug]  Number of slave worker threads on host Slave1(172.18.14.71:3306): 4
Thu Apr 14 17:29:36 2016 - [debug]  Connected to: Slave2(172.18.14.72:3306), user=root
Thu Apr 14 17:29:36 2016 - [debug]  Number of slave worker threads on host Slave2(172.18.14.72:3306): 4
Thu Apr 14 17:29:36 2016 - [debug]  Comparing MySQL versions..
Thu Apr 14 17:29:36 2016 - [debug]   Comparing MySQL versions done.
Thu Apr 14 17:29:36 2016 - [debug] Connecting to servers done.
Thu Apr 14 17:29:36 2016 - [info] GTID failover mode = 1
Thu Apr 14 17:29:36 2016 - [info] Dead Servers:
Thu Apr 14 17:29:36 2016 - [info]   Master(172.18.14.70:3306)
Thu Apr 14 17:29:36 2016 - [info] Alive Servers:
Thu Apr 14 17:29:36 2016 - [info]   Slave1(172.18.14.71:3306)
Thu Apr 14 17:29:36 2016 - [info]   Slave2(172.18.14.72:3306)
Thu Apr 14 17:29:36 2016 - [info] Alive Slaves:
Thu Apr 14 17:29:36 2016 - [info]   Slave1(172.18.14.71:3306)  Version=5.7.11-log (oldest major version between slaves) log-bin:enabled
Thu Apr 14 17:29:36 2016 - [info]     GTID ON
Thu Apr 14 17:29:36 2016 - [debug]    Relay log info repository: TABLE
Thu Apr 14 17:29:36 2016 - [info]     Replicating from 172.18.14.70(172.18.14.70:3306)
Thu Apr 14 17:29:36 2016 - [info]     Primary candidate for the new Master (candidate_master is set)  ## 找到候选节点 Slave1
Thu Apr 14 17:29:36 2016 - [info]   Slave2(172.18.14.72:3306)  Version=5.7.11-log (oldest major version between slaves) log-bin:enabled
Thu Apr 14 17:29:36 2016 - [info]     GTID ON
Thu Apr 14 17:29:36 2016 - [debug]    Relay log info repository: TABLE
Thu Apr 14 17:29:36 2016 - [info]     Not candidate for the new Master (no_master is set)   ## Slave2 不是候选节点
Thu Apr 14 17:29:36 2016 - [info] Checking slave configurations..  ## 检查 Slave上的MySQL配置，如果 read_only=0 会有警告
Thu Apr 14 17:29:36 2016 - [info] Checking replication filtering settings..
Thu Apr 14 17:29:36 2016 - [info]  Replication filtering check ok.
Thu Apr 14 17:29:36 2016 - [info] Master is down!
Thu Apr 14 17:29:36 2016 - [info] Terminating monitoring script.
Thu Apr 14 17:29:36 2016 - [debug]  Disconnected from Slave1(172.18.14.71:3306)
Thu Apr 14 17:29:36 2016 - [debug]  Disconnected from Slave2(172.18.14.72:3306)
Thu Apr 14 17:29:36 2016 - [info] Got exit code 20 (Master dead).
Thu Apr 14 17:29:36 2016 - [info] MHA::MasterFailover version 0.57.
Thu Apr 14 17:29:36 2016 - [info] Starting master failover.
Thu Apr 14 17:29:36 2016 - [info]
#==================第一阶段：配置检查（包括MySQL的配置）==============
Thu Apr 14 17:29:36 2016 - [info] * Phase 1: Configuration Check Phase..
Thu Apr 14 17:29:36 2016 - [info]
Thu Apr 14 17:29:36 2016 - [debug] Skipping connecting to dead master Master.
Thu Apr 14 17:29:36 2016 - [debug] Connecting to servers..
Thu Apr 14 17:29:36 2016 - [debug]  Connected to: Slave1(172.18.14.71:3306), user=root
Thu Apr 14 17:29:36 2016 - [debug]  Number of slave worker threads on host Slave1(172.18.14.71:3306): 4
Thu Apr 14 17:29:36 2016 - [debug]  Connected to: Slave2(172.18.14.72:3306), user=root
Thu Apr 14 17:29:36 2016 - [debug]  Number of slave worker threads on host Slave2(172.18.14.72:3306): 4
Thu Apr 14 17:29:36 2016 - [debug]  Comparing MySQL versions..
Thu Apr 14 17:29:36 2016 - [debug]   Comparing MySQL versions done.
Thu Apr 14 17:29:36 2016 - [debug] Connecting to servers done.
Thu Apr 14 17:29:36 2016 - [info] GTID failover mode = 1
Thu Apr 14 17:29:36 2016 - [info] Dead Servers:
Thu Apr 14 17:29:36 2016 - [info]   Master(172.18.14.70:3306)
Thu Apr 14 17:29:36 2016 - [info] Alive Servers:
Thu Apr 14 17:29:36 2016 - [info]   Slave1(172.18.14.71:3306)
Thu Apr 14 17:29:36 2016 - [info]   Slave2(172.18.14.72:3306)
Thu Apr 14 17:29:36 2016 - [info] Alive Slaves:
Thu Apr 14 17:29:36 2016 - [info]   Slave1(172.18.14.71:3306)  Version=5.7.11-log (oldest major version between slaves) log-bin:enabled
Thu Apr 14 17:29:36 2016 - [info]     GTID ON
Thu Apr 14 17:29:36 2016 - [debug]    Relay log info repository: TABLE
Thu Apr 14 17:29:36 2016 - [info]     Replicating from 172.18.14.70(172.18.14.70:3306)
Thu Apr 14 17:29:36 2016 - [info]     Primary candidate for the new Master (candidate_master is set)
Thu Apr 14 17:29:36 2016 - [info]   Slave2(172.18.14.72:3306)  Version=5.7.11-log (oldest major version between slaves) log-bin:enabled
Thu Apr 14 17:29:36 2016 - [info]     GTID ON
Thu Apr 14 17:29:36 2016 - [debug]    Relay log info repository: TABLE
Thu Apr 14 17:29:36 2016 - [info]     Replicating from 172.18.14.70(172.18.14.70:3306)
Thu Apr 14 17:29:36 2016 - [info]     Not candidate for the new Master (no_master is set)
Thu Apr 14 17:29:37 2016 - [info] Starting GTID based failover.
Thu Apr 14 17:29:37 2016 - [info]
Thu Apr 14 17:29:37 2016 - [info] ** Phase 1: Configuration Check Phase completed.
# ========================第一阶段完成======================
Thu Apr 14 17:29:37 2016 - [info]
# ==================第二阶段：  开始清理Master==================
Thu Apr 14 17:29:37 2016 - [info] * Phase 2: Dead Master Shutdown Phase..
Thu Apr 14 17:29:37 2016 - [info]
Thu Apr 14 17:29:37 2016 - [info] Forcing shutdown so that applications never connect to the current master..
Thu Apr 14 17:29:37 2016 - [info] Executing master IP deactivation script:
Thu Apr 14 17:29:37 2016 - [info]   /usr/local/bin/master_ip_failover --orig_master_host=Master --orig_master_ip=172.18.14.70 --orig_master_port=3306 --command=stop
Thu Apr 14 17:29:37 2016 - [debug]  Stopping IO thread on Slave1(172.18.14.71:3306)..
Thu Apr 14 17:29:37 2016 - [debug]  Stopping IO thread on Slave2(172.18.14.72:3306)..


IN SCRIPT TEST====/sbin/ifconfig ens3:88 down==/sbin/ifconfig ens3:88 172.18.14.88/24===

Disabling the VIP on old master: Master   ## VIP被清除
Thu Apr 14 17:29:37 2016 - [info]  done.
Thu Apr 14 17:29:37 2016 - [warning] shutdown_script is not set. Skipping explicit shutting down of the dead master.
# shutdown_script 是关机脚本，为了防止脑裂，由于配置中被注释了，所以这里忽略
Thu Apr 14 17:29:37 2016 - [debug]  Stop IO thread on Slave2(172.18.14.72:3306) done.
Thu Apr 14 17:29:37 2016 - [debug]  Stop IO thread on Slave1(172.18.14.71:3306) done.
Thu Apr 14 17:29:37 2016 - [info] * Phase 2: Dead Master Shutdown Phase completed.
# ========================第二阶段完成======================
Thu Apr 14 17:29:37 2016 - [info]
# ====================第三阶段：Master日志拉取==================
# 如果Master服务器挂了，这个阶段应该跳过
Thu Apr 14 17:29:37 2016 - [info] * Phase 3: Master Recovery Phase..
Thu Apr 14 17:29:37 2016 - [info]
Thu Apr 14 17:29:37 2016 - [info] * Phase 3.1: Getting Latest Slaves Phase..
Thu Apr 14 17:29:37 2016 - [info]
Thu Apr 14 17:29:37 2016 - [debug] Fetching current slave status..
Thu Apr 14 17:29:37 2016 - [debug]  Fetching current slave status done.
Thu Apr 14 17:29:37 2016 - [info] The latest binary log file/position on all slaves is bin.000004:3215446  ## 得到Master上的binlog
Thu Apr 14 17:29:37 2016 - [info] Retrieved Gtid Set: 2d408b04-0154-11e6-83b6-5254f035dabc:134-3485 ## 以及对应的GTID
Thu Apr 14 17:29:37 2016 - [info] Latest slaves (Slaves that received relay log files to the latest): # 接收最新的relay-log
Thu Apr 14 17:29:37 2016 - [info]   Slave1(172.18.14.71:3306)  Version=5.7.11-log (oldest major version between slaves) log-bin:enabled
Thu Apr 14 17:29:37 2016 - [info]     GTID ON
Thu Apr 14 17:29:37 2016 - [debug]    Relay log info repository: TABLE
Thu Apr 14 17:29:37 2016 - [info]     Replicating from 172.18.14.70(172.18.14.70:3306)
Thu Apr 14 17:29:37 2016 - [info]     Primary candidate for the new Master (candidate_master is set)
Thu Apr 14 17:29:37 2016 - [info]   Slave2(172.18.14.72:3306)  Version=5.7.11-log (oldest major version between slaves) log-bin:enabled
Thu Apr 14 17:29:37 2016 - [info]     GTID ON
Thu Apr 14 17:29:37 2016 - [debug]    Relay log info repository: TABLE
Thu Apr 14 17:29:37 2016 - [info]     Replicating from 172.18.14.70(172.18.14.70:3306)
Thu Apr 14 17:29:37 2016 - [info]     Not candidate for the new Master (no_master is set)
Thu Apr 14 17:29:37 2016 - [info] The oldest binary log file/position on all slaves is bin.000004:5071556
Thu Apr 14 17:29:37 2016 - [info] Retrieved Gtid Set: 2d408b04-0154-11e6-83b6-5254f035dabc:134-3485
Thu Apr 14 17:29:37 2016 - [info] Oldest slaves:
Thu Apr 14 17:29:37 2016 - [info]   Slave1(172.18.14.71:3306)  Version=5.7.11-log (oldest major version between slaves) log-bin:enabled
Thu Apr 14 17:29:37 2016 - [info]     GTID ON
Thu Apr 14 17:29:37 2016 - [debug]    Relay log info repository: TABLE
Thu Apr 14 17:29:37 2016 - [info]     Replicating from 172.18.14.70(172.18.14.70:3306)
Thu Apr 14 17:29:37 2016 - [info]     Primary candidate for the new Master (candidate_master is set)
Thu Apr 14 17:29:37 2016 - [info]   Slave2(172.18.14.72:3306)  Version=5.7.11-log (oldest major version between slaves) log-bin:enabled
Thu Apr 14 17:29:37 2016 - [info]     GTID ON
Thu Apr 14 17:29:37 2016 - [debug]    Relay log info repository: TABLE
Thu Apr 14 17:29:37 2016 - [info]     Replicating from 172.18.14.70(172.18.14.70:3306)
Thu Apr 14 17:29:37 2016 - [info]     Not candidate for the new Master (no_master is set)
Thu Apr 14 17:29:37 2016 - [info]
# ----------------------选举新的Master------------------
Thu Apr 14 17:29:37 2016 - [info] * Phase 3.3: Determining New Master Phase..
Thu Apr 14 17:29:37 2016 - [info]
Thu Apr 14 17:29:37 2016 - [debug] Checking replication delay on Slave1(172.18.14.71:3306)..
Thu Apr 14 17:29:37 2016 - [debug]  ok.
Thu Apr 14 17:29:37 2016 - [info] Searching new master from slaves..
Thu Apr 14 17:29:37 2016 - [info]  Candidate masters from the configuration file:
Thu Apr 14 17:29:37 2016 - [info]   Slave1(172.18.14.71:3306)  Version=5.7.11-log (oldest major version between slaves) log-bin:enabled
Thu Apr 14 17:29:37 2016 - [info]     GTID ON
Thu Apr 14 17:29:37 2016 - [debug]    Relay log info repository: TABLE
Thu Apr 14 17:29:37 2016 - [info]     Replicating from 172.18.14.70(172.18.14.70:3306)
Thu Apr 14 17:29:37 2016 - [info]     Primary candidate for the new Master (candidate_master is set)
Thu Apr 14 17:29:37 2016 - [info]  Non-candidate masters:
Thu Apr 14 17:29:37 2016 - [info]   Slave2(172.18.14.72:3306)  Version=5.7.11-log (oldest major version between slaves) log-bin:enabled
Thu Apr 14 17:29:37 2016 - [info]     GTID ON
Thu Apr 14 17:29:37 2016 - [debug]    Relay log info repository: TABLE
Thu Apr 14 17:29:37 2016 - [info]     Replicating from 172.18.14.70(172.18.14.70:3306)
Thu Apr 14 17:29:37 2016 - [info]     Not candidate for the new Master (no_master is set)
Thu Apr 14 17:29:37 2016 - [info]  Searching from candidate_master slaves which have received the latest relay log events..
Thu Apr 14 17:29:37 2016 - [info] New master is Slave1(172.18.14.71:3306)
#----------------------开始切换 new master----------------
Thu Apr 14 17:29:37 2016 - [info] Starting master failover..
Thu Apr 14 17:29:37 2016 - [info]
From:
Master(172.18.14.70:3306) (current master)
 +--Slave1(172.18.14.71:3306)
 +--Slave2(172.18.14.72:3306)

To:
Slave1(172.18.14.71:3306) (new master)  ##  Slave1 提升为New Master
 +--Slave2(172.18.14.72:3306)
Thu Apr 14 17:29:37 2016 - [info]
# ------------------新master恢复--------------
Thu Apr 14 17:29:37 2016 - [info] * Phase 3.3: New Master Recovery Phase..
Thu Apr 14 17:29:37 2016 - [info]
# --------------------等待日志被应用------------
Thu Apr 14 17:29:37 2016 - [info]  Waiting all logs to be applied..
Thu Apr 14 17:29:37 2016 - [debug] Sql Thread Done: 0, Worker Thread done: 0, Ended workers: 3
Thu Apr 14 17:29:38 2016 - [debug] Sql Thread Done: 0, Worker Thread done: 0, Ended workers: 2
Thu Apr 14 17:29:39 2016 - [debug] Sql Thread Done: 0, Worker Thread done: 0, Ended workers: 2
Thu Apr 14 17:29:40 2016 - [debug] Sql Thread Done: 0, Worker Thread done: 0, Ended workers: 2
Thu Apr 14 17:29:41 2016 - [debug] Sql Thread Done: 0, Worker Thread done: 0, Ended workers: 1
Thu Apr 14 17:29:42 2016 - [debug] Sql Thread Done: 0, Worker Thread done: 0, Ended workers: 3
Thu Apr 14 17:29:43 2016 - [debug] Sql Thread Done: 0, Worker Thread done: 0, Ended workers: 3
Thu Apr 14 17:29:44 2016 - [debug] Sql Thread Done: 0, Worker Thread done: 0, Ended workers: 3
Thu Apr 14 17:29:45 2016 - [debug] Sql Thread Done: 0, Worker Thread done: 0, Ended workers: 3
Thu Apr 14 17:29:46 2016 - [debug] Sql Thread Done: 0, Worker Thread done: 0, Ended workers: 1
```

```
Thu Apr 14 17:29:47 2016 - [debug] Sql Thread Done: 0, Worker Thread done: 0, Ended workers: 3
Thu Apr 14 17:29:48 2016 - [debug] Sql Thread Done: 0, Worker Thread done: 0, Ended workers: 3
Thu Apr 14 17:29:49 2016 - [debug] Sql Thread Done: 0, Worker Thread done: 0, Ended workers: 3
Thu Apr 14 17:29:50 2016 - [debug] Sql Thread Done: 0, Worker Thread done: 0, Ended workers: 1
Thu Apr 14 17:29:51 2016 - [debug] Sql Thread Done: 0, Worker Thread done: 0, Ended workers: 2
Thu Apr 14 17:29:52 2016 - [debug] Sql Thread Done: 0, Worker Thread done: 0, Ended workers: 3
Thu Apr 14 17:29:53 2016 - [debug] Sql Thread Done: 0, Worker Thread done: 0, Ended workers: 2
Thu Apr 14 17:29:54 2016 - [debug] Sql Thread Done: 0, Worker Thread done: 0, Ended workers: 3
Thu Apr 14 17:29:55 2016 - [debug] Sql Thread Done: 0, Worker Thread done: 0, Ended workers: 2
Thu Apr 14 17:29:56 2016 - [debug] Sql Thread Done: 0, Worker Thread done: 0, Ended workers: 2
Thu Apr 14 17:29:57 2016 - [debug] Sql Thread Done: 0, Worker Thread done: 0, Ended workers: 1
Thu Apr 14 17:29:58 2016 - [debug] Sql Thread Done: 0, Worker Thread done: 0, Ended workers: 3
Thu Apr 14 17:29:59 2016 - [debug] Sql Thread Done: 0, Worker Thread done: 0, Ended workers: 2
Thu Apr 14 17:30:00 2016 - [debug] Sql Thread Done: 0, Worker Thread done: 0, Ended workers: 0
Thu Apr 14 17:30:01 2016 - [debug] Sql Thread Done: 0, Worker Thread done: 0, Ended workers: 3
Thu Apr 14 17:30:02 2016 - [debug] Sql Thread Done: 0, Worker Thread done: 0, Ended workers: 3
Thu Apr 14 17:30:03 2016 - [debug] Sql Thread Done: 0, Worker Thread done: 0, Ended workers: 2
Thu Apr 14 17:30:04 2016 - [debug] Sql Thread Done: 0, Worker Thread done: 0, Ended workers: 3
Thu Apr 14 17:30:05 2016 - [debug] Sql Thread Done: 0, Worker Thread done: 0, Ended workers: 2
Thu Apr 14 17:30:06 2016 - [debug] Sql Thread Done: 0, Worker Thread done: 0, Ended workers: 2
Thu Apr 14 17:30:07 2016 - [debug] Sql Thread Done: 0, Worker Thread done: 0, Ended workers: 2
Thu Apr 14 17:30:08 2016 - [debug] Sql Thread Done: 0, Worker Thread done: 0, Ended workers: 3
Thu Apr 14 17:30:09 2016 - [debug] Sql Thread Done: 0, Worker Thread done: 0, Ended workers: 2
Thu Apr 14 17:30:10 2016 - [debug] Sql Thread Done: 0, Worker Thread done: 0, Ended workers: 2
Thu Apr 14 17:30:11 2016 - [debug] Sql Thread Done: 0, Worker Thread done: 0, Ended workers: 2
Thu Apr 14 17:30:12 2016 - [info]   done.
Thu Apr 14 17:30:12 2016 - [debug]  Stopping slave IO/SQL thread on Slave1(172.18.14.71:3306)..
Thu Apr 14 17:30:12 2016 - [debug]   done.
# --------------Slave 2同步最新的日志------------
Thu Apr 14 17:30:12 2016 - [info]  Replicating from the latest slave Slave2(172.18.14.72:3306) and waiting to apply..
Thu Apr 14 17:30:12 2016 - [info]  Waiting all logs to be applied on the latest slave..
Thu Apr 14 17:30:12 2016 - [info]  Resetting slave Slave1(172.18.14.71:3306) and starting replication from the new master Slave2(172.18.14.72:3306)..
Thu Apr 14 17:30:12 2016 - [debug]  Stopping slave IO/SQL thread on Slave1(172.18.14.71:3306)..
Thu Apr 14 17:30:12 2016 - [debug]   done.
# ------------------执行Change Master操作--------------
Thu Apr 14 17:30:12 2016 - [info]  Executed CHANGE MASTER.
Thu Apr 14 17:30:12 2016 - [debug]  Starting slave IO/SQL thread on Slave1(172.18.14.71:3306)..
Thu Apr 14 17:30:12 2016 - [debug]   done.
Thu Apr 14 17:30:12 2016 - [info]  Slave started.
Thu Apr 14 17:30:12 2016 - [info]  Waiting to execute all relay logs on Slave1(172.18.14.71:3306)..
Thu Apr 14 17:30:12 2016 - [info]  master_pos_wait(bin.000002:3192427) completed on Slave1(172.18.14.71:3306). Executed 1 events.
Thu Apr 14 17:30:12 2016 - [info]   done.
Thu Apr 14 17:30:12 2016 - [debug]  Stopping SQL thread on Slave1(172.18.14.71:3306)..
Thu Apr 14 17:30:12 2016 - [debug]   done.
Thu Apr 14 17:30:12 2016 - [info]   done.
Thu Apr 14 17:30:12 2016 - [info] Getting new master's binlog name and position..
Thu Apr 14 17:30:12 2016 - [info]  bin.000002:3192427
Thu Apr 14 17:30:12 2016 - [info]  All other slaves should start replication from here. Statement should be: CHANGE MASTER TO MASTER_HOST='Slave1 or 172.18.14.71', MASTER_PORT=3306, MASTER_AUTO_POSITION=1, MASTER_USER='rpl', MASTER_PASSWORD='xxx';
# ------------------Master恢复完成----------------
Thu Apr 14 17:30:12 2016 - [info] Master Recovery succeeded. File:Pos:Exec_Gtid_Set: bin.000002, 3192427, 2d408b04-0154-11e6-83b6-5254f035dabc:1-2260
Thu Apr 14 17:30:12 2016 - [info] Executing master IP activate script:
Thu Apr 14 17:30:12 2016 - [info]    /usr/local/bin/master_ip_failover --command=start --ssh_user=root --orig_master_host=Master --orig_master_ip=172.18.14.70 --orig_master_port=3306 --new_master_host=Slave1 --new_master_ip=172.18.14.71 --new_master_port=3306 --new_master_user='root'   --new_master_password=xxx
Unknown option: new_master_user   # 这里的告警无所谓，因为ssh打通了
Unknown option: new_master_password  #  是否需要密码无所谓了


IN SCRIPT TEST====/sbin/ifconfig ens3:88 down==/sbin/ifconfig ens3:88 172.18.14.88/24===

# ----------------将 VIP 漂移到 Slave1 上----------------
Enabling the VIP - 172.18.14.88/24 on the new master - Slave1
Thu Apr 14 17:30:22 2016 - [info]  OK.
Thu Apr 14 17:30:22 2016 - [info] Setting read_only=0 on Slave1(172.18.14.71:3306)..
Thu Apr 14 17:30:22 2016 - [info]  ok.
Thu Apr 14 17:30:22 2016 - [info] ** Finished master recovery successfully.
Thu Apr 14 17:30:22 2016 - [info] * Phase 3: Master Recovery Phase completed.
# ====================== 第三阶段 完成 ======================
# ====================== 第四阶段 slave恢复 ======================
Thu Apr 14 17:30:22 2016 - [info]
Thu Apr 14 17:30:22 2016 - [info] * Phase 4: Slaves Recovery Phase..
Thu Apr 14 17:30:22 2016 - [info]
Thu Apr 14 17:30:22 2016 - [info]
Thu Apr 14 17:30:22 2016 - [info] * Phase 4.1: Starting Slaves in parallel..
Thu Apr 14 17:30:22 2016 - [info]
Thu Apr 14 17:30:22 2016 - [info] -- Slave recovery on host Slave2(172.18.14.72:3306) started, pid: 5304. Check tmp log /var/log/masterha/app1/Slave2_3306_20160414172936.log if it takes time..
Thu Apr 14 17:30:23 2016 - [info]
Thu Apr 14 17:30:23 2016 - [info] Log messages from Slave2 ...
Thu Apr 14 17:30:23 2016 - [info]
Thu Apr 14 17:30:22 2016 - [info]  Resetting slave Slave2(172.18.14.72:3306) and starting replication from the new master Slave1(172.18.14.71:3306)..
Thu Apr 14 17:30:22 2016 - [debug]  Stopping slave IO/SQL thread on Slave2(172.18.14.72:3306)..
Thu Apr 14 17:30:22 2016 - [debug]   done.
Thu Apr 14 17:30:23 2016 - [info]  Executed CHANGE MASTER. ## 执行change master
Thu Apr 14 17:30:23 2016 - [debug]  Starting slave IO/SQL thread on Slave2(172.18.14.72:3306)..
Thu Apr 14 17:30:23 2016 - [debug]   done.
Thu Apr 14 17:30:23 2016 - [info]  Slave started.
Thu Apr 14 17:30:23 2016 - [info]  gtid_wait(2d408b04-0154-11e6-83b6-5254f035dabc:1-2260) completed on Slave2(172.18.14.72:3306). Executed 0 events.
Thu Apr 14 17:30:23 2016 - [info] End of log messages from Slave2.
Thu Apr 14 17:30:23 2016 - [info] -- Slave on host Slave2(172.18.14.72:3306) started.
Thu Apr 14 17:30:23 2016 - [info] All new slave servers recovered successfully.
Thu Apr 14 17:30:23 2016 - [info]
# ====================== 第四阶段 完成，所有从机恢复完成 ======================
# ====================== 第四阶段 New Master清理 ======================
Thu Apr 14 17:30:23 2016 - [info] * Phase 5: New master cleanup phase..
Thu Apr 14 17:30:23 2016 - [info]
Thu Apr 14 17:30:23 2016 - [info] Resetting slave info on the new master..
Thu Apr 14 17:30:23 2016 - [debug]  Clearing slave info..
Thu Apr 14 17:30:23 2016 - [debug]  Stopping slave IO/SQL thread on Slave1(172.18.14.71:3306)..
Thu Apr 14 17:30:23 2016 - [debug]   done.
Thu Apr 14 17:30:23 2016 - [debug]  SHOW SLAVE STATUS shows new master does not replicate from anywhere. OK.
Thu Apr 14 17:30:23 2016 - [info]  Slave1: Resetting slave info succeeded.
Thu Apr 14 17:30:23 2016 - [info] Master failover to Slave1(172.18.14.71:3306) completed successfully.
Thu Apr 14 17:30:23 2016 - [info] Deleted server1 entry from /etc/masterha/app1.conf .
Thu Apr 14 17:30:23 2016 - [debug]  Disconnected from Slave1(172.18.14.71:3306)
Thu Apr 14 17:30:23 2016 - [debug]  Disconnected from Slave2(172.18.14.72:3306)
Thu Apr 14 17:30:23 2016 - [info]

----- Failover Report -----

app1: MySQL Master failover Master(172.18.14.70:3306) to Slave1(172.18.14.71:3306) succeeded

Master Master(172.18.14.70:3306) is down!

Check MHA Manager logs at MHA:/var/log/masterha/app1/manager.log for details.

Started automated(non-interactive) failover.
Invalidated master IP address on Master(172.18.14.70:3306)
Selected Slave1(172.18.14.71:3306) as a new master.
Slave1(172.18.14.71:3306): OK: Applying all logs succeeded.
Slave1(172.18.14.71:3306): OK: Activated master IP address.
Slave2(172.18.14.72:3306): OK: Slave started, replicating from Slave1(172.18.14.71:3306)
Slave1(172.18.14.71:3306): Resetting slave info succeeded.
Master failover to Slave1(172.18.14.71:3306) completed successfully.
# ======================在线切换成功======================
```

从上面的日志可以了解到Failover过程，大致包含如下步骤：

1. 多次探测（ *包括透过Slave进行二次探测* ）Master，确认是否宕机；
2. 检查配置文件阶段， 包括MySQL实例的当前的配置 ；
3. 处理Master，前提是还能ssh到Master，否则跳过该步骤；
   - 包括删除虚拟IP，还有关机操作（防止脑裂，但是关机配置我们注释了）；
4. 如果还能SSH到Master上，则 复制binlog 到MHA Manager上，否则跳过；
   - 这一步在源码和之前网上别人的日志中是能看到 scp 信息的，但是我这里没出现scp信息，不知道是否是MHA0.57的问题，但是最终数据是能一致（表明确实是获取binlog了）
   - 从这里看出，单纯的靠MHA是不能完全保证数据不丢的；
5. 确认包含最新更新的Slave ；
6. 应用从master保存的binlog ；
7. 将配置中配置为 candidate_master=1 的Slave，提升为 New Master ；
8. 配置其他的Slave连接（ *CHANGE MASTER TO* ） New Master；

2.9.6. 查看最终复制关系

```
##
## Slave 1 端
##
Shell> ifconfig
ens3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 172.18.14.71  netmask 255.255.255.0  broadcast 172.18.14.255
        inet6 fe80::5054:f0ff:fe34:66c1  prefixlen 64  scopeid 0x20<link>
        ether 52:54:f0:34:66:c1  txqueuelen 1000  (Ethernet)
        RX packets 120664  bytes 14418436 (13.7 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 98765  bytes 115162049 (109.8 MiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

# 虚拟IP漂移到了Slave1
ens3:88: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 172.18.14.88  netmask 255.255.255.0  broadcast 172.18.14.255
        ether 52:54:f0:34:66:c1  txqueuelen 1000  (Ethernet)

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 0  (Local Loopback)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0


--
--  Slave 1 端 （New Master）
--
mysql> show master status\G
*************************** 1. row ***************************
             File: bin.000002
         Position: 5035941
     Binlog_Do_DB:
 Binlog_Ignore_DB:
Executed_Gtid_Set: 2d408b04-0154-11e6-83b6-5254f035dabc:1-3485
1 row in set (0.00 sec)
```

```
--
-- Slave 2 端
--
mysql> show slave status\G
*************************** 1. row ***************************
               Slave_IO_State: Waiting for master to send event
                  Master_Host: 172.18.14.71  -- 已经指向了 Slave 1
                  Master_User: rpl
                  Master_Port: 3306
                Connect_Retry: 60
              Master_Log_File: bin.000002
          Read_Master_Log_Pos: 5035941
               Relay_Log_File: relay.000002
                Relay_Log_Pos: 396
        Relay_Master_Log_File: bin.000002
             Slave_IO_Running: Yes    -- IO 线程正常
            Slave_SQL_Running: Yes    -- Slave 线程正常
              Replicate_Do_DB:
          Replicate_Ignore_DB:
           Replicate_Do_Table:
       Replicate_Ignore_Table:
      Replicate_Wild_Do_Table:
  Replicate_Wild_Ignore_Table:
                   Last_Errno: 0
                   Last_Error:
                 Skip_Counter: 0
          Exec_Master_Log_Pos: 5035941
              Relay_Log_Space: 593
              Until_Condition: None
               Until_Log_File:
                Until_Log_Pos: 0
           Master_SSL_Allowed: No
           Master_SSL_CA_File:
           Master_SSL_CA_Path:
              Master_SSL_Cert:
            Master_SSL_Cipher:
               Master_SSL_Key:
        Seconds_Behind_Master: 0
Master_SSL_Verify_Server_Cert: No
                Last_IO_Errno: 0
                Last_IO_Error:
               Last_SQL_Errno: 0
               Last_SQL_Error:
  Replicate_Ignore_Server_Ids:
             Master_Server_Id: 101
                  Master_UUID: 34b1b436-0154-11e6-8558-5254f03466c1
             Master_Info_File: mysql.slave_master_info
                    SQL_Delay: 0
          SQL_Remaining_Delay: NULL
      Slave_SQL_Running_State: Slave has read all relay log; waiting for more updates
           Master_Retry_Count: 86400
                  Master_Bind:
      Last_IO_Error_Timestamp:
     Last_SQL_Error_Timestamp:
               Master_SSL_Crl:
           Master_SSL_Crlpath:
           Retrieved_Gtid_Set:
            Executed_Gtid_Set: 2d408b04-0154-11e6-83b6-5254f035dabc:1-3485
                Auto_Position: 1
         Replicate_Rewrite_DB:
                 Channel_Name:
           Master_TLS_Version:
1 row in set (0.01 sec)

-- 同时查看一下burn_test.test_rpl的数据是否都过来了

mysql> select * from burn_test.test_rpl;
+----+
| a  |
+----+
|  1 |
|  2 |
|  3 |
|  4 |
|  5 |
|  6 |
|  7 |
|  8 |
|  9 |
| 10 |
| 11 |
| 12 |
| 13 |
| 14 |
| 15 |
| 16 |
| 17 |
| 18 |
| 19 |
| 20 |
+----+
20 rows in set (0.00 sec)
-- 最后宕机前追加的数据都过来了
```

2.9.7. 查看masterha_manager的配置

```
[server default]
log_level=debug
manager_log=/var/log/masterha/app1/manager.log
manager_workdir=/var/log/masterha/app1
master_binlog_dir=/data/mysql_data/5.7.11/
master_ip_failover_script=/usr/local/bin/master_ip_failover
master_ip_online_change_script=/usr/local/bin/master_ip_online_change
password=123
ping_interval=3
ping_type=INSERT
remote_workdir=/tmp
repl_password=123
repl_user=rpl
secondary_check_script=/usr/local/bin/masterha_secondary_check -s 172.18.14.71 -s 172.18.14.72 --user=root --master_host=172.18.14.70 --mast
er_port=3306
ssh_user=root
user=root
#-----------------> 这里原来的[server1]被 自动 删除了
[server2]
candidate_master=1
check_repl_delay=0
hostname=Slave1
ip=172.18.14.71
port=3306

[server3]
hostname=Slave2
ip=172.18.14.72
no_master=1
port=3306
```

原来的配置中是存在大量注释的，且含有 `[server1]` 的配置（也就是Master）；
现在由于 `masterha_manager` 配置了 `--remove_dead_master_conf` 的参数，在 `Failover完成后` ，自动将 `Dead Master` 从配置文件中去除，这样可以确保 `重新启动masterha_manager` 时，不会把 `Dead Master` 加入到MySQL集群中去，从而引发错误。

2.10. 相关注意事项

2.10.1. Failover后MHA Manager自动退出

当自动Failover完成后，MHA Manager服务器上的 `masterha_manager` 进程 `自动退出` ，官方文档的解释如下：

> Currently MHA Manager process does not run as a daemon. `If failover completed successfully or the master process was killed by accident, the manager stops working.` To run as a daemon, daemontool. or any external daemon program can be used. Here is an example to run from daemontools.

官方示例中使用 `daemontools` 进行托管，但是 `CentOS-7` 上已经不提供yum安装，需要手工编译，而且需要放到systemd中去，相对比较麻烦

这里使用 `supervisor` 来进行托管（*python程序员用的很多*），该软件的主要作用就是托管程序，并且可以简单的检测程序状态，发现停止后可以自动重启运行；

> 注意：在MHA重启后， `日志` 部分会被 `自动回滚覆盖` ，MHA日志中的binlog的filename和pos、以及GTID信息也会消失。所以是否自动重启还是需要看DBA和业务需要。

- **supervisord安装**

```
Shell> yum install supervisor
```

- **配置文件/etc/supervisord.conf**

```
[unix_http_server]
file=/var/run/supervisor.sock   ; (the path to the socket file)


[supervisord]
logfile=/var/log/supervisord.log ; (main log file;default $CWD/supervisord.log)
logfile_maxbytes=50MB        ; (max main logfile bytes b4 rotation;default 50MB)
logfile_backups=10           ; (num of main logfile rotation backups;default 10)
loglevel=info                ; (log level;default info; others: debug,warn,trace)
pidfile=/var/run/supervisord.pid ; (supervisord pidfile;default supervisord.pid)
nodaemon=false               ; (start in foreground if true;default false)
minfds=1024                  ; (min. avail startup file descriptors;default 1024)
minprocs=200                 ; (min. avail process descriptors;default 200)

[rpcinterface:supervisor]
supervisor.rpcinterface_factory = supervisor.rpcinterface:make_main_rpcinterface

[supervisorctl]
serverurl=unix:///var/run/supervisor.sock ; use a unix:// URL  for a unix socket

[program:mha_manager]
command=nohup /usr/local/bin/masterha_manager --conf=/etc/masterha/app1.conf --remove_dead_master_conf --ignore_last_failover
process_name=%(program_name)s
autostart=true
autorestart=true
redirect_stderr=true
stdout_logfile=/var/log/masterha/app1/manager.log
stdout_logfile_maxbytes=10MB
stdout_logfile_backups=5
user=root
```

- **启动supervisord**

```
Shell> systemctl start supervisord.service

# 开启自启动
Shell> systemctl enabled supervisord.service
```

- **supervisord测试**

```
Shell> ps -ef |grep perl
root     18564 18563  0 23:01 ?        00:00:01 perl /usr/local/bin/masterha_manager --conf=/etc/masterha/app1.conf --remove_dead_master_conf --ignore_last_failover
#---------省略其他输出----------

# 当前 masterha_manager 的PID为 18564

Shell> kill  -9 18564   # kill -9 强行杀死
Shell> ps -ef |grep perl
root     20559 18563  0 23:37 ?        00:00:00 perl /usr/local/bin/masterha_manager --conf=/etc/masterha/app1.conf --remove_dead_master_conf --ignore_last_failover
#---------省略其他输出----------

# 当前 masterha_manager 的PID更新为 20559


Shell> kill  -9  20559  # 再次 kill -9 强行杀死
Shell> ps -ef |grep perl
root     20584 18563  3 23:38 ?        00:00:00 perl /usr/local/bin/masterha_manager --conf=/etc/masterha/app1.conf --remove_dead_master_conf --ignore_last_failover

# 当前 masterha_manager 的PID更新为 20584
```

且对应的supervisord有对应的日志输出

```
Shell> cat  /var/log/supervisord.log
#---------省略其他输出----------
2016-04-13 23:01:13,083 INFO spawned: 'mha_manager' with pid 18564
2016-04-13 23:01:14,282 INFO success: mha_manager entered RUNNING state, process has stayed up for > than 1 seconds (startsecs)
2016-04-13 23:37:34,043 INFO exited: mha_manager (terminated by SIGKILL; not expected)
2016-04-13 23:37:35,047 INFO spawned: 'mha_manager' with pid 20559
2016-04-13 23:37:36,254 INFO success: mha_manager entered RUNNING state, process has stayed up for > than 1 seconds (startsecs)
2016-04-13 23:38:01,131 INFO exited: mha_manager (terminated by SIGKILL; not expected)
2016-04-13 23:38:02,135 INFO spawned: 'mha_manager' with pid 20584
2016-04-13 23:38:03,327 INFO success: mha_manager entered RUNNING state, process has stayed up for > than 1 seconds (startsecs)
```

注意：如果 `masterha_manager`可以自动重启 后，那一定要开启 `--remove_dead_master_conf` 参数。

因为既然切换（Failover）了（*masterha_manager默认自动退出*），说明配置文件列表中有server是宕机了，需要踢出MySQL集群，那如果此时没有配置该参数，当masterha_manager自动重启时， 仍能读取到宕机的server的配置 ，就会启动失败，然后一直循环启动。

> 但是注意，如果 重新启动 的时候，如果 节点数 少于等于 2台 ，仍然 启动失败 ；
> 日志中会报 `None of slaves can be master` ，即没有Slave可以提升为Master（ MHA强制要求MySQL节点必须不少于3台 ）
>
> 这样会导致supervisord频繁重启 [mha_manager] 这个服务，但是会由于太频繁，在若干次之后，被强制停止，所以无需担心。

2.11. 手动Failover测试

2.11.1. 注意事项

在使用手动Failover的时候，masterha_manager（在线Failover功能）需要关闭（ *自动切换还是手工切换自己选择* ），这个也是MHA留给我们自己决定的地方，线上使用可以选择使用自动切换，也可以通过报警后，处理问题，然后决定是否手工切换。

手工切换命令如下：

```
##
## MHA Manager 端
##
Shell> masterha_master_switch --master_state=dead --conf=/etc/masterha/app1.conf --dead_master_host=Master --dead_master_port=3306 --new_master_host=Slave1 --new_master_port=3306 --ignore_last_failover
```

> 注意事项：
>
> 1. **如果你的 `app1.conf` 中配置的 `hostname` 为 `Master`、`Slave1` 等，则上述命令中的 `--dead_master_host` 和 `--new_master_host` 也需要写成 `Master`、`Slave1` ，而** 不能写成**`IP`**地址
> 2. **如果你的 `app1.conf` 中配置的 `hostname` 为IP地址，则 `--dead_master_host` 和 `--new_master_host` 也要配置成IP地址**

如果这里配置错误，会报如下错误：

```
Detected dead master Master(172.18.14.70:3306) does not match with specified dead master 172.18.14.70(172.18.14.70:3306)!
```

在进行手动Failover的时候，请确保Master上的MySQL挂了，否则会报如下错误：

```
None of server is dead. Stop failover.  # 提示我们没有服务器挂
```

2.11.2. 切换测试

```
##
## Master 端
##
Shell> service mysqld stop


##
## MHA Manager 端
##
Shell> masterha_master_switch --master_state=dead --conf=/etc/masterha/app1.conf --dead_master_host=Master --dead_master_port=3306 --new_master_host=Slave1 --new_master_port=3306 --ignore_last_failover
# ------------省略其他输出------------
# 期间会问你两个问题，直接输入 yes 即可
# 问题 1:
# Master Master(172.18.14.70:3306) is dead. Proceed? (yes/NO): yes
# 问题 2:
# Starting master switch from Master(172.18.14.70:3306) to Slave1(172.18.14.71:3306)? (yes/NO): yes

# ------------省略其他输出------------
----- Failover Report -----

app1: MySQL Master failover Master(172.18.14.70:3306) to Slave1(172.18.14.71:3306) succeeded

Master Master(172.18.14.70:3306) is down!

Check MHA Manager logs at MHA for details.

Started manual(interactive) failover.
Invalidated master IP address on Master(172.18.14.70:3306)
Selected Slave1(172.18.14.71:3306) as a new master.
Slave1(172.18.14.71:3306): OK: Applying all logs succeeded.
Slave1(172.18.14.71:3306): OK: Activated master IP address.
Slave2(172.18.14.72:3306): OK: Slave started, replicating from Slave1(172.18.14.71:3306)
Slave1(172.18.14.71:3306): Resetting slave info succeeded.
Master failover to Slave1(172.18.14.71:3306) completed successfully.
```

2.11.3. 查看状态

```
Shell> ifconfig
ens3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 172.18.14.71  netmask 255.255.255.0  broadcast 172.18.14.255
        inet6 fe80::5054:f0ff:fe34:66c1  prefixlen 64  scopeid 0x20<link>
        ether 52:54:f0:34:66:c1  txqueuelen 1000  (Ethernet)
        RX packets 249053  bytes 25385300 (24.2 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 111470  bytes 118934787 (113.4 MiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

# VIP 漂移到了Slave1
ens3:88: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 172.18.14.88  netmask 255.255.255.0  broadcast 172.18.14.255
        ether 52:54:f0:34:66:c1  txqueuelen 1000  (Ethernet)

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 0  (Local Loopback)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0


mysql> show slave status\G
*************************** 1. row ***************************
               Slave_IO_State: Waiting for master to send event
                  Master_Host: 172.18.14.71
                  Master_User: rpl
                  Master_Port: 3306
                Connect_Retry: 60
              Master_Log_File: bin.000002
          Read_Master_Log_Pos: 1111
               Relay_Log_File: relay.000004
                Relay_Log_Pos: 436
        Relay_Master_Log_File: bin.000002
             Slave_IO_Running: Yes
            Slave_SQL_Running: Yes
              Replicate_Do_DB:
          Replicate_Ignore_DB:
-- ------------省略其他输出------------
```

上述输出和自动切换类似，到这里MHA手动Failover完成

2.12. MHA总结

2.12.1. 部署操作过程

1. 配置好MySQL 复制关系 ，至少三个节点，且确保 `rpl用户` 传递到Slave上；
2. 在 `MHA Manager` 上安装 `Manager` 组件；
3. 在 所有节点 上安装 `Node` 组件；
4. 配置好 `/etc/masterha/app1.conf` ；
5. 启动 `masterha_manager` （ *supervisord托管* ）；
6. 确认 `masterha_check_status` 状态；

2.12.2. Old Master恢复

2.12.2.1. 日志处理

当切换完成后， 假如Old Master修复完成 ，这时需要对比一下数据是否一致；如果主从不一致，即Old Master 宕机时刻的binlog 没有传到Slave， 且MHA也无法获取到这部分binlog ，需要通过 `Flashback` 工具将这部分数据 切除 ；

1. `Old Master` 的binlog信息可以通过 `show master status\G` 看到，或者通过 `mysqlbinlog` 进行查看；
2. `New Master` 上查看执行到的 `Old Master` 复制过来的binlog信息可以通过 `show global variables like "%gtid%";` 来进行查看（mysql会保留之前执行过的GTID信息）；
    - 在没有使用MHA，或者使用MHA手动Failover的时候，可以通过在Slave上执行 `show slave status\G` 通过 `Exec_Master_Log_Pos` 观察到执行到的位置（ *等待回放完毕* ）
    - 而使用MHA切换后， `show slave status\G` 的信息会被MHA给reset掉（除非自己给MHA打补丁，将信息记录下来）；

通过上述方法，大致可以判断是否需要Flashback或者哪些数据需要Flashback；

2.12.2.2. 角色处理

当Old Master恢复后且日志处理完成，建议将该服务器 作为New Master的Slave节点 ，通过 `CHANGE MASTER（MASTER_AUTO_POSITION=1）` 可以很方便的建立主从关系。

```
--
-- Old Master 端
--
mysql> change master to master_host='172.18.14.71', master_port=3306, master_auto_position=1, master_user='rpl', master_password='123';

mysql> start slave;
```

如果中间产生问题，和之前处理GTID复制出错一样， 跳过执行的部分 即可：

```
mysql> stop slave;
mysql> reset master;
mysql> set @@global.gtid_purged='2d408b04-0154-11e6-83b6-5254f035dabc:1-4389';
mysql> change master to master_host='172.18.14.71', master_port=3306, master_auto_position=1, master_user='rpl', master_password='123';
mysql> start slave;
```

```
Shell>  mysqlrplshow --master=root:123@172.18.14.71:3306 --discover-slaves-login=root:123 --verbose
WARNING: Using a password on the command line interface can be insecure.
# master on 172.18.14.71: ... connected.
# Finding slaves for master: 172.18.14.71:3306

# Replication Topology Graph
172.18.14.71:3306 (MASTER)
   |
   +--- 172.18.14.70:3306 [IO: Yes, SQL: Yes] - (SLAVE)
   |
   +--- 172.18.14.72:3306 [IO: Yes, SQL: Yes] - (SLAVE)
```