# MHA

## 简介

MHA（Master High Availability）目前在MySQL高可用方面是一个相对成熟的解决方案，它由日本DeNA公司youshimaton（现就职于Facebook公司）开发，是一套优秀的作为MySQL高可用性环境下故障切换和主从提升的高可用软件。在MySQL故障切换过程中，MHA能做到在0~30秒之内自动完成数据库的故障切换操作，并且在进行故障切换的过程中，MHA能在最大程度上保证数据的一致性，以达到真正意义上的高可用。

**该软件由两部分组成：MHA Manager（管理节点）和MHA Node（数据节点）**。MHA Manager可以单独部署在一台独立的机器上管理多个master-slave集群，也可以部署在一台slave节点上。MHA Node运行在每台MySQL服务器上，MHA Manager会定时探测集群中的master节点，当master出现故障时，它可以自动将最新数据的slave提升为新的master，然后将所有其他的slave重新指向新的master。整个故障转移过程对应用程序完全透明。

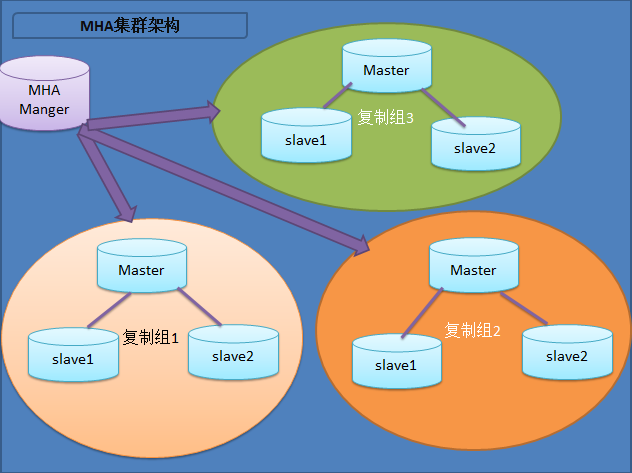
在MHA自动故障切换过程中，MHA试图从宕机的主服务器上保存二进制日志，最大程度的保证数据的不丢失，但这并不总是可行的。例如，如果主服务器硬件故障或无法通过ssh访问，MHA没法保存二进制日志，只进行故障转移而丢失了最新的数据。使用MySQL 5.5的半同步复制，可以大大降低数据丢失的风险。MHA可以与半同步复制结合起来。如果只有一个slave已经收到了最新的二进制日志，MHA可以将最新的二进制日志应用于其他所有的slave服务器上，因此可以保证所有节点的数据一致性。

目前MHA主要支持一主多从的架构，**要搭建MHA,要求一个复制集群中必须最少有三台数据库服务器，**一主二从，即一台充当master，一台充当备用master，另外一台充当从库，因为至少需要三台服务器，出于机器成本的考虑，淘宝也在该基础上进行了改造，目前淘宝TMHA已经支持一主一从。另外对于想快速搭建的可以参考：[MHA快速搭建](http://www.cnblogs.com/gomysql/p/6547797.html)

我们自己使用其实也可以使用1主1从，但是master主机宕机后无法切换，以及无法补全binlog。master的mysqld进程crash后，还是可以切换成功，以及补全binlog的。

官方介绍：<https://code.google.com/p/mysql-master-ha/>

图01展示了如何通过MHA Manager管理多组主从复制。可以将MHA工作原理总结为如下：



                                 （ 图01 ）

**（1）从宕机崩溃的master保存二进制日志事件（binlog events）;**

**（2）识别含有最新更新的slave；**

**（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）；**

**（5）提升一个slave为新的master；**

**（6）使其他的slave连接新的master进行复制；**

MHA软件由两部分组成，Manager工具包和Node工具包，具体的说明如下。

Manager工具包主要包括以下几个工具：

[复制代码](javascript:void(0);)

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 控制故障转移（自动或者手动）

masterha\_conf\_host 添加或删除配置的server信息

[复制代码](javascript:void(0);)

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已不再使用这个工具）

purge\_relay\_logs 清除中继日志（不会阻塞SQL线程）

**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 5.5的半同步复制。关于半同步复制原理各位自己进行查阅。（不是必须）**

## 搭建

### 1.部署MHA

接下来部署MHA，具体的搭建环境如下（所有操作系统均为centos 6.2 64bit，不是必须，server03和server04是server02的从，复制环境搭建后面会简单演示，但是相关的安全复制不会详细说明，需要的童鞋请参考前面的文章，[MySQL Replication需要注意的问题](http://www.cnblogs.com/gomysql/p/3662492.html)）：

[复制代码](javascript:void(0);)

角色 ip地址 主机名 server\_id 类型

Monitor host 192.168.0.20 server01 - 监控复制组

Master 192.168.0.50 server02 1 写入

Candicate master 192.168.0.60 server03 2 读

Slave 192.168.0.70 server04 3 读

[复制代码](javascript:void(0);)

**其中master对外提供写服务，备选master（实际的slave，主机名server03）提供读服务，slave也提供相关的读服务，一旦master宕机，将会把备选master提升为新的master，slave指向新的master**

（1）在所有节点安装MHA node所需的perl模块（DBD:mysql），安装脚本如下：

[复制代码](javascript:void(0);)

[root@192.168.0.50 ~]# cat install.sh

#!/bin/bash

wget http://xrl.us/cpanm --no-check-certificate

mv cpanm /usr/bin

chmod 755 /usr/bin/cpanm

cat > /root/list << EOF

install DBD::mysql

EOF

for package in `cat /root/list`

do

cpanm $package

done

[root@192.168.0.50 ~]#

[复制代码](javascript:void(0);)

如果有安装epel源，也可以使用yum安装

yum install perl-DBD-MySQL -y

（2）在所有的节点安装mha node：

wget http://mysql-master-ha.googlecode.com/files/mha4mysql-node-0.53.tar.gz

tar xf mha4mysql-node-0.53.tar.gz

cd mha4mysql-node-0.53

perl Makefile.PL

make && make install

安装完成后会在/usr/local/bin目录下生成以下脚本文件：

[复制代码](javascript:void(0);)

[root@192.168.0.50 bin]# pwd

/usr/local/bin

[root@192.168.0.50 bin]# ll

total 40

-r-xr-xr-x 1 root root 15498 Apr 20 10:05 apply\_diff\_relay\_logs

-r-xr-xr-x 1 root root 4807 Apr 20 10:05 filter\_mysqlbinlog

-r-xr-xr-x 1 root root 7401 Apr 20 10:05 purge\_relay\_logs

-r-xr-xr-x 1 root root 7263 Apr 20 10:05 save\_binary\_logs

[root@192.168.0.50 bin]#

[复制代码](javascript:void(0);)

关于上面脚本的功能，上面已经介绍过了，这里不再重复了。

### 2.安装MHA Manager

MHA Manager中主要包括了几个管理员的命令行工具，例如master\_manger，master\_master\_switch等。MHA Manger也依赖于perl模块，具体如下：

（1）安装MHA Node软件包之前需要安装依赖。我这里使用yum完成，没有epel源的可以使用上面提到的脚本（epel源安装也简单）。注意：在MHA Manager的主机也是需要安装MHA Node。

rpm -ivh http://dl.fedoraproject.org/pub/epel/6/x86\_64/epel-release-6-8.noarch.rpm

yum install perl-DBD-MySQL -y

安装MHA Node软件包，和上面的方法一样，如下：

wget http://mysql-master-ha.googlecode.com/files/mha4mysql-node-0.53.tar.gz

tar xf mha4mysql-node-0.53.tar.gz

cd mha4mysql-node-0.53

perl Makefile.PL

make && make install

（2）安装MHA Manager。首先安装MHA Manger依赖的perl模块（我这里使用yum安装）：

yum install perl-DBD-MySQL perl-Config-Tiny perl-Log-Dispatch perl-Parallel-ForkManager perl-Time-HiRes -y

安装MHA Manager软件包：

wget http://mysql-master-ha.googlecode.com/files/mha4mysql-manager-0.53.tar.gz

tar xf mha4mysql-manager-0.53.tar.gz

cd mha4mysql-manager-0.53

perl Makefile.PL

make && make install

安装完成后会在/usr/local/bin目录下面生成以下脚本文件，前面已经说过这些脚本的作用，这里不再重复

[复制代码](javascript:void(0);)

[root@192.168.0.20 bin]# pwd

/usr/local/bin

[root@192.168.0.20 bin]# ll

total 76

-r-xr-xr-x 1 root root 15498 Apr 20 10:58 apply\_diff\_relay\_logs

-r-xr-xr-x 1 root root 4807 Apr 20 10:58 filter\_mysqlbinlog

-r-xr-xr-x 1 root root 1995 Apr 20 11:33 masterha\_check\_repl

-r-xr-xr-x 1 root root 1779 Apr 20 11:33 masterha\_check\_ssh

-r-xr-xr-x 1 root root 1865 Apr 20 11:33 masterha\_check\_status

-r-xr-xr-x 1 root root 3201 Apr 20 11:33 masterha\_conf\_host

-r-xr-xr-x 1 root root 2517 Apr 20 11:33 masterha\_manager

-r-xr-xr-x 1 root root 2165 Apr 20 11:33 masterha\_master\_monitor

-r-xr-xr-x 1 root root 2373 Apr 20 11:33 masterha\_master\_switch

-r-xr-xr-x 1 root root 3749 Apr 20 11:33 masterha\_secondary\_check

-r-xr-xr-x 1 root root 1739 Apr 20 11:33 masterha\_stop

-r-xr-xr-x 1 root root 7401 Apr 20 10:58 purge\_relay\_logs

-r-xr-xr-x 1 root root 7263 Apr 20 10:58 save\_binary\_logs

[root@192.168.0.20 bin]#

[复制代码](javascript:void(0);)

**复制相关脚本到/usr/local/bin目录(软件包解压缩后就有了，不是必须，因为这些脚本不完整，需要自己修改，这是软件开发着留给我们自己发挥的,如果开启下面的任何一个脚本对应的参数，而对应这里的脚本又没有修改，则会抛错，自己被坑的很惨)**

[复制代码](javascript:void(0);)

[root@192.168.0.20 scripts]# pwd

/root/mha4mysql-manager-0.53/samples/scripts

[root@192.168.0.20 scripts]# ll

total 32

-rwxr-xr-x 1 root root 3443 Jan 8 2012 master\_ip\_failover #自动切换时vip管理的脚本，不是必须，如果我们使用keepalived的，我们可以自己编写脚本完成对vip的管理，比如监控mysql，如果mysql异常，我们停止keepalived就行，这样vip就会自动漂移

-rwxr-xr-x 1 root root 9186 Jan 8 2012 master\_ip\_online\_change #在线切换时vip的管理，不是必须，同样可以可以自行编写简单的shell完成

-rwxr-xr-x 1 root root 11867 Jan 8 2012 power\_manager #故障发生后关闭主机的脚本，不是必须

-rwxr-xr-x 1 root root 1360 Jan 8 2012 send\_report #因故障切换后发送报警的脚本，不是必须，可自行编写简单的shell完成。

[root@192.168.0.20 scripts]# cp \* /usr/local/bin/

[root@192.168.0.20 scripts]#

[复制代码](javascript:void(0);)

### 3.配置SSH登录无密码验证

**（使用key登录，工作中常用）我的测试环境已经是使用key登录，服务器之间无需密码验证的。关于配置使用key登录，我想我不再重复。但是有一点需要注意：不能禁止 password 登陆，否则会出现错误**

### 4.搭建主从复制环境

**注意：binlog-do-db 和 replicate-ignore-db 设置必须相同。 MHA 在启动时候会检测过滤规则，如果过滤规则不同，MHA 不启动监控和故障转移。**

（1）在server02上执行备份（192.168.0.50）

[root@192.168.0.50 ~]# mysqldump --master-data=2 --single-transaction -R --triggers -A > all.sql

其中--master-data=2代表备份时刻记录master的Binlog位置和Position，--single-transaction意思是获取一致性快照，-R意思是备份存储过程和函数，--triggres的意思是备份触发器，-A代表备份所有的库。更多信息请自行mysqldump --help查看。

（2）在server02上创建复制用户：

[复制代码](javascript:void(0);)

mysql> grant replication slave on \*.\* to 'repl'@'192.168.0.%' identified by '123456';

Query OK, **0** rows affected (**0.00** sec)

mysql> flush privileges;

Query OK, **0** rows affected (**0.00** sec)

mysql>

[复制代码](javascript:void(0);)

（3）查看主库备份时的binlog名称和位置，MASTER\_LOG\_FILE和MASTER\_LOG\_POS：

[root@192.168.0.50 ~]# head -n 30 all.sql | grep 'CHANGE MASTER TO'

-- CHANGE MASTER TO MASTER\_LOG\_FILE='mysql-bin.000010', MASTER\_LOG\_POS=112;

[root@192.168.0.50 ~]#

（4）把备份复制到server03和server04，也就是192.168.0.60和192.168.0.70

scp all.sql server03:/data/

scp all.sql server04:/data/

（5）导入备份到server03，执行复制相关命令

mysql < /data/all.sql

[复制代码](javascript:void(0);)

mysql> CHANGE MASTER TO MASTER\_HOST='192.168.0.50',MASTER\_USER='repl', MASTER\_PASSWORD='123456',MASTER\_LOG\_FILE='mysql-bin.000010',MASTER\_LOG\_POS=**112**;

Query OK, **0** rows affected (**0.02** sec)

mysql> start slave;

Query OK, **0** rows affected (**0.01** sec)

mysql>

[复制代码](javascript:void(0);)

查看复制状态（可以看见复制成功）：

[root@192.168.0.60 ~]# mysql -e 'show slave status\G' | egrep 'Slave\_IO|Slave\_SQL'

Slave\_IO\_State: Waiting for master to send event

Slave\_IO\_Running: Yes

Slave\_SQL\_Running: Yes

[root@192.168.0.60 ~]#

（6）在server04（192.168.0.70）上搭建复制环境，操作和上面一样。

mysql < /data/all.sql

[复制代码](javascript:void(0);)

mysql> CHANGE MASTER TO MASTER\_HOST='192.168.0.50',MASTER\_USER='repl', MASTER\_PASSWORD='123456',MASTER\_LOG\_FILE='mysql-bin.000010',MASTER\_LOG\_POS=**112**;

Query OK, **0** rows affected (**0.07** sec)

mysql> start slave;

Query OK, **0** rows affected (**0.00** sec)

mysql>

[复制代码](javascript:void(0);)

查看复制状态：

[root@192.168.0.70 ~]# mysql -e 'show slave status\G' | egrep 'Slave\_IO|Slave\_SQL'

Slave\_IO\_State: Waiting for master to send event

Slave\_IO\_Running: Yes

Slave\_SQL\_Running: Yes

[root@192.168.0.70 ~]#

（7）两台slave服务器设置read\_only（从库对外提供读服务，只所以没有写进配置文件，是因为**随时slave会提升为master**）

[root@192.168.0.60 ~]# mysql -e 'set global read\_only=1'

[root@192.168.0.60 ~]#

[root@192.168.0.70 ~]# mysql -e 'set global read\_only=1'

[root@192.168.0.70 ~]#

（8）创建监控用户（在master上执行，也就是192.168.0.50）：

[复制代码](javascript:void(0);)

mysql> grant all privileges on \*.\* to 'root'@'192.168.0.%' identified by '123456';

Query OK, **0** rows affected (**0.00** sec)

mysql> flush privileges;

Query OK, **0** rows affected (**0.01** sec)

mysql>

[复制代码](javascript:void(0);)

到这里整个集群环境已经搭建完毕，剩下的就是配置MHA软件了。

### 5.配置MHA

（1）创建MHA的工作目录，并且创建相关配置文件（在软件包解压后的目录里面有样例配置文件）。

[root@192.168.0.20 ~]# mkdir -p /etc/masterha

[root@192.168.0.20 ~]# cp mha4mysql-manager-0.53/samples/conf/app1.cnf /etc/masterha/

[root@192.168.0.20 ~]#

修改app1.cnf配置文件，修改后的文件内容如下（**注意，配置文件中的注释需要去掉，我这里是为了解释清楚**）：

[复制代码](javascript:void(0);)

[root@192.168.0.20 ~]# cat /etc/masterha/app1.cnf

[server default]

manager\_workdir=/var/log/masterha/app1.log //设置manager的工作目录

manager\_log=/var/log/masterha/app1/manager.log //设置manager的日志

master\_binlog\_dir=/data/mysql //设置master 保存binlog的位置，以便MHA可以找到master的日志，我这里的也就是mysql的数据目录

master\_ip\_failover\_script= /usr/local/bin/master\_ip\_failover //设置自动failover时候的切换脚本

master\_ip\_online\_change\_script= /usr/local/bin/master\_ip\_online\_change //设置手动切换时候的切换脚本

password=123456 //设置mysql中root用户的密码，这个密码是前文中创建监控用户的那个密码

user=root 设置监控用户root

ping\_interval=1 //设置监控主库，发送ping包的时间间隔，默认是3秒，尝试三次没有回应的时候自动进行railover

remote\_workdir=/tmp //设置远端mysql在发生切换时binlog的保存位置

repl\_password=123456 //设置复制用户的密码

repl\_user=repl //设置复制环境中的复制用户名

report\_script=/usr/local/send\_report //设置发生切换后发送的报警的脚本

secondary\_check\_script= /usr/local/bin/masterha\_secondary\_check -s server03 -s server02   
shutdown\_script="" //设置故障发生后关闭故障主机脚本（该脚本的主要作用是关闭主机放在发生脑裂,这里没有使用）

ssh\_user=root //设置ssh的登录用户名

[server1]

hostname=192.168.0.50

port=3306

[server2]

hostname=192.168.0.60

port=3306

candidate\_master=1 //设置为候选master，如果设置该参数以后，发生主从切换以后将会将此从库提升为主库，即使这个主库不是集群中事件最新的slave

check\_repl\_delay=0 //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

[server3]

hostname=192.168.0.70

port=3306

[root@192.168.0.20 ~]#

[复制代码](javascript:void(0);)

（2）设置relay log的清除方式（在每个slave节点上）：

[root@192.168.0.60 ~]# mysql -e 'set global relay\_log\_purge=0'

[root@192.168.0.70 ~]# mysql -e 'set global relay\_log\_purge=0'

**注意：**

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

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

pure\_relay\_logs脚本参数如下所示：

[复制代码](javascript:void(0);)

--user mysql 用户名

--password mysql 密码

--port 端口号

--workdir 指定创建relay log的硬链接的位置，默认是/var/tmp，由于系统不同分区创建硬链接文件会失败，故需要执行硬链接具体位置，成功执行脚本后，硬链接的中继日志文件被删除

--disable\_relay\_log\_purge 默认情况下，如果relay\_log\_purge=1，脚本会什么都不清理，自动退出，通过设定这个参数，当relay\_log\_purge=1的情况下会将relay\_log\_purge设置为0。清理relay log之后，最后将参数设置为OFF。

[复制代码](javascript:void(0);)

（3）设置定期清理relay脚本（两台slave服务器）

[复制代码](javascript:void(0);)

[root@192.168.0.60 ~]# cat purge\_relay\_log.sh

#!/bin/bash

user=root

passwd=123456

port=3306

log\_dir='/data/masterha/log'

work\_dir='/data'

purge='/usr/local/bin/purge\_relay\_logs'

if [ ! -d $log\_dir ]

then

mkdir $log\_dir -p

fi

$purge --user=$user --password=$passwd --disable\_relay\_log\_purge --port=$port --workdir=$work\_dir >> $log\_dir/purge\_relay\_logs.log 2>&1

[root@192.168.0.60 ~]#

[复制代码](javascript:void(0);)

添加到crontab定期执行

[root@192.168.0.60 ~]# crontab -l

0 4 \* \* \* /bin/bash /root/purge\_relay\_log.sh

[root@192.168.0.60 ~]#

purge\_relay\_logs脚本删除中继日志不会阻塞SQL线程。下面我们手动执行看看什么情况。

[复制代码](javascript:void(0);)

[root@192.168.0.60 ~]# purge\_relay\_logs --user=root --password=123456 --port=3306 -disable\_relay\_log\_purge --workdir=/data/

2014-04-20 15:47:24: purge\_relay\_logs script started.

Found relay\_log.info: /data/mysql/relay-log.info

Removing hard linked relay log files server03-relay-bin\* under /data/.. done.

Current relay log file: /data/mysql/server03-relay-bin.000002

Archiving unused relay log files (up to /data/mysql/server03-relay-bin.000001) ...

Creating hard link for /data/mysql/server03-relay-bin.000001 under /data//server03-relay-bin.000001 .. ok.

Creating hard links for unused relay log files completed.

Executing SET GLOBAL relay\_log\_purge=1; FLUSH LOGS; sleeping a few seconds so that SQL thread can delete older relay log files (if it keeps up); SET GLOBAL relay\_log\_purge=0; .. ok.

Removing hard linked relay log files server03-relay-bin\* under /data/.. done.

2014-04-20 15:47:27: All relay log purging operations succeeded.

[root@192.168.0.60 ~]#

[复制代码](javascript:void(0);)

### 6.检查SSH配置

检查MHA Manger到所有MHA Node的SSH连接状态：

[复制代码](javascript:void(0);)

[root@192.168.0.20 ~]# masterha\_check\_ssh --conf=/etc/masterha/app1.cnf

Sun Apr 20 17:17:39 2014 - [warning] Global configuration file /etc/masterha\_default.cnf not found. Skipping.

Sun Apr 20 17:17:39 2014 - [info] Reading application default configurations from /etc/masterha/app1.cnf..

Sun Apr 20 17:17:39 2014 - [info] Reading server configurations from /etc/masterha/app1.cnf..

Sun Apr 20 17:17:39 2014 - [info] Starting SSH connection tests..

Sun Apr 20 17:17:40 2014 - [debug]

Sun Apr 20 17:17:39 2014 - [debug] Connecting via SSH from root@192.168.0.50(192.168.0.50:22) to root@192.168.0.60(192.168.0.60:22)..

Sun Apr 20 17:17:39 2014 - [debug] ok.

Sun Apr 20 17:17:39 2014 - [debug] Connecting via SSH from root@192.168.0.50(192.168.0.50:22) to root@192.168.0.70(192.168.0.70:22)..

Sun Apr 20 17:17:39 2014 - [debug] ok.

Sun Apr 20 17:17:40 2014 - [debug]

Sun Apr 20 17:17:40 2014 - [debug] Connecting via SSH from root@192.168.0.60(192.168.0.60:22) to root@192.168.0.50(192.168.0.50:22)..

Sun Apr 20 17:17:40 2014 - [debug] ok.

Sun Apr 20 17:17:40 2014 - [debug] Connecting via SSH from root@192.168.0.60(192.168.0.60:22) to root@192.168.0.70(192.168.0.70:22)..

Sun Apr 20 17:17:40 2014 - [debug] ok.

Sun Apr 20 17:17:41 2014 - [debug]

Sun Apr 20 17:17:40 2014 - [debug] Connecting via SSH from root@192.168.0.70(192.168.0.70:22) to root@192.168.0.50(192.168.0.50:22)..

Sun Apr 20 17:17:40 2014 - [debug] ok.

Sun Apr 20 17:17:40 2014 - [debug] Connecting via SSH from root@192.168.0.70(192.168.0.70:22) to root@192.168.0.60(192.168.0.60:22)..

Sun Apr 20 17:17:41 2014 - [debug] ok.

Sun Apr 20 17:17:41 2014 - [info] All SSH connection tests passed successfully.

[复制代码](javascript:void(0);)

可以看见各个节点ssh验证都是ok的。

### 7.检查整个复制环境状况

通过masterha\_check\_repl脚本查看整个集群的状态

[复制代码](javascript:void(0);)

[root@192.168.0.20 ~]# masterha\_check\_repl --conf=/etc/masterha/app1.cnf

Sun Apr 20 18:36:55 2014 - [info] Checking replication health on 192.168.0.60..

Sun Apr 20 18:36:55 2014 - [info] ok.

Sun Apr 20 18:36:55 2014 - [info] Checking replication health on 192.168.0.70..

Sun Apr 20 18:36:55 2014 - [info] ok.

Sun Apr 20 18:36:55 2014 - [info] Checking master\_ip\_failover\_script status:

Sun Apr 20 18:36:55 2014 - [info] /usr/local/bin/master\_ip\_failover --command=status --ssh\_user=root --orig\_master\_host=192.168.0.50 --orig\_master\_ip=192.168.0.50 --orig\_master\_port=3306

Bareword "FIXME\_xxx" not allowed while "strict subs" in use at /usr/local/bin/master\_ip\_failover line 88.

Execution of /usr/local/bin/master\_ip\_failover aborted due to compilation errors.

Sun Apr 20 18:36:55 2014 - [error][/usr/local/share/perl5/MHA/MasterMonitor.pm, ln214] Failed to get master\_ip\_failover\_script status with return code 255:0.

Sun Apr 20 18:36:55 2014 - [error][/usr/local/share/perl5/MHA/MasterMonitor.pm, ln383] Error happend on checking configurations. at /usr/local/bin/masterha\_check\_repl line 48

Sun Apr 20 18:36:55 2014 - [error][/usr/local/share/perl5/MHA/MasterMonitor.pm, ln478] Error happened on monitoring servers.

Sun Apr 20 18:36:55 2014 - [info] Got exit code 1 (Not master dead).

MySQL Replication Health is NOT OK!

[复制代码](javascript:void(0);)

**发现最后的结论说我的复制不是ok的。但是上面的信息明明说是正常的，自己也进数据库查看了。这里一直踩坑。一直纠结，后来无意中发现火丁笔记的博客，这才知道了原因，原来Failover两种方式：一种是虚拟IP地址，一种是全局配置文件。MHA并没有限定使用哪一种方式，而是让用户自己选择，虚拟IP地址的方式会牵扯到其它的软件,比如keepalive软件，而且还要修改脚本master\_ip\_failover。(最后修改脚本后才没有这个报错，自己不懂perl也是折腾的半死，去年买了块表)**

**如果发现如下错误：**

Can't exec "mysqlbinlog": No such file or directory at /usr/local/share/perl5/MHA/BinlogManager.pm line 99.

mysqlbinlog version not found!

Testing mysql connection and privileges..sh: mysql: command not found

解决方法如下，添加软连接（所有节点）

ln -s /usr/local/mysql/bin/mysqlbinlog /usr/local/bin/mysqlbinlog

ln -s /usr/local/mysql/bin/mysql /usr/local/bin/mysql

所以先暂时注释master\_ip\_failover\_script= /usr/local/bin/master\_ip\_failover这个选项。后面引入keepalived后和修改该脚本以后再开启该选项。

[root@192.168.0.20 ~]# grep master\_ip\_failover /etc/masterha/app1.cnf

#master\_ip\_failover\_script= /usr/local/bin/master\_ip\_failover

[root@192.168.0.20 ~]#

再次进行状态查看：

[复制代码](javascript:void(0);)

Sun Apr 20 18:46:08 2014 - [info] Checking replication health on 192.168.0.60..

Sun Apr 20 18:46:08 2014 - [info] ok.

Sun Apr 20 18:46:08 2014 - [info] Checking replication health on 192.168.0.70..

Sun Apr 20 18:46:08 2014 - [info] ok.

Sun Apr 20 18:46:08 2014 - [warning] master\_ip\_failover\_script is not defined.

Sun Apr 20 18:46:08 2014 - [warning] shutdown\_script is not defined.

Sun Apr 20 18:46:08 2014 - [info] Got exit code 0 (Not master dead).

MySQL Replication Health is OK.

[复制代码](javascript:void(0);)

已经没有明显报错，只有两个警告而已，复制也显示正常了。

### 8.检查MHA Manager的状态

通过master\_check\_status脚本查看Manager的状态：

[root@192.168.0.20 ~]# masterha\_check\_status --conf=/etc/masterha/app1.cnf

app1 is stopped(2:NOT\_RUNNING).

[root@192.168.0.20 ~]#

注意：如果正常，会显示"PING\_OK"，否则会显示"NOT\_RUNNING"，这代表MHA监控没有开启。

### 9.开启MHA Manager监控

[root@192.168.0.20 ~]# nohup masterha\_manager --conf=/etc/masterha/app1.cnf --remove\_dead\_master\_conf --ignore\_last\_failover < /dev/null > /var/log/masterha/app1/manager.log 2>&1 &

[1] 30867

[root@192.168.0.20 ~]#

**启动参数介绍：**

--remove\_dead\_master\_conf      该参数代表当发生主从切换后，老的主库的ip将会从配置文件中移除。

--manger\_log                            日志存放位置

--ignore\_last\_failover                 在缺省情况下，如果MHA检测到连续发生宕机，且两次宕机间隔不足8小时的话，则不会进行Failover，之所以这样限制是为了避免ping-pong效应。**该参数代表忽略上次MHA触发切换产生的文件**，默认情况下，MHA发生切换后会在日志目录，也就是上面我设置的/data产生app1.failover.complete文件，下次再次切换的时候如果发现该目录下存在该文件将不允许触发切换，除非在第一次切换后收到删除该文件，为了方便，这里设置为--ignore\_last\_failover。

查看MHA Manager监控是否正常：

[root@192.168.0.20 ~]# masterha\_check\_status --conf=/etc/masterha/app1.cnf

app1 (pid:20386) is running(0:PING\_OK), master:192.168.0.50

[root@192.168.0.20 ~]#

可以看见已经在监控了，而且master的主机为192.168.0.50

### 10.查看启动日志

[复制代码](javascript:void(0);)

[root@192.168.0.20 ~]# tail -n20 /var/log/masterha/app1/manager.log

Sun Apr 20 19:12:01 2014 - [info] Connecting to root@192.168.0.70(192.168.0.70:22)..

Checking slave recovery environment settings..

Opening /data/mysql/relay-log.info ... ok.

Relay log found at /data/mysql, up to server04-relay-bin.000002

Temporary relay log file is /data/mysql/server04-relay-bin.000002

Testing mysql connection and privileges.. done.

Testing mysqlbinlog output.. done.

Cleaning up test file(s).. done.

Sun Apr 20 19:12:01 2014 - [info] Slaves settings check done.

Sun Apr 20 19:12:01 2014 - [info]

192.168.0.50 (current master)

+--192.168.0.60

+--192.168.0.70

Sun Apr 20 19:12:01 2014 - [warning] master\_ip\_failover\_script is not defined.

Sun Apr 20 19:12:01 2014 - [warning] shutdown\_script is not defined.

Sun Apr 20 19:12:01 2014 - [info] Set master ping interval 1 seconds.

Sun Apr 20 19:12:01 2014 - [info] Set secondary check script: /usr/local/bin/masterha\_secondary\_check -s server03 -s server02 --user=root --master\_host=server02 --master\_ip=192.168.0.50 --master\_port=3306

Sun Apr 20 19:12:01 2014 - [info] Starting ping health check on 192.168.0.50(192.168.0.50:3306)..

Sun Apr 20 19:12:01 2014 - [info] Ping(SELECT) succeeded, waiting until MySQL doesn't respond..

[root@192.168.0.20 ~]#

[复制代码](javascript:void(0);)

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

### **11.关闭MHA Manage监控**

关闭很简单，使用masterha\_stop命令完成。

[root@192.168.0.20 ~]# masterha\_stop --conf=/etc/masterha/app1.cnf

Stopped app1 successfully.

[1]+ Exit 1 nohup masterha\_manager --conf=/etc/masterha/app1.cnf --remove\_dead\_master\_conf --ignore\_last\_failover --manager\_log=/data/mamanager.log

[root@192.168.0.20 ~]#

### 12.配置VIP

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

1.keepalived方式管理虚拟ip，keepalived配置方法如下：

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

[root@192.168.0.50 ~]# wget http://www.keepalived.org/software/keepalived-1.2.12.tar.gz

[复制代码](javascript:void(0);)

tar xf keepalived-1.2.12.tar.gz

cd keepalived-1.2.12

./configure --prefix=/usr/local/keepalived

make && make install

cp /usr/local/keepalived/etc/rc.d/init.d/keepalived /etc/init.d/

cp /usr/local/keepalived/etc/sysconfig/keepalived /etc/sysconfig/

mkdir /etc/keepalived

cp /usr/local/keepalived/etc/keepalived/keepalived.conf /etc/keepalived/

cp /usr/local/keepalived/sbin/keepalived /usr/sbin/

[复制代码](javascript:void(0);)

（2）配置keepalived的配置文件，在master上配置（192.168.0.50）

[复制代码](javascript:void(0);)

[root@192.168.0.50 ~]# cat /etc/keepalived/keepalived.conf

! Configuration File for keepalived

global\_defs {

notification\_email {

saltstack@163.com

}

notification\_email\_from dba@dbserver.com

smtp\_server 127.0.0.1

smtp\_connect\_timeout 30

router\_id MySQL-HA

}

vrrp\_instance VI\_1 {

state BACKUP

interface eth1

virtual\_router\_id 51

priority 150

advert\_int 1

nopreempt

authentication {

auth\_type PASS

auth\_pass 1111

}

virtual\_ipaddress {

192.168.0.88

}

}

[root@192.168.0.50 ~]#

[复制代码](javascript:void(0);)

其中router\_id MySQL HA表示设定keepalived组的名称，将192.168.0.88这个虚拟ip绑定到该主机的eth1网卡上，并且设置了状态为backup模式，将keepalived的模式设置为非抢占模式（nopreempt），priority 150表示设置的优先级为150。下面的配置略有不同，但是都是一个意思。  
在候选master上配置（192.168.0.60）

[复制代码](javascript:void(0);)

[root@192.168.0.60 ~]# cat /etc/keepalived/keepalived.conf

! Configuration File for keepalived

global\_defs {

notification\_email {

saltstack@163.com

}

notification\_email\_from dba@dbserver.com

smtp\_server 127.0.0.1

smtp\_connect\_timeout 30

router\_id MySQL-HA

}

vrrp\_instance VI\_1 {

state BACKUP

interface eth1

virtual\_router\_id 51

priority 120

advert\_int 1

nopreempt

authentication {

auth\_type PASS

auth\_pass 1111

}

virtual\_ipaddress {

192.168.0.88

}

}

[root@192.168.0.60 ~]#

[复制代码](javascript:void(0);)

（3）启动keepalived服务，在master上启动并查看日志

[复制代码](javascript:void(0);)

[root@192.168.0.50 ~]# /etc/init.d/keepalived start

Starting keepalived: [ OK ]

[root@192.168.0.50 ~]# tail -f /var/log/messages

Apr 20 20:22:16 192 Keepalived\_healthcheckers[15334]: Opening file '/etc/keepalived/keepalived.conf'.

Apr 20 20:22:16 192 Keepalived\_healthcheckers[15334]: Configuration is using : 7231 Bytes

Apr 20 20:22:16 192 kernel: IPVS: Connection hash table configured (size=4096, memory=64Kbytes)

Apr 20 20:22:16 192 kernel: IPVS: ipvs loaded.

Apr 20 20:22:16 192 Keepalived\_healthcheckers[15334]: Using LinkWatch kernel netlink reflector...

Apr 20 20:22:19 192 Keepalived\_vrrp[15335]: VRRP\_Instance(VI\_1) Transition to MASTER STATE

Apr 20 20:22:20 192 Keepalived\_vrrp[15335]: VRRP\_Instance(VI\_1) Entering MASTER STATE

Apr 20 20:22:20 192 Keepalived\_vrrp[15335]: VRRP\_Instance(VI\_1) setting protocol VIPs.

Apr 20 20:22:20 192 Keepalived\_vrrp[15335]: VRRP\_Instance(VI\_1) Sending gratuitous ARPs on eth1 for 192.168.0.88

Apr 20 20:22:20 192 Keepalived\_healthcheckers[15334]: Netlink reflector reports IP 192.168.0.88 added

Apr 20 20:22:25 192 Keepalived\_vrrp[15335]: VRRP\_Instance(VI\_1) Sending gratuitous ARPs on eth1 for 192.168.0.88

[复制代码](javascript:void(0);)

发现已经将虚拟ip 192.168.0.88绑定了网卡eth1上。  
（4）查看绑定情况

[root@192.168.0.50 ~]# ip addr | grep eth1

3: eth1: <BROADCAST,MULTICAST,UP,LOWER\_UP> mtu 1500 qdisc pfifo\_fast state UP qlen 1000

inet 192.168.0.50/24 brd 192.168.0.255 scope global eth1

inet 192.168.0.88/32 scope global eth1

[root@192.168.0.50 ~]#

在另外一台服务器，候选master上启动keepalived服务，并观察

[复制代码](javascript:void(0);)

[root@192.168.0.60 ~]# /etc/init.d/keepalived start ; tail -f /var/log/messages

Starting keepalived: [ OK ]

Apr 20 20:26:18 192 Keepalived\_vrrp[9472]: Registering gratuitous ARP shared channel

Apr 20 20:26:18 192 Keepalived\_vrrp[9472]: Opening file '/etc/keepalived/keepalived.conf'.

Apr 20 20:26:18 192 Keepalived\_vrrp[9472]: Configuration is using : 62976 Bytes

Apr 20 20:26:18 192 Keepalived\_vrrp[9472]: Using LinkWatch kernel netlink reflector...

Apr 20 20:26:18 192 Keepalived\_vrrp[9472]: VRRP\_Instance(VI\_1) Entering BACKUP STATE

Apr 20 20:26:18 192 Keepalived\_vrrp[9472]: VRRP sockpool: [ifindex(3), proto(112), unicast(0), fd(10,11)]

Apr 20 20:26:18 192 Keepalived\_healthcheckers[9471]: Netlink reflector reports IP 192.168.80.138 added

Apr 20 20:26:18 192 Keepalived\_healthcheckers[9471]: Netlink reflector reports IP 192.168.0.60 added

Apr 20 20:26:18 192 Keepalived\_healthcheckers[9471]: Netlink reflector reports IP fe80::20c:29ff:fe9d:6a9e added

Apr 20 20:26:18 192 Keepalived\_healthcheckers[9471]: Netlink reflector reports IP fe80::20c:29ff:fe9d:6aa8 added

Apr 20 20:26:18 192 Keepalived\_healthcheckers[9471]: Registering Kernel netlink reflector

Apr 20 20:26:18 192 Keepalived\_healthcheckers[9471]: Registering Kernel netlink command channel

Apr 20 20:26:18 192 Keepalived\_healthcheckers[9471]: Opening file '/etc/keepalived/keepalived.conf'.

Apr 20 20:26:18 192 Keepalived\_healthcheckers[9471]: Configuration is using : 7231 Bytes

Apr 20 20:26:18 192 kernel: IPVS: Registered protocols (TCP, UDP, AH, ESP)

Apr 20 20:26:18 192 kernel: IPVS: Connection hash table configured (size=4096, memory=64Kbytes)

Apr 20 20:26:18 192 kernel: IPVS: ipvs loaded.

Apr 20 20:26:18 192 Keepalived\_healthcheckers[9471]: Using LinkWatch kernel netlink reflector...

[复制代码](javascript:void(0);)

从上面的信息可以看到keepalived已经配置成功。  
**注意：**

**上面两台服务器的keepalived都设置为了BACKUP模式，在keepalived中2种模式，分别是master->backup模式和backup->backup模式。这两种模式有很大区别。在master->backup模式下，一旦主库宕机，虚拟ip会自动漂移到从库，当主库修复后，keepalived启动后，还会把虚拟ip抢占过来，即使设置了非抢占模式（nopreempt）抢占ip的动作也会发生。在backup->backup模式下，当主库宕机后虚拟ip会自动漂移到从库上，当原主库恢复和keepalived服务启动后，并不会抢占新主的虚拟ip，即使是优先级高于从库的优先级别，也不会发生抢占。为了减少ip漂移次数，通常是把修复好的主库当做新的备库。**

**（5）MHA引入keepalived（MySQL服务进程挂掉时通过MHA 停止keepalived）:**

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

编辑脚本/usr/local/bin/master\_ip\_failover，修改后如下，我对perl不熟悉，所以我这里完整贴出该脚本（主库上操作，192.168.0.50）。

在MHA Manager修改脚本修改后的内容如下（参考资料比较少）：

[复制代码](javascript:void(0);)

#!/usr/bin/env perl

use strict;

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

use Getopt::Long;

my (

$command, $ssh\_user, $orig\_master\_host, $orig\_master\_ip,

$orig\_master\_port, $new\_master\_host, $new\_master\_ip, $new\_master\_port

);

my $vip = **'192.168.0.88'**;

my $ssh\_start\_vip = **"/etc/init.d/keepalived start"**;

my $ssh\_stop\_vip = **"/etc/init.d/keepalived stop"**;

GetOptions(

**'command=s'** => \$command,

**'ssh\_user=s'** => \$ssh\_user,

**'orig\_master\_host=s'** => \$orig\_master\_host,

**'orig\_master\_ip=s'** => \$orig\_master\_ip,

**'orig\_master\_port=i'** => \$orig\_master\_port,

**'new\_master\_host=s'** => \$new\_master\_host,

**'new\_master\_ip=s'** => \$new\_master\_ip,

**'new\_master\_port=i'** => \$new\_master\_port,

);

exit &main();

sub main {

print **"\n\nIN SCRIPT TEST====$ssh\_stop\_vip==$ssh\_start\_vip===\n\n"**;

if ( $command eq **"stop"** || $command eq **"stopssh"** ) {

my $exit\_code = 1;

eval {

print **"Disabling the VIP on old master: $orig\_master\_host \n"**;

&stop\_vip();

$exit\_code = 0;

};

if ($@) {

warn **"Got Error: $@\n"**;

exit $exit\_code;

}

exit $exit\_code;

}

elsif ( $command eq **"start"** ) {

my $exit\_code = 10;

eval {

print **"Enabling the VIP - $vip on the new master - $new\_master\_host \n"**;

&start\_vip();

$exit\_code = 0;

};

if ($@) {

warn $@;

exit $exit\_code;

}

exit $exit\_code;

}

elsif ( $command eq **"status"** ) {

print **"Checking the Status of the script.. OK \n"**;

#`ssh $ssh\_user\@cluster1 \" $ssh\_start\_vip \"`;

exit 0;

}

else {

&usage();

exit 1;

}

}

# A simple system call that enable the VIP on the new master

sub start\_vip() {

`ssh $ssh\_user\@$new\_master\_host \**" $ssh\_start\_vip \"`;**

**}**

**# A simple system call that disable the VIP on the old\_master**

**sub stop\_vip() {  
 return 0  unless  ($ssh\_user);**

**`ssh $ssh\_user\@$orig\_master\_host \" $ssh\_stop\_vip \"`;**

**}**

**sub usage {**

**print**

**"**Usage: master\_ip\_failover --command=start|stop|stopssh|status --orig\_master\_host=host --orig\_master\_ip=ip --orig\_master\_port=port --new\_master\_host=host --new\_master\_ip=ip --new\_master\_port=port\n**";**

**}**

[复制代码](javascript:void(0);)

现在已经修改这个脚本了，我们现在打开在上面提到过的参数，再检查集群状态，看是否会报错。

[root@192.168.0.20 ~]# grep 'master\_ip\_failover\_script' /etc/masterha/app1.cnf

master\_ip\_failover\_script= /usr/local/bin/master\_ip\_failover

[root@192.168.0.20 ~]#

[复制代码](javascript:void(0);)

[root@192.168.0.20 ~]# masterha\_check\_repl --conf=/etc/masterha/app1.cnf

Sun Apr 20 23:10:01 2014 - [info] Slaves settings check done.

Sun Apr 20 23:10:01 2014 - [info]

192.168.0.50 (current master)

+--192.168.0.60

+--192.168.0.70

Sun Apr 20 23:10:01 2014 - [info] Checking replication health on 192.168.0.60..

Sun Apr 20 23:10:01 2014 - [info] ok.

Sun Apr 20 23:10:01 2014 - [info] Checking replication health on 192.168.0.70..

Sun Apr 20 23:10:01 2014 - [info] ok.

Sun Apr 20 23:10:01 2014 - [info] Checking master\_ip\_failover\_script status:

Sun Apr 20 23:10:01 2014 - [info] /usr/local/bin/master\_ip\_failover --command=status --ssh\_user=root --orig\_master\_host=192.168.0.50 --orig\_master\_ip=192.168.0.50 --orig\_master\_port=3306

Sun Apr 20 23:10:01 2014 - [info] OK.

Sun Apr 20 23:10:01 2014 - [warning] shutdown\_script is not defined.

Sun Apr 20 23:10:01 2014 - [info] Got exit code 0 (Not master dead).

MySQL Replication Health is OK.

[复制代码](javascript:void(0);)

可以看见已经没有报错了。哈哈  
 /usr/local/bin/master\_ip\_failover添加或者修改的内容意思是当主库数据库发生故障时，会触发MHA切换，MHA Manager会停掉主库上的keepalived服务，触发虚拟ip漂移到备选从库，从而完成切换。当然可以在keepalived里面引入脚本，这个脚本监控mysql是否正常运行，如果不正常，则调用该脚本杀掉keepalived进程。

**2.通过脚本的方式管理VIP。这里是修改/usr/local/bin/master\_ip\_failover，也可以使用其他的语言完成，比如php语言。使用php脚本编写的failover这里就不介绍了。修改完成后内容如下，而且如果使用脚本管理vip的话，需要手动在master服务器上绑定一个vip（发现修改修改对perl竟然有感觉了。难道我适合学Perl？^\_^）**

[root@192.168.0.50 ~]# /sbin/ifconfig eth1:1 192.168.0.88/24

通过脚本来维护vip的测试我这里就不说明了，童鞋们自行测试，脚本如下（测试通过）

[复制代码](javascript:void(0);)

#!/usr/bin/env perl

use strict;

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

use Getopt::Long;

my (

$command, $ssh\_user, $orig\_master\_host, $orig\_master\_ip,

$orig\_master\_port, $new\_master\_host, $new\_master\_ip, $new\_master\_port

);

my $vip = **'192.168.0.88/24'**;

my $key = **'1'**;

my $ssh\_start\_vip = **"/sbin/ifconfig eth1:$key $vip"**;

my $ssh\_stop\_vip = **"/sbin/ifconfig eth1:$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**";**

**}**

## MHA工作情况

为了防止脑裂发生，推荐生产环境采用脚本的方式来管理虚拟ip，而不是使用keepalived来完成。到此为止，基本MHA集群已经配置完毕。接下来就是实际的测试环节了。通过一些测试来看一下MHA到底是如何进行工作的。下面将从MHA自动failover，我们手动failover，在线切换三种方式来介绍MHA的工作情况。

### 1.自动Failover

**（必须先启动MHA Manager，否则无法自动切换，当然手动切换不需要开启MHA Manager监控。各位童鞋请参考前面启动MHA Manager）**

测试环境再次贴一下，文章太长，自己都搞晕了。

角色 ip地址 主机名 server\_id 类型

Monitor host 192.168.0.20 server01 - 监控复制组

Master 192.168.0.50 server02 1 写入

Candicate master 192.168.0.60 server03 2 读

Slave 192.168.0.70 server04 3 读

自动failover模拟测试的操作步骤如下。  
（1）使用sysbench生成测试数据（使用yum快速安装）

yum install sysbench -y

在主库（192.168.0.50）上进行sysbench数据生成，在sbtest库下生成sbtest表，共100W记录。

[root@192.168.0.50 ~]# sysbench --test=oltp --oltp-table-size=1000000 --oltp-read-only=off --init-rng=on --num-threads=16 --max-requests=0 --oltp-dist-type=uniform --max-time=1800 --mysql-user=root --mysql-socket=/tmp/mysql.sock --mysql-password=123456 --db-driver=mysql --mysql-table-engine=innodb --oltp-test-mode=complex prepare

（2）停掉slave sql线程，模拟主从延时。（192.168.0.60）

mysql> stop slave io\_thread;

Query OK, **0** rows affected (**0.08** sec)

mysql>

另外一台slave我们没有停止io线程，所以还在继续接收日志。

（3）模拟sysbench压力测试。

在主库上（192.168.0.50）进行压力测试，持续时间为3分钟，产生大量的binlog。

[复制代码](javascript:void(0);)

[root@192.168.0.50 ~]# sysbench --test=oltp --oltp-table-size=1000000 --oltp-read-only=off --init-rng=on --num-threads=16 --max-requests=0 --oltp-dist-type=uniform --max-time=180 --mysql-user=root --mysql-socket=/tmp/mysql.sock --mysql-password=123456 --db-driver=mysql --mysql-table-engine=innodb --oltp-test-mode=complex run

sysbench 0.4.12: multi-threaded system evaluation benchmark

Running the test with following options:

Number of threads: 16

Initializing random number generator from timer.

Doing OLTP test.

Running mixed OLTP test

Using Uniform distribution

Using "BEGIN" for starting transactions

Using auto\_inc on the id column

Threads started!

Time limit exceeded, exiting...

(last message repeated 15 times)

Done.

OLTP test statistics:

queries performed:

read: 15092

write: 5390

other: 2156

total: 22638

transactions: 1078 (5.92 per sec.)

deadlocks: 0 (0.00 per sec.)

read/write requests: 20482 (112.56 per sec.)

other operations: 2156 (11.85 per sec.)

Test execution summary:

total time: 181.9728s

total number of events: 1078

total time taken by event execution: 2910.4518

per-request statistics:

min: 934.29ms

avg: 2699.86ms

max: 7679.95ms

approx. 95 percentile: 4441.47ms

Threads fairness:

events (avg/stddev): 67.3750/1.49

execution time (avg/stddev): 181.9032/0.11

[复制代码](javascript:void(0);)

（4）开启slave（192.168.0.60）上的IO线程，追赶落后于master的binlog。

mysql> start slave io\_thread;

Query OK, **0** rows affected (**0.00** sec)

mysql>

（5）杀掉主库mysql进程，模拟主库发生故障，进行自动failover操作。

[root@192.168.0.50 ~]# pkill -9 mysqld

（6）查看MHA切换日志，了解整个切换过程，在192.168.0.20上查看日志：

https://images.cnblogs.com/OutliningIndicators/ContractedBlock.gifhttps://images.cnblogs.com/OutliningIndicators/ExpandedBlockStart.gif

[root@192.168.0.20 ~]# cat /var/log/masterha/app1/manager.log

Mon Apr 21 20:15:45 2014 - [warning] Got error on MySQL select ping: 2006 (MySQL server has gone away)

Mon Apr 21 20:15:45 2014 - [info] Executing seconary network check script: /usr/local/bin/masterha\_secondary\_check -s server03 -s server02 --user=root --master\_host=server02 --master\_ip=192.168.0.50 --master\_ Creating /tmp if not exists.. ok.

Checking output directory is accessible or not..

ok.

Binlog found at /data/mysql, up to mysql-bin.000018

Mon Apr 21 20:15:48 2014 - [warning] Global configuration file /etc/masterha\_default.cnf not found. Skipping.

Mon Apr 21 20:15:48 2014 - [info] Reading application default configurations from /etc/masterha/app1.cnf..

Mon Apr 21 20:15:48 2014 - [info] Reading server configurations from /etc/masterha/app1.cnf..

ble from server03. OK.

Monitoring server server02 is reachable, Master is not reachable from server02. OK.

Mon Apr 21 20:15:46 2014 - [info] Master is not reachable from all other monitoring servers. Failover should start.

Mon Apr 21 20:15:46 2014 - [warning] Got error on MySQL connect: 2013 (Lost connection to MySQL server at 'reading initial communication packet', system error: 111)

Mon Apr 21 20:15:46 2014 - [warning] Connection failed 1 time(s)..

Mon Apr 21 20:15:47 2014 - [warning] Got error on MySQL connect: 2013 (Lost connection to MySQL server at 'reading initial communication packet', system error: 111)

Mon Apr 21 20:15:47 2014 - [warning] Connection failed 2 time(s)..

Mon Apr 21 20:15:48 2014 - [warning] Got error on MySQL connect: 2013 (Lost connection to MySQL server at 'reading initial communication packet', system error: 111)

Mon Apr 21 20:15:48 2014 - [warning] Connection failed 3 time(s)..

Mon Apr 21 20:15:48 2014 - [warning] Master is not reachable from health checker!

Mon Apr 21 20:15:48 2014 - [warning] Master 192.168.0.50(192.168.0.50:3306) is not reachable!

Mon Apr 21 20:15:48 2014 - [warning] SSH is reachable.

Mon Apr 21 20:15:48 2014 - [info] Connecting to a master server failed. Reading configuration file /etc/masterha\_default.cnf and /etc/masterha/app1.cnf again, and trying to connect to all servers to check server status..

Mon Apr 21 20:15:48 2014 - [warning] Global configuration file /etc/masterha\_default.cnf not found. Skipping.

Mon Apr 21 20:15:48 2014 - [info] Reading application default configurations from /etc/masterha/app1.cnf..

Mon Apr 21 20:15:48 2014 - [info] Reading server configurations from /etc/masterha/app1.cnf..

Mon Apr 21 20:15:48 2014 - [info] Dead Servers:

Mon Apr 21 20:15:48 2014 - [info] 192.168.0.50(192.168.0.50:3306)

Mon Apr 21 20:15:48 2014 - [info] Alive Servers:

Mon Apr 21 20:15:48 2014 - [info] 192.168.0.60(192.168.0.60:3306)

Mon Apr 21 20:15:48 2014 - [info] 192.168.0.70(192.168.0.70:3306)

Mon Apr 21 20:15:48 2014 - [info] Alive Slaves:

Mon Apr 21 20:15:48 2014 - [info] 192.168.0.60(192.168.0.60:3306) Version=5.5.19-ndb-7.2.4-gpl-log (oldest major version between slaves) log-bin:enabled

Mon Apr 21 20:15:48 2014 - [info] Replicating from 192.168.0.50(192.168.0.50:3306)

Mon Apr 21 20:15:48 2014 - [info] Primary candidate for the new Master (candidate\_master is set)

Mon Apr 21 20:15:48 2014 - [info] 192.168.0.70(192.168.0.70:3306) Version=5.5.19-ndb-7.2.4-gpl-log (oldest major version between slaves) log-bin:enabled

Mon Apr 21 20:15:48 2014 - [info] Replicating from 192.168.0.50(192.168.0.50:3306)

Mon Apr 21 20:15:48 2014 - [info] Checking slave configurations..

Mon Apr 21 20:15:48 2014 - [info] Checking replication filtering settings..

Mon Apr 21 20:15:48 2014 - [info] Replication filtering check ok.

Mon Apr 21 20:15:48 2014 - [info] Master is down!

Mon Apr 21 20:15:48 2014 - [info] Terminating monitoring script.

Mon Apr 21 20:15:48 2014 - [info] Got exit code 20 (Master dead).

Mon Apr 21 20:15:48 2014 - [info] MHA::MasterFailover version 0.53.

Mon Apr 21 20:15:48 2014 - [info] Starting master failover.

Mon Apr 21 20:15:48 2014 - [info]

Mon Apr 21 20:15:48 2014 - [info] \* Phase 1: Configuration Check Phase..

Mon Apr 21 20:15:48 2014 - [info]

Mon Apr 21 20:15:48 2014 - [info] Dead Servers:

Mon Apr 21 20:15:48 2014 - [info] 192.168.0.50(192.168.0.50:3306)

Mon Apr 21 20:15:48 2014 - [info] Checking master reachability via mysql(double check)..

Mon Apr 21 20:15:48 2014 - [info] ok.

Mon Apr 21 20:15:48 2014 - [info] Alive Servers:

Mon Apr 21 20:15:48 2014 - [info] 192.168.0.60(192.168.0.60:3306)

Mon Apr 21 20:15:48 2014 - [info] 192.168.0.70(192.168.0.70:3306)

Mon Apr 21 20:15:48 2014 - [info] Alive Slaves:

Mon Apr 21 20:15:48 2014 - [info] 192.168.0.60(192.168.0.60:3306) Version=5.5.19-ndb-7.2.4-gpl-log (oldest major version between slaves) log-bin:enabled

Mon Apr 21 20:15:48 2014 - [info] Replicating from 192.168.0.50(192.168.0.50:3306)

Mon Apr 21 20:15:48 2014 - [info] Primary candidate for the new Master (candidate\_master is set)

Mon Apr 21 20:15:48 2014 - [info] 192.168.0.70(192.168.0.70:3306) Version=5.5.19-ndb-7.2.4-gpl-log (oldest major version between slaves) log-bin:enabled

Mon Apr 21 20:15:48 2014 - [info] Replicating from 192.168.0.50(192.168.0.50:3306)

Mon Apr 21 20:15:49 2014 - [info] \*\* Phase 1: Configuration Check Phase completed.

Mon Apr 21 20:15:49 2014 - [info]

Mon Apr 21 20:15:49 2014 - [info] \* Phase 2: Dead Master Shutdown Phase..

Mon Apr 21 20:15:49 2014 - [info]

Mon Apr 21 20:15:49 2014 - [info] Forcing shutdown so that applications never connect to the current master..

Mon Apr 21 20:15:49 2014 - [info] Executing master IP deactivatation script:

Mon Apr 21 20:15:49 2014 - [info] /usr/local/bin/master\_ip\_failover --orig\_master\_host=192.168.0.50 --orig\_master\_ip=192.168.0.50 --orig\_master\_port=3306 --command=stopssh --ssh\_user=root

IN SCRIPT TEST====/etc/init.d/keepalived stop==/etc/init.d/keepalived start===

Disabling the VIP on old master: 192.168.0.50

Mon Apr 21 20:15:49 2014 - [info] done.

Mon Apr 21 20:15:49 2014 - [warning] shutdown\_script is not set. Skipping explicit shutting down of the dead master.

Mon Apr 21 20:15:49 2014 - [info] \* Phase 2: Dead Master Shutdown Phase completed.

Mon Apr 21 20:15:49 2014 - [info]

Mon Apr 21 20:15:49 2014 - [info] \* Phase 3: Master Recovery Phase..

Mon Apr 21 20:15:49 2014 - [info]

Mon Apr 21 20:15:49 2014 - [info] \* Phase 3.1: Getting Latest Slaves Phase..

Mon Apr 21 20:15:49 2014 - [info]

Mon Apr 21 20:15:49 2014 - [info] The latest binary log file/position on all slaves is mysql-bin.000018:112

Mon Apr 21 20:15:49 2014 - [info] Latest slaves (Slaves that received relay log files to the latest):

Mon Apr 21 20:15:49 2014 - [info] 192.168.0.60(192.168.0.60:3306) Version=5.5.19-ndb-7.2.4-gpl-log (oldest major version between slaves) log-bin:enabled

Mon Apr 21 20:15:49 2014 - [info] Replicating from 192.168.0.50(192.168.0.50:3306)

Mon Apr 21 20:15:49 2014 - [info] Primary candidate for the new Master (candidate\_master is set)

Mon Apr 21 20:15:49 2014 - [info] 192.168.0.70(192.168.0.70:3306) Version=5.5.19-ndb-7.2.4-gpl-log (oldest major version between slaves) log-bin:enabled

Mon Apr 21 20:15:49 2014 - [info] Replicating from 192.168.0.50(192.168.0.50:3306)

Mon Apr 21 20:15:49 2014 - [info] The oldest binary log file/position on all slaves is mysql-bin.000018:112

Mon Apr 21 20:15:49 2014 - [info] Oldest slaves:

Mon Apr 21 20:15:49 2014 - [info] 192.168.0.60(192.168.0.60:3306) Version=5.5.19-ndb-7.2.4-gpl-log (oldest major version between slaves) log-bin:enabled

Mon Apr 21 20:15:49 2014 - [info] Replicating from 192.168.0.50(192.168.0.50:3306)

Mon Apr 21 20:15:49 2014 - [info] Primary candidate for the new Master (candidate\_master is set)

Mon Apr 21 20:15:49 2014 - [info] 192.168.0.70(192.168.0.70:3306) Version=5.5.19-ndb-7.2.4-gpl-log (oldest major version between slaves) log-bin:enabled

Mon Apr 21 20:15:49 2014 - [info] Replicating from 192.168.0.50(192.168.0.50:3306)

Mon Apr 21 20:15:49 2014 - [info]

Mon Apr 21 20:15:49 2014 - [info] \* Phase 3.2: Saving Dead Master's Binlog Phase..

Mon Apr 21 20:15:49 2014 - [info]

Mon Apr 21 20:15:49 2014 - [info] Fetching dead master's binary logs..

Mon Apr 21 20:15:49 2014 - [info] Executing command on the dead master 192.168.0.50(192.168.0.50:3306): save\_binary\_logs --command=save --start\_file=mysql-bin.000018 --start\_pos=112 --binlog\_dir=/data/mysql --output\_file=/tmp/saved\_master\_binlog\_from\_192.168.0.50\_3306\_20140421201548.binlog --handle\_raw\_binlog=1 --disable\_log\_bin=0 --manager\_version=0.53

Creating /tmp if not exists.. ok.

Concat binary/relay logs from mysql-bin.000018 pos 112 to mysql-bin.000018 EOF into /tmp/saved\_master\_binlog\_from\_192.168.0.50\_3306\_20140421201548.binlog ..

Dumping binlog format description event, from position 0 to 112.. ok.

Dumping effective binlog data from /data/mysql/mysql-bin.000018 position 112 to tail(131).. ok.

Concat succeeded.

Mon Apr 21 20:15:50 2014 - [info] scp from root@192.168.0.50:/tmp/saved\_master\_binlog\_from\_192.168.0.50\_3306\_20140421201548.binlog to local:/var/log/masterha/app1.log/saved\_master\_binlog\_from\_192.168.0.50\_3306\_20140421201548.binlog succeeded.

Mon Apr 21 20:15:50 2014 - [info] HealthCheck: SSH to 192.168.0.60 is reachable.

Mon Apr 21 20:15:50 2014 - [info] HealthCheck: SSH to 192.168.0.70 is reachable.

Mon Apr 21 20:15:50 2014 - [info]

Mon Apr 21 20:15:50 2014 - [info] \* Phase 3.3: Determining New Master Phase..

Mon Apr 21 20:15:50 2014 - [info]

Mon Apr 21 20:15:50 2014 - [info] Finding the latest slave that has all relay logs for recovering other slaves..

Mon Apr 21 20:15:50 2014 - [info] All slaves received relay logs to the same position. No need to resync each other.

Mon Apr 21 20:15:50 2014 - [info] Searching new master from slaves..

Mon Apr 21 20:15:50 2014 - [info] Candidate masters from the configuration file:

Mon Apr 21 20:15:50 2014 - [info] 192.168.0.60(192.168.0.60:3306) Version=5.5.19-ndb-7.2.4-gpl-log (oldest major version between slaves) log-bin:enabled

Mon Apr 21 20:15:50 2014 - [info] Replicating from 192.168.0.50(192.168.0.50:3306)

Mon Apr 21 20:15:50 2014 - [info] Primary candidate for the new Master (candidate\_master is set)

Mon Apr 21 20:15:50 2014 - [info] Non-candidate masters:

Mon Apr 21 20:15:50 2014 - [info] Searching from candidate\_master slaves which have received the latest relay log events..

Mon Apr 21 20:15:50 2014 - [info] New master is 192.168.0.60(192.168.0.60:3306)

Mon Apr 21 20:15:50 2014 - [info] Starting master failover..

Mon Apr 21 20:15:50 2014 - [info]

From:

192.168.0.50 (current master)

+--192.168.0.60

+--192.168.0.70

To:

192.168.0.60 (new master)

+--192.168.0.70

Mon Apr 21 20:15:50 2014 - [info]

Mon Apr 21 20:15:50 2014 - [info] \* Phase 3.3: New Master Diff Log Generation Phase..

Mon Apr 21 20:15:50 2014 - [info]

Mon Apr 21 20:15:50 2014 - [info] This server has all relay logs. No need to generate diff files from the latest slave.

Mon Apr 21 20:15:50 2014 - [info] Sending binlog..

Mon Apr 21 20:15:51 2014 - [info] scp from local:/var/log/masterha/app1.log/saved\_master\_binlog\_from\_192.168.0.50\_3306\_20140421201548.binlog to root@192.168.0.60:/tmp/saved\_master\_binlog\_from\_192.168.0.50\_3306\_20140421201548.binlog succeeded.

Mon Apr 21 20:15:51 2014 - [info]

Mon Apr 21 20:15:51 2014 - [info] \* Phase 3.4: Master Log Apply Phase..

Mon Apr 21 20:15:51 2014 - [info]

Mon Apr 21 20:15:51 2014 - [info] \*NOTICE: If any error happens from this phase, manual recovery is needed.

Mon Apr 21 20:15:51 2014 - [info] Starting recovery on 192.168.0.60(192.168.0.60:3306)..

Mon Apr 21 20:15:51 2014 - [info] Generating diffs succeeded.

Mon Apr 21 20:15:51 2014 - [info] Waiting until all relay logs are applied.

Mon Apr 21 20:15:51 2014 - [info] done.

Mon Apr 21 20:15:51 2014 - [info] Getting slave status..

Mon Apr 21 20:15:51 2014 - [info] This slave(192.168.0.60)'s Exec\_Master\_Log\_Pos equals to Read\_Master\_Log\_Pos(mysql-bin.000018:112). No need to recover from Exec\_Master\_Log\_Pos.

Mon Apr 21 20:15:51 2014 - [info] Connecting to the target slave host 192.168.0.60, running recover script..

Mon Apr 21 20:15:51 2014 - [info] Executing command: apply\_diff\_relay\_logs --command=apply --slave\_user=root --slave\_host=192.168.0.60 --slave\_ip=192.168.0.60 --slave\_port=3306 --apply\_files=/tmp/saved\_master\_binlog\_from\_192.168.0.50\_3306\_20140421201548.binlog --workdir=/tmp --target\_version=5.5.19-ndb-7.2.4-gpl-log --timestamp=20140421201548 --handle\_raw\_binlog=1 --disable\_log\_bin=0 --manager\_version=0.53 --slave\_pass=xxx

Mon Apr 21 20:15:51 2014 - [info]

Applying differential binary/relay log files /tmp/saved\_master\_binlog\_from\_192.168.0.50\_3306\_20140421201548.binlog on 192.168.0.60:3306. This may take long time...

Applying log files succeeded.

Mon Apr 21 20:15:51 2014 - [info] All relay logs were successfully applied.

Mon Apr 21 20:15:51 2014 - [info] Getting new master's binlog name and position..

Mon Apr 21 20:15:51 2014 - [info] mysql-bin.000022:506716

Mon Apr 21 20:15:51 2014 - [info] All other slaves should start replication from here. Statement should be: CHANGE MASTER TO MASTER\_HOST='192.168.0.60', MASTER\_PORT=3306, MASTER\_LOG\_FILE='mysql-bin.000022', MASTER\_LOG\_POS=506716, MASTER\_USER='repl', MASTER\_PASSWORD='xxx';

Mon Apr 21 20:15:51 2014 - [info] Executing master IP activate script:

Mon Apr 21 20:15:51 2014 - [info] /usr/local/bin/master\_ip\_failover --command=start --ssh\_user=root --orig\_master\_host=192.168.0.50 --orig\_master\_ip=192.168.0.50 --orig\_master\_port=3306 --new\_master\_host=192.168.0.60 --new\_master\_ip=192.168.0.60 --new\_master\_port=3306

IN SCRIPT TEST====/etc/init.d/keepalived stop==/etc/init.d/keepalived start===

Enabling the VIP - 192.168.0.88 on the new master - 192.168.0.60

Mon Apr 21 20:15:52 2014 - [info] OK.

Mon Apr 21 20:15:52 2014 - [info] Setting read\_only=0 on 192.168.0.60(192.168.0.60:3306)..

Mon Apr 21 20:15:52 2014 - [info] ok.

Mon Apr 21 20:15:52 2014 - [info] \*\* Finished master recovery successfully.

Mon Apr 21 20:15:52 2014 - [info] \* Phase 3: Master Recovery Phase completed.

Mon Apr 21 20:15:52 2014 - [info]

Mon Apr 21 20:15:52 2014 - [info] \* Phase 4: Slaves Recovery Phase..

Mon Apr 21 20:15:52 2014 - [info]

Mon Apr 21 20:15:52 2014 - [info] \* Phase 4.1: Starting Parallel Slave Diff Log Generation Phase..

Mon Apr 21 20:15:52 2014 - [info]

Mon Apr 21 20:15:52 2014 - [info] -- Slave diff file generation on host 192.168.0.70(192.168.0.70:3306) started, pid: 31321. Check tmp log /var/log/masterha/app1.log/192.168.0.70\_3306\_20140421201548.log if it takes time..

Mon Apr 21 20:15:52 2014 - [info]

Mon Apr 21 20:15:52 2014 - [info] Log messages from 192.168.0.70 ...

Mon Apr 21 20:15:52 2014 - [info]

Mon Apr 21 20:15:52 2014 - [info] This server has all relay logs. No need to generate diff files from the latest slave.

Mon Apr 21 20:15:52 2014 - [info] End of log messages from 192.168.0.70.

Mon Apr 21 20:15:52 2014 - [info] -- 192.168.0.70(192.168.0.70:3306) has the latest relay log events.

Mon Apr 21 20:15:52 2014 - [info] Generating relay diff files from the latest slave succeeded.

Mon Apr 21 20:15:52 2014 - [info]

Mon Apr 21 20:15:52 2014 - [info] \* Phase 4.2: Starting Parallel Slave Log Apply Phase..

Mon Apr 21 20:15:52 2014 - [info]

Mon Apr 21 20:15:52 2014 - [info] -- Slave recovery on host 192.168.0.70(192.168.0.70:3306) started, pid: 31323. Check tmp log /var/log/masterha/app1.log/192.168.0.70\_3306\_20140421201548.log if it takes time..

Mon Apr 21 20:15:52 2014 - [info]

Mon Apr 21 20:15:52 2014 - [info] Log messages from 192.168.0.70 ...

Mon Apr 21 20:15:52 2014 - [info]

Mon Apr 21 20:15:52 2014 - [info] Sending binlog..

Mon Apr 21 20:15:52 2014 - [info] scp from local:/var/log/masterha/app1.log/saved\_master\_binlog\_from\_192.168.0.50\_3306\_20140421201548.binlog to root@192.168.0.70:/tmp/saved\_master\_binlog\_from\_192.168.0.50\_3306\_20140421201548.binlog succeeded.

Mon Apr 21 20:15:52 2014 - [info] Starting recovery on 192.168.0.70(192.168.0.70:3306)..

Mon Apr 21 20:15:52 2014 - [info] Generating diffs succeeded.

Mon Apr 21 20:15:52 2014 - [info] Waiting until all relay logs are applied.

Mon Apr 21 20:15:52 2014 - [info] done.

Mon Apr 21 20:15:52 2014 - [info] Getting slave status..

Mon Apr 21 20:15:52 2014 - [info] This slave(192.168.0.70)'s Exec\_Master\_Log\_Pos equals to Read\_Master\_Log\_Pos(mysql-bin.000018:112). No need to recover from Exec\_Master\_Log\_Pos.

Mon Apr 21 20:15:52 2014 - [info] Connecting to the target slave host 192.168.0.70, running recover script..

Mon Apr 21 20:15:52 2014 - [info] Executing command: apply\_diff\_relay\_logs --command=apply --slave\_user=root --slave\_host=192.168.0.70 --slave\_ip=192.168.0.70 --slave\_port=3306 --apply\_files=/tmp/saved\_master\_binlog\_from\_192.168.0.50\_3306\_20140421201548.binlog --workdir=/tmp --target\_version=5.5.19-ndb-7.2.4-gpl-log --timestamp=20140421201548 --handle\_raw\_binlog=1 --disable\_log\_bin=0 --manager\_version=0.53 --slave\_pass=xxx

Mon Apr 21 20:15:52 2014 - [info]

Applying differential binary/relay log files /tmp/saved\_master\_binlog\_from\_192.168.0.50\_3306\_20140421201548.binlog on 192.168.0.70:3306. This may take long time...

Applying log files succeeded.

Mon Apr 21 20:15:52 2014 - [info] All relay logs were successfully applied.

Mon Apr 21 20:15:52 2014 - [info] Resetting slave 192.168.0.70(192.168.0.70:3306) and starting replication from the new master 192.168.0.60(192.168.0.60:3306)..

Mon Apr 21 20:15:52 2014 - [info] Executed CHANGE MASTER.

Mon Apr 21 20:15:52 2014 - [info] Slave started.

Mon Apr 21 20:15:52 2014 - [info] End of log messages from 192.168.0.70.

Mon Apr 21 20:15:52 2014 - [info] -- Slave recovery on host 192.168.0.70(192.168.0.70:3306) succeeded.

Mon Apr 21 20:15:52 2014 - [info] All new slave servers recovered successfully.

Mon Apr 21 20:15:52 2014 - [info]

Mon Apr 21 20:15:52 2014 - [info] \* Phase 5: New master cleanup phease..

Mon Apr 21 20:15:52 2014 - [info]

Mon Apr 21 20:15:52 2014 - [info] Resetting slave info on the new master..

Mon Apr 21 20:15:53 2014 - [info] 192.168.0.60: Resetting slave info succeeded.

Mon Apr 21 20:15:53 2014 - [info] Master failover to 192.168.0.60(192.168.0.60:3306) completed successfully.

Mon Apr 21 20:15:53 2014 - [info] Deleted server1 entry from /etc/masterha/app1.cnf .

Mon Apr 21 20:15:53 2014 - [info]

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

app1: MySQL Master failover 192.168.0.50 to 192.168.0.60 succeeded

Master 192.168.0.50 is down!

Check MHA Manager logs at server01:/var/log/masterha/app1/manager.log for details.

Started automated(non-interactive) failover.

Invalidated master IP address on 192.168.0.50.

The latest slave 192.168.0.60(192.168.0.60:3306) has all relay logs for recovery.

Selected 192.168.0.60 as a new master.

192.168.0.60: OK: Applying all logs succeeded.

192.168.0.60: OK: Activated master IP address.

192.168.0.70: This host has the latest relay log events.

Generating relay diff files from the latest slave succeeded.

192.168.0.70: OK: Applying all logs succeeded. Slave started, replicating from 192.168.0.60.

192.168.0.60: Resetting slave info succeeded.

Master failover to 192.168.0.60(192.168.0.60:3306) completed successfully.

[root@192.168.0.20 ~]#

View Code

看到最后的Master failover to 192.168.0.60(192.168.0.60:3306) completed successfully.说明备选master现在已经上位了。

**从上面的输出可以看出整个MHA的切换过程，共包括以下的步骤：**

**1.配置文件检查阶段，这个阶段会检查整个集群配置文件配置**

**2.宕机的master处理，这个阶段包括虚拟ip摘除操作，主机关机操作（这个我这里还没有实现，需要研究）**

**3.复制dead maste和最新slave相差的relay log，并保存到MHA Manger具体的目录下**

**4.识别含有最新更新的slave**

**5.应用从master保存的二进制日志事件（binlog events）**

**6.提升一个slave为新的master进行复制**

**7.使其他的slave连接新的master进行复制**

最后启动MHA Manger监控，查看集群里面现在谁是master（在切换后监控就停止了。。。还有东西没搞对？）后来在官方网站看到这句话就明白了 。

Running MHA Manager from daemontools

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.

[root@192.168.0.20 ~]# masterha\_check\_status --conf=/etc/masterha/app1.cnf

app1 (pid:23971) is running(0:PING\_OK), master:192.168.0.60

[root@192.168.0.20 ~]#

### 2.手动Failover

**（MHA Manager必须没有运行）**

手动failover，这种场景意味着在业务上没有启用MHA自动切换功能，当主服务器故障时，人工手动调用MHA来进行故障切换操作，具体命令如下：

**注意：如果，MHA manager检测到没有dead的server，将报错，并结束failover：**

Mon Apr 21 21:23:33 2014 - [info] Dead Servers:

Mon Apr 21 21:23:33 2014 - [error][/usr/local/share/perl5/MHA/MasterFailover.pm, ln181] None of server is dead. Stop failover.

Mon Apr 21 21:23:33 2014 - [error][/usr/local/share/perl5/MHA/ManagerUtil.pm, ln178] Got ERROR: at /usr/local/bin/masterha\_master\_switch line 53

进行手动切换命令如下：

[root@192.168.0.20 ~]# masterha\_master\_switch --master\_state=dead --conf=/etc/masterha/app1.cnf --dead\_master\_host=192.168.0.50 --dead\_master\_port=3306 --new\_master\_host=192.168.0.60 --new\_master\_port=3306 --ignore\_last\_failover

输出的信息会询问你是否进行切换：

https://images.cnblogs.com/OutliningIndicators/ContractedBlock.gifhttps://images.cnblogs.com/OutliningIndicators/ExpandedBlockStart.gif

Mon Apr 21 21:28:00 2014 - [warning] Global configuration file /etc/masterha\_default.cnf not found. Skipping.

Mon Apr 21 21:28:00 2014 - [info] Reading application default configurations from /etc/masterha/app1.cnf..

Mon Apr 21 21:28:00 2014 - [info] Reading server configurations from /etc/masterha/app1.cnf..

Mon Apr 21 21:28:00 2014 - [info] MHA::MasterFailover version 0.53.

Mon Apr 21 21:28:00 2014 - [info] Starting master failover.

Mon Apr 21 21:28:00 2014 - [info]

Mon Apr 21 21:28:00 2014 - [info] \* Phase 1: Configuration Check Phase..

Mon Apr 21 21:28:00 2014 - [info]

Mon Apr 21 21:28:00 2014 - [info] Dead Servers:

Mon Apr 21 21:28:00 2014 - [info] 192.168.0.50(192.168.0.50:3306)

Mon Apr 21 21:28:00 2014 - [info] Checking master reachability via mysql(double check)..

Mon Apr 21 21:28:00 2014 - [info] ok.

Mon Apr 21 21:28:00 2014 - [info] Alive Servers:

Mon Apr 21 21:28:00 2014 - [info] 192.168.0.60(192.168.0.60:3306)

Mon Apr 21 21:28:00 2014 - [info] 192.168.0.70(192.168.0.70:3306)

Mon Apr 21 21:28:00 2014 - [info] Alive Slaves:

Mon Apr 21 21:28:00 2014 - [info] 192.168.0.60(192.168.0.60:3306) Version=5.5.19-ndb-7.2.4-gpl-log (oldest major version between slaves) log-bin:enabled

Mon Apr 21 21:28:00 2014 - [info] Replicating from 192.168.0.50(192.168.0.50:3306)

Mon Apr 21 21:28:00 2014 - [info] Primary candidate for the new Master (candidate\_master is set)

Mon Apr 21 21:28:00 2014 - [info] 192.168.0.70(192.168.0.70:3306) Version=5.5.19-ndb-7.2.4-gpl-log (oldest major version between slaves) log-bin:enabled

Mon Apr 21 21:28:00 2014 - [info] Replicating from 192.168.0.50(192.168.0.50:3306)

Master 192.168.0.50 is dead. Proceed? (yes/NO): yes

Mon Apr 21 21:36:01 2014 - [info] \*\* Phase 1: Configuration Check Phase completed.

Mon Apr 21 21:36:01 2014 - [info]

Mon Apr 21 21:36:01 2014 - [info] \* Phase 2: Dead Master Shutdown Phase..

Mon Apr 21 21:36:01 2014 - [info]

Mon Apr 21 21:36:01 2014 - [info] HealthCheck: SSH to 192.168.0.50 is reachable.

Mon Apr 21 21:36:01 2014 - [info] Forcing shutdown so that applications never connect to the current master..

Mon Apr 21 21:36:01 2014 - [info] Executing master IP deactivatation script:

Mon Apr 21 21:36:01 2014 - [info] /usr/local/bin/master\_ip\_failover --orig\_master\_host=192.168.0.50 --orig\_master\_ip=192.168.0.50 --orig\_master\_port=3306 --command=stopssh --ssh\_user=root

IN SCRIPT TEST====/sbin/ifconfig eth1:1 down==/sbin/ifconfig eth1:1 192.168.0.88/24===

Disabling the VIP on old master: 192.168.0.50

Mon Apr 21 21:36:02 2014 - [info] done.

Mon Apr 21 21:36:02 2014 - [warning] shutdown\_script is not set. Skipping explicit shutting down of the dead master.

Mon Apr 21 21:36:02 2014 - [info] \* Phase 2: Dead Master Shutdown Phase completed.

Mon Apr 21 21:36:02 2014 - [info]

Mon Apr 21 21:36:02 2014 - [info] \* Phase 3: Master Recovery Phase..

Mon Apr 21 21:36:02 2014 - [info]

Mon Apr 21 21:36:02 2014 - [info] \* Phase 3.1: Getting Latest Slaves Phase..

Mon Apr 21 21:36:02 2014 - [info]

Mon Apr 21 21:36:02 2014 - [info] The latest binary log file/position on all slaves is mysql-bin.000020:112

Mon Apr 21 21:36:02 2014 - [info] Latest slaves (Slaves that received relay log files to the latest):

Mon Apr 21 21:36:02 2014 - [info] 192.168.0.60(192.168.0.60:3306) Version=5.5.19-ndb-7.2.4-gpl-log (oldest major version between slaves) log-bin:enabled

Mon Apr 21 21:36:02 2014 - [info] Replicating from 192.168.0.50(192.168.0.50:3306)

Mon Apr 21 21:36:02 2014 - [info] Primary candidate for the new Master (candidate\_master is set)

Mon Apr 21 21:36:02 2014 - [info] 192.168.0.70(192.168.0.70:3306) Version=5.5.19-ndb-7.2.4-gpl-log (oldest major version between slaves) log-bin:enabled

Mon Apr 21 21:36:02 2014 - [info] Replicating from 192.168.0.50(192.168.0.50:3306)

Mon Apr 21 21:36:02 2014 - [info] The oldest binary log file/position on all slaves is mysql-bin.000020:112

Mon Apr 21 21:36:02 2014 - [info] Oldest slaves:

Mon Apr 21 21:36:02 2014 - [info] 192.168.0.60(192.168.0.60:3306) Version=5.5.19-ndb-7.2.4-gpl-log (oldest major version between slaves) log-bin:enabled

Mon Apr 21 21:36:02 2014 - [info] Replicating from 192.168.0.50(192.168.0.50:3306)

Mon Apr 21 21:36:02 2014 - [info] Primary candidate for the new Master (candidate\_master is set)

Mon Apr 21 21:36:02 2014 - [info] 192.168.0.70(192.168.0.70:3306) Version=5.5.19-ndb-7.2.4-gpl-log (oldest major version between slaves) log-bin:enabled

Mon Apr 21 21:36:02 2014 - [info] Replicating from 192.168.0.50(192.168.0.50:3306)

Mon Apr 21 21:36:02 2014 - [info]

Mon Apr 21 21:36:02 2014 - [info] \* Phase 3.2: Saving Dead Master's Binlog Phase..

Mon Apr 21 21:36:02 2014 - [info]

Mon Apr 21 21:36:02 2014 - [info] Fetching dead master's binary logs..

Mon Apr 21 21:36:02 2014 - [info] Executing command on the dead master 192.168.0.50(192.168.0.50:3306): save\_binary\_logs --command=save --start\_file=mysql-bin.000020 --start\_pos=112 --binlog\_dir=/data/mysql --output\_file=/tmp/saved\_master\_binlog\_from\_192.168.0.50\_3306\_20140421212800.binlog --handle\_raw\_binlog=1 --disable\_log\_bin=0 --manager\_version=0.53

Creating /tmp if not exists.. ok.

Concat binary/relay logs from mysql-bin.000020 pos 112 to mysql-bin.000020 EOF into /tmp/saved\_master\_binlog\_from\_192.168.0.50\_3306\_20140421212800.binlog ..

Dumping binlog format description event, from position 0 to 112.. ok.

Dumping effective binlog data from /data/mysql/mysql-bin.000020 position 112 to tail(131).. ok.

Concat succeeded.

saved\_master\_binlog\_from\_192.168.0.50\_3306\_20140421212800.binlog 100% 131 0.1KB/s 00:00

Mon Apr 21 21:36:02 2014 - [info] scp from root@192.168.0.50:/tmp/saved\_master\_binlog\_from\_192.168.0.50\_3306\_20140421212800.binlog to local:/var/log/masterha/app1.log/saved\_master\_binlog\_from\_192.168.0.50\_3306\_20140421212800.binlog succeeded.

Mon Apr 21 21:36:02 2014 - [info] HealthCheck: SSH to 192.168.0.60 is reachable.

Mon Apr 21 21:36:03 2014 - [info] HealthCheck: SSH to 192.168.0.70 is reachable.

Mon Apr 21 21:36:03 2014 - [info]

Mon Apr 21 21:36:03 2014 - [info] \* Phase 3.3: Determining New Master Phase..

Mon Apr 21 21:36:03 2014 - [info]

Mon Apr 21 21:36:03 2014 - [info] Finding the latest slave that has all relay logs for recovering other slaves..

Mon Apr 21 21:36:03 2014 - [info] All slaves received relay logs to the same position. No need to resync each other.

Mon Apr 21 21:36:03 2014 - [info] 192.168.0.60 can be new master.

Mon Apr 21 21:36:03 2014 - [info] New master is 192.168.0.60(192.168.0.60:3306)

Mon Apr 21 21:36:03 2014 - [info] Starting master failover..

Mon Apr 21 21:36:03 2014 - [info]

From:

192.168.0.50 (current master)

+--192.168.0.60

+--192.168.0.70

To:

192.168.0.60 (new master)

+--192.168.0.70

Starting master switch from 192.168.0.50(192.168.0.50:3306) to 192.168.0.60(192.168.0.60:3306)? (yes/NO): yes

Mon Apr 21 21:36:06 2014 - [info] New master decided manually is 192.168.0.60(192.168.0.60:3306)

Mon Apr 21 21:36:06 2014 - [info]

Mon Apr 21 21:36:06 2014 - [info] \* Phase 3.3: New Master Diff Log Generation Phase..

Mon Apr 21 21:36:06 2014 - [info]

Mon Apr 21 21:36:06 2014 - [info] This server has all relay logs. No need to generate diff files from the latest slave.

Mon Apr 21 21:36:06 2014 - [info] Sending binlog..

saved\_master\_binlog\_from\_192.168.0.50\_3306\_20140421212800.binlog 100% 131 0.1KB/s 00:00

Mon Apr 21 21:36:07 2014 - [info] scp from local:/var/log/masterha/app1.log/saved\_master\_binlog\_from\_192.168.0.50\_3306\_20140421212800.binlog to root@192.168.0.60:/tmp/saved\_master\_binlog\_from\_192.168.0.50\_3306\_20140421212800.binlog succeeded.

Mon Apr 21 21:36:07 2014 - [info]

Mon Apr 21 21:36:07 2014 - [info] \* Phase 3.4: Master Log Apply Phase..

Mon Apr 21 21:36:07 2014 - [info]

Mon Apr 21 21:36:07 2014 - [info] \*NOTICE: If any error happens from this phase, manual recovery is needed.

Mon Apr 21 21:36:07 2014 - [info] Starting recovery on 192.168.0.60(192.168.0.60:3306)..

Mon Apr 21 21:36:07 2014 - [info] Generating diffs succeeded.

Mon Apr 21 21:36:07 2014 - [info] Waiting until all relay logs are applied.

Mon Apr 21 21:36:07 2014 - [info] done.

Mon Apr 21 21:36:07 2014 - [info] Getting slave status..

Mon Apr 21 21:36:07 2014 - [info] This slave(192.168.0.60)'s Exec\_Master\_Log\_Pos equals to Read\_Master\_Log\_Pos(mysql-bin.000020:112). No need to recover from Exec\_Master\_Log\_Pos.

Mon Apr 21 21:36:07 2014 - [info] Connecting to the target slave host 192.168.0.60, running recover script..

Mon Apr 21 21:36:07 2014 - [info] Executing command: apply\_diff\_relay\_logs --command=apply --slave\_user=root --slave\_host=192.168.0.60 --slave\_ip=192.168.0.60 --slave\_port=3306 --apply\_files=/tmp/saved\_master\_binlog\_from\_192.168.0.50\_3306\_20140421212800.binlog --workdir=/tmp --target\_version=5.5.19-ndb-7.2.4-gpl-log --timestamp=20140421212800 --handle\_raw\_binlog=1 --disable\_log\_bin=0 --manager\_version=0.53 --slave\_pass=xxx

Mon Apr 21 21:36:07 2014 - [info]

Applying differential binary/relay log files /tmp/saved\_master\_binlog\_from\_192.168.0.50\_3306\_20140421212800.binlog on 192.168.0.60:3306. This may take long time...

Applying log files succeeded.

Mon Apr 21 21:36:07 2014 - [info] All relay logs were successfully applied.

Mon Apr 21 21:36:07 2014 - [info] Getting new master's binlog name and position..

Mon Apr 21 21:36:07 2014 - [info] mysql-bin.000022:506716

Mon Apr 21 21:36:07 2014 - [info] All other slaves should start replication from here. Statement should be: CHANGE MASTER TO MASTER\_HOST='192.168.0.60', MASTER\_PORT=3306, MASTER\_LOG\_FILE='mysql-bin.000022', MASTER\_LOG\_POS=506716, MASTER\_USER='repl', MASTER\_PASSWORD='xxx';

Mon Apr 21 21:36:07 2014 - [info] Executing master IP activate script:

Mon Apr 21 21:36:07 2014 - [info] /usr/local/bin/master\_ip\_failover --command=start --ssh\_user=root --orig\_master\_host=192.168.0.50 --orig\_master\_ip=192.168.0.50 --orig\_master\_port=3306 --new\_master\_host=192.168.0.60 --new\_master\_ip=192.168.0.60 --new\_master\_port=3306

IN SCRIPT TEST====/sbin/ifconfig eth1:1 down==/sbin/ifconfig eth1:1 192.168.0.88/24===

Enabling the VIP - 192.168.0.88/24 on the new master - 192.168.0.60

Mon Apr 21 21:36:08 2014 - [info] OK.

Mon Apr 21 21:36:08 2014 - [info] Setting read\_only=0 on 192.168.0.60(192.168.0.60:3306)..

Mon Apr 21 21:36:08 2014 - [info] ok.

Mon Apr 21 21:36:08 2014 - [info] \*\* Finished master recovery successfully.

Mon Apr 21 21:36:08 2014 - [info] \* Phase 3: Master Recovery Phase completed.

Mon Apr 21 21:36:08 2014 - [info]

Mon Apr 21 21:36:08 2014 - [info] \* Phase 4: Slaves Recovery Phase..

Mon Apr 21 21:36:08 2014 - [info]

Mon Apr 21 21:36:08 2014 - [info] \* Phase 4.1: Starting Parallel Slave Diff Log Generation Phase..

Mon Apr 21 21:36:08 2014 - [info]

Mon Apr 21 21:36:08 2014 - [info] -- Slave diff file generation on host 192.168.0.70(192.168.0.70:3306) started, pid: 33518. Check tmp log /var/log/masterha/app1.log/192.168.0.70\_3306\_20140421212800.log if it takes time..

Mon Apr 21 21:36:08 2014 - [info]

Mon Apr 21 21:36:08 2014 - [info] Log messages from 192.168.0.70 ...

Mon Apr 21 21:36:08 2014 - [info]

Mon Apr 21 21:36:08 2014 - [info] This server has all relay logs. No need to generate diff files from the latest slave.

Mon Apr 21 21:36:08 2014 - [info] End of log messages from 192.168.0.70.

Mon Apr 21 21:36:08 2014 - [info] -- 192.168.0.70(192.168.0.70:3306) has the latest relay log events.

Mon Apr 21 21:36:08 2014 - [info] Generating relay diff files from the latest slave succeeded.

Mon Apr 21 21:36:08 2014 - [info]

Mon Apr 21 21:36:08 2014 - [info] \* Phase 4.2: Starting Parallel Slave Log Apply Phase..

Mon Apr 21 21:36:08 2014 - [info]

Mon Apr 21 21:36:08 2014 - [info] -- Slave recovery on host 192.168.0.70(192.168.0.70:3306) started, pid: 33520. Check tmp log /var/log/masterha/app1.log/192.168.0.70\_3306\_20140421212800.log if it takes time..

saved\_master\_binlog\_from\_192.168.0.50\_3306\_20140421212800.binlog 100% 131 0.1KB/s 00:00

Mon Apr 21 21:36:09 2014 - [info]

Mon Apr 21 21:36:09 2014 - [info] Log messages from 192.168.0.70 ...

Mon Apr 21 21:36:09 2014 - [info]

Mon Apr 21 21:36:08 2014 - [info] Sending binlog..

Mon Apr 21 21:36:08 2014 - [info] scp from local:/var/log/masterha/app1.log/saved\_master\_binlog\_from\_192.168.0.50\_3306\_20140421212800.binlog to root@192.168.0.70:/tmp/saved\_master\_binlog\_from\_192.168.0.50\_3306\_20140421212800.binlog succeeded.

Mon Apr 21 21:36:08 2014 - [info] Starting recovery on 192.168.0.70(192.168.0.70:3306)..

Mon Apr 21 21:36:08 2014 - [info] Generating diffs succeeded.

Mon Apr 21 21:36:08 2014 - [info] Waiting until all relay logs are applied.

Mon Apr 21 21:36:08 2014 - [info] done.

Mon Apr 21 21:36:08 2014 - [info] Getting slave status..

Mon Apr 21 21:36:08 2014 - [info] This slave(192.168.0.70)'s Exec\_Master\_Log\_Pos equals to Read\_Master\_Log\_Pos(mysql-bin.000020:112). No need to recover from Exec\_Master\_Log\_Pos.

Mon Apr 21 21:36:08 2014 - [info] Connecting to the target slave host 192.168.0.70, running recover script..

Mon Apr 21 21:36:08 2014 - [info] Executing command: apply\_diff\_relay\_logs --command=apply --slave\_user=root --slave\_host=192.168.0.70 --slave\_ip=192.168.0.70 --slave\_port=3306 --apply\_files=/tmp/saved\_master\_binlog\_from\_192.168.0.50\_3306\_20140421212800.binlog --workdir=/tmp --target\_version=5.5.19-ndb-7.2.4-gpl-log --timestamp=20140421212800 --handle\_raw\_binlog=1 --disable\_log\_bin=0 --manager\_version=0.53 --slave\_pass=xxx

Mon Apr 21 21:36:09 2014 - [info]

Applying differential binary/relay log files /tmp/saved\_master\_binlog\_from\_192.168.0.50\_3306\_20140421212800.binlog on 192.168.0.70:3306. This may take long time...

Applying log files succeeded.

Mon Apr 21 21:36:09 2014 - [info] All relay logs were successfully applied.

Mon Apr 21 21:36:09 2014 - [info] Resetting slave 192.168.0.70(192.168.0.70:3306) and starting replication from the new master 192.168.0.60(192.168.0.60:3306)..

Mon Apr 21 21:36:09 2014 - [info] Executed CHANGE MASTER.

Mon Apr 21 21:36:09 2014 - [info] Slave started.

Mon Apr 21 21:36:09 2014 - [info] End of log messages from 192.168.0.70.

Mon Apr 21 21:36:09 2014 - [info] -- Slave recovery on host 192.168.0.70(192.168.0.70:3306) succeeded.

Mon Apr 21 21:36:09 2014 - [info] All new slave servers recovered successfully.

Mon Apr 21 21:36:09 2014 - [info]

Mon Apr 21 21:36:09 2014 - [info] \* Phase 5: New master cleanup phease..

Mon Apr 21 21:36:09 2014 - [info]

Mon Apr 21 21:36:09 2014 - [info] Resetting slave info on the new master..

Mon Apr 21 21:36:09 2014 - [info] 192.168.0.60: Resetting slave info succeeded.

Mon Apr 21 21:36:09 2014 - [info] Master failover to 192.168.0.60(192.168.0.60:3306) completed successfully.

Mon Apr 21 21:36:09 2014 - [info]

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

app1: MySQL Master failover 192.168.0.50 to 192.168.0.60 succeeded

Master 192.168.0.50 is down!

Check MHA Manager logs at server01 for details.

Started manual(interactive) failover.

Invalidated master IP address on 192.168.0.50.

The latest slave 192.168.0.60(192.168.0.60:3306) has all relay logs for recovery.

Selected 192.168.0.60 as a new master.

192.168.0.60: OK: Applying all logs succeeded.

192.168.0.60: OK: Activated master IP address.

192.168.0.70: This host has the latest relay log events.

Generating relay diff files from the latest slave succeeded.

192.168.0.70: OK: Applying all logs succeeded. Slave started, replicating from 192.168.0.60.

192.168.0.60: Resetting slave info succeeded.

Master failover to 192.168.0.60(192.168.0.60:3306) completed successfully.

View Code

上述模拟了master宕机的情况下手动把192.168.0.60提升为主库的操作过程。

### 3.在线进行切换

 在许多情况下， 需要将现有的主服务器迁移到另外一台服务器上。 比如主服务器硬件故障，RAID 控制卡需要重建，将主服务器移到性能更好的服务器上等等。维护主服务器引起性能下降， 导致停机时间至少无法写入数据。 另外， 阻塞或杀掉当前运行的会话会导致主主之间数据不一致的问题发生。 MHA 提供快速切换和优雅的阻塞写入，这个切换过程只需要 0.5-2s 的时间，这段时间内数据是无法写入的。在很多情况下，0.5-2s 的阻塞写入是可以接受的。因此切换主服务器不需要计划分配维护时间窗口。

**MHA在线切换的大概过程：**  
1.检测复制设置和确定当前主服务器  
2.确定新的主服务器  
3.阻塞写入到当前主服务器  
4.等待所有从服务器赶上复制  
5.授予写入到新的主服务器  
6.重新设置从服务器

**注意，在线切换的时候应用架构需要考虑以下两个问题：**

1.自动识别master和slave的问题（master的机器可能会切换），如果采用了vip的方式，基本可以解决这个问题。

2.负载均衡的问题（可以定义大概的读写比例，每台机器可承担的负载比例，当有机器离开集群时，需要考虑这个问题）

**为了保证数据完全一致性，在最快的时间内完成切换，MHA的在线切换必须满足以下条件才会切换成功，否则会切换失败。**

**1.所有slave的IO线程都在运行**

**2.所有slave的SQL线程都在运行**

**3.所有的show slave status的输出中Seconds\_Behind\_Master参数小于或者等于running\_updates\_limit秒，如果在切换过程中不指定running\_updates\_limit,那么默认情况下running\_updates\_limit为1秒。**

**4.在master端，通过show processlist输出，没有一个更新花费的时间大于running\_updates\_limit秒。**

**在线切换步骤如下：**

首先，停掉MHA监控：

[root@192.168.0.20 ~]# masterha\_stop --conf=/etc/masterha/app1.cnf

其次，进行在线切换操作（模拟在线切换主库操作，原主库192.168.0.50变为slave，192.168.0.60提升为新的主库）

[root@192.168.0.20 ~]# masterha\_master\_switch --conf=/etc/masterha/app1.cnf --master\_state=alive --new\_master\_host=192.168.0.60 --new\_master\_port=3306 --orig\_master\_is\_new\_slave --running\_updates\_limit=10000

最后查看日志，了解切换过程，输出信息如下：

https://images.cnblogs.com/OutliningIndicators/ContractedBlock.gifhttps://images.cnblogs.com/OutliningIndicators/ExpandedBlockStart.gif

[root@192.168.0.20 ~]# masterha\_master\_switch --conf=/etc/masterha/app1.cnf --master\_state=alive --new\_master\_host=192.168.0.60 --new\_master\_port=3306 --orig\_master\_is\_new\_slave --running\_updates\_limit=10000

Wed Apr 23 00:27:39 2014 - [info] MHA::MasterRotate version 0.53.

Wed Apr 23 00:27:39 2014 - [info] Starting online master switch..

Wed Apr 23 00:27:39 2014 - [info]

Wed Apr 23 00:27:39 2014 - [info] \* Phase 1: Configuration Check Phase..

Wed Apr 23 00:27:39 2014 - [info]

Wed Apr 23 00:27:39 2014 - [info] Reading default configuratoins from /etc/masterha\_default.cnf..

Wed Apr 23 00:27:39 2014 - [info] Reading application default configurations from /etc/masterha/app1.cnf..

Wed Apr 23 00:27:39 2014 - [info] Reading server configurations from /etc/masterha/app1.cnf..

Wed Apr 23 00:27:39 2014 - [info] Multi-master configuration is detected. Current primary(writable) master is 192.168.0.50(192.168.0.50:3306)

Wed Apr 23 00:27:39 2014 - [info] Master configurations are as below:

Master 192.168.0.60(192.168.0.60:3306), replicating from 192.168.0.50(192.168.0.50:3306), read-only

Master 192.168.0.50(192.168.0.50:3306), replicating from 192.168.0.60(192.168.0.60:3306)

Wed Apr 23 00:27:39 2014 - [info] Current Alive Master: 192.168.0.50(192.168.0.50:3306)

Wed Apr 23 00:27:39 2014 - [info] Alive Slaves:

Wed Apr 23 00:27:39 2014 - [info] 192.168.0.60(192.168.0.60:3306) Version=5.5.19-ndb-7.2.4-gpl-log (oldest major version between slaves) log-bin:enabled

Wed Apr 23 00:27:39 2014 - [info] Replicating from 192.168.0.50(192.168.0.50:3306)

Wed Apr 23 00:27:39 2014 - [info] Primary candidate for the new Master (candidate\_master is set)

Wed Apr 23 00:27:39 2014 - [info] 192.168.0.70(192.168.0.70:3306) Version=5.5.19-ndb-7.2.4-gpl-log (oldest major version between slaves) log-bin:enabled

Wed Apr 23 00:27:39 2014 - [info] Replicating from 192.168.0.50(192.168.0.50:3306)

It is better to execute FLUSH NO\_WRITE\_TO\_BINLOG TABLES on the master before switching. Is it ok to execute on 192.168.0.50(192.168.0.50:3306)? (YES/no): yes

Wed Apr 23 00:27:40 2014 - [info] Executing FLUSH NO\_WRITE\_TO\_BINLOG TABLES. This may take long time..

Wed Apr 23 00:27:40 2014 - [info] ok.

Wed Apr 23 00:27:40 2014 - [info] Checking MHA is not monitoring or doing failover..

Wed Apr 23 00:27:40 2014 - [info] Checking replication health on 192.168.0.60..

Wed Apr 23 00:27:40 2014 - [info] ok.

Wed Apr 23 00:27:40 2014 - [info] Checking replication health on 192.168.0.70..

Wed Apr 23 00:27:40 2014 - [info] ok.

Wed Apr 23 00:27:40 2014 - [info] 192.168.0.60 can be new master.

Wed Apr 23 00:27:40 2014 - [info]

From:

192.168.0.50 (current master)

+--192.168.0.60

+--192.168.0.70

To:

192.168.0.60 (new master)

+--192.168.0.70

+--192.168.0.50

Starting master switch from 192.168.0.50(192.168.0.50:3306) to 192.168.0.60(192.168.0.60:3306)? (yes/NO): yes

Wed Apr 23 00:27:41 2014 - [info] Checking whether 192.168.0.60(192.168.0.60:3306) is ok for the new master..

Wed Apr 23 00:27:41 2014 - [info] ok.

Wed Apr 23 00:27:41 2014 - [info] \*\* Phase 1: Configuration Check Phase completed.

Wed Apr 23 00:27:41 2014 - [info]

Wed Apr 23 00:27:41 2014 - [info] \* Phase 2: Rejecting updates Phase..

Wed Apr 23 00:27:41 2014 - [info]

Wed Apr 23 00:27:41 2014 - [info] Executing master ip online change script to disable write on the current master:

Wed Apr 23 00:27:41 2014 - [info] /usr/local/bin/master\_ip\_online\_change.pl --command=stop --orig\_master\_host=192.168.0.50 --orig\_master\_ip=192.168.0.50 --orig\_master\_port=3306 --new\_master\_host=192.168.0.60 --new\_master\_ip=192.168.0.60 --new\_master\_port=3306

Wed Apr 23 00:27:41 2014 714804 Set read\_only on the new master.. ok.

Wed Apr 23 00:27:41 2014 719969 Set read\_only=1 on the orig master.. ok.

Disabling the VIP on old master: 192.168.0.50

reverse mapping checking getaddrinfo for bogon [192.168.0.50] failed - POSSIBLE BREAK-IN ATTEMPT!

Wed Apr 23 00:27:51 2014 963762 Killing all application threads..

Wed Apr 23 00:27:51 2014 963869 done.

Wed Apr 23 00:27:51 2014 - [info] ok.

Wed Apr 23 00:27:51 2014 - [info] Locking all tables on the orig master to reject updates from everybody (including root):

Wed Apr 23 00:27:51 2014 - [info] Executing FLUSH TABLES WITH READ LOCK..

Wed Apr 23 00:27:51 2014 - [info] ok.

Wed Apr 23 00:27:51 2014 - [info] Orig master binlog:pos is mysql-bin.000028:112.

Wed Apr 23 00:27:51 2014 - [info] Waiting to execute all relay logs on 192.168.0.60(192.168.0.60:3306)..

Wed Apr 23 00:27:51 2014 - [info] master\_pos\_wait(mysql-bin.000028:112) completed on 192.168.0.60(192.168.0.60:3306). Executed 0 events.

Wed Apr 23 00:27:51 2014 - [info] done.

Wed Apr 23 00:27:51 2014 - [info] Getting new master's binlog name and position..

Wed Apr 23 00:27:51 2014 - [info] mysql-bin.000023:1550

Wed Apr 23 00:27:51 2014 - [info] All other slaves should start replication from here. Statement should be: CHANGE MASTER TO MASTER\_HOST='192.168.0.60', MASTER\_PORT=3306, MASTER\_LOG\_FILE='mysql-bin.000023', MASTER\_LOG\_POS=1550, MASTER\_USER='repl', MASTER\_PASSWORD='xxx';

Wed Apr 23 00:27:51 2014 - [info] Executing master ip online change script to allow write on the new master:

Wed Apr 23 00:27:51 2014 - [info] /usr/local/bin/master\_ip\_online\_change.pl --command=start --orig\_master\_host=192.168.0.50 --orig\_master\_ip=192.168.0.50 --orig\_master\_port=3306 --new\_master\_host=192.168.0.60 --new\_master\_ip=192.168.0.60 --new\_master\_port=3306

Wed Apr 23 00:27:52 2014 077334 Set read\_only=0 on the new master.

Enabling the VIP - 192.168.0.88/24 on the new master - 192.168.0.60

reverse mapping checking getaddrinfo for bogon [192.168.0.60] failed - POSSIBLE BREAK-IN ATTEMPT!

Wed Apr 23 00:28:02 2014 - [info] ok.

Wed Apr 23 00:28:02 2014 - [info]

Wed Apr 23 00:28:02 2014 - [info] \* Switching slaves in parallel..

Wed Apr 23 00:28:02 2014 - [info]

Wed Apr 23 00:28:02 2014 - [info] -- Slave switch on host 192.168.0.70(192.168.0.70:3306) started, pid: 3036

Wed Apr 23 00:28:02 2014 - [info]

Wed Apr 23 00:28:02 2014 - [info] Log messages from 192.168.0.70 ...

Wed Apr 23 00:28:02 2014 - [info]

Wed Apr 23 00:28:02 2014 - [info] Waiting to execute all relay logs on 192.168.0.70(192.168.0.70:3306)..

Wed Apr 23 00:28:02 2014 - [info] master\_pos\_wait(mysql-bin.000028:112) completed on 192.168.0.70(192.168.0.70:3306). Executed 0 events.

Wed Apr 23 00:28:02 2014 - [info] done.

Wed Apr 23 00:28:02 2014 - [info] Resetting slave 192.168.0.70(192.168.0.70:3306) and starting replication from the new master 192.168.0.60(192.168.0.60:3306)..

Wed Apr 23 00:28:02 2014 - [info] Executed CHANGE MASTER.

Wed Apr 23 00:28:02 2014 - [info] Slave started.

Wed Apr 23 00:28:02 2014 - [info] End of log messages from 192.168.0.70 ...

Wed Apr 23 00:28:02 2014 - [info]

Wed Apr 23 00:28:02 2014 - [info] -- Slave switch on host 192.168.0.70(192.168.0.70:3306) succeeded.

Wed Apr 23 00:28:02 2014 - [info] Unlocking all tables on the orig master:

Wed Apr 23 00:28:02 2014 - [info] Executing UNLOCK TABLES..

Wed Apr 23 00:28:02 2014 - [info] ok.

Wed Apr 23 00:28:02 2014 - [info] Starting orig master as a new slave..

Wed Apr 23 00:28:02 2014 - [info] Resetting slave 192.168.0.50(192.168.0.50:3306) and starting replication from the new master 192.168.0.60(192.168.0.60:3306)..

Wed Apr 23 00:28:02 2014 - [info] Executed CHANGE MASTER.

Wed Apr 23 00:28:02 2014 - [info] Slave started.

Wed Apr 23 00:28:02 2014 - [info] All new slave servers switched successfully.

Wed Apr 23 00:28:02 2014 - [info]

Wed Apr 23 00:28:02 2014 - [info] \* Phase 5: New master cleanup phease..

Wed Apr 23 00:28:02 2014 - [info]

Wed Apr 23 00:28:02 2014 - [info] 192.168.0.60: Resetting slave info succeeded.

Wed Apr 23 00:28:02 2014 - [info] Switching master to 192.168.0.60(192.168.0.60:3306) completed successfully.

https://images.cnblogs.com/OutliningIndicators/ExpandedBlockStart.gif

[复制代码](javascript:void(0);)

[root@192.168.0.20 ~]# masterha\_master\_switch --conf=/etc/masterha/app1.cnf --master\_state=alive --new\_master\_host=192.168.0.60 --new\_master\_port=3306 --orig\_master\_is\_new\_slave --running\_updates\_limit=10000

Wed Apr 23 00:27:39 2014 - [info] MHA::MasterRotate version 0.53.

Wed Apr 23 00:27:39 2014 - [info] Starting online master switch..

Wed Apr 23 00:27:39 2014 - [info]

Wed Apr 23 00:27:39 2014 - [info] \* Phase 1: Configuration Check Phase..

Wed Apr 23 00:27:39 2014 - [info]

Wed Apr 23 00:27:39 2014 - [info] Reading default configuratoins from /etc/masterha\_default.cnf..

Wed Apr 23 00:27:39 2014 - [info] Reading application default configurations from /etc/masterha/app1.cnf..

Wed Apr 23 00:27:39 2014 - [info] Reading server configurations from /etc/masterha/app1.cnf..

Wed Apr 23 00:27:39 2014 - [info] Multi-master configuration is detected. Current primary(writable) master is 192.168.0.50(192.168.0.50:3306)

Wed Apr 23 00:27:39 2014 - [info] Master configurations are as below:

Master 192.168.0.60(192.168.0.60:3306), replicating from 192.168.0.50(192.168.0.50:3306), read-only

Master 192.168.0.50(192.168.0.50:3306), replicating from 192.168.0.60(192.168.0.60:3306)

Wed Apr 23 00:27:39 2014 - [info] Current Alive Master: 192.168.0.50(192.168.0.50:3306)

Wed Apr 23 00:27:39 2014 - [info] Alive Slaves:

Wed Apr 23 00:27:39 2014 - [info] 192.168.0.60(192.168.0.60:3306) Version=5.5.19-ndb-7.2.4-gpl-log (oldest major version between slaves) log-bin:enabled

Wed Apr 23 00:27:39 2014 - [info] Replicating from 192.168.0.50(192.168.0.50:3306)

Wed Apr 23 00:27:39 2014 - [info] Primary candidate for the new Master (candidate\_master is set)

Wed Apr 23 00:27:39 2014 - [info] 192.168.0.70(192.168.0.70:3306) Version=5.5.19-ndb-7.2.4-gpl-log (oldest major version between slaves) log-bin:enabled

Wed Apr 23 00:27:39 2014 - [info] Replicating from 192.168.0.50(192.168.0.50:3306)

It is better to execute FLUSH NO\_WRITE\_TO\_BINLOG TABLES on the master before switching. Is it ok to execute on 192.168.0.50(192.168.0.50:3306)? (YES/no): yes

Wed Apr 23 00:27:40 2014 - [info] Executing FLUSH NO\_WRITE\_TO\_BINLOG TABLES. This may take long time..

Wed Apr 23 00:27:40 2014 - [info] ok.

Wed Apr 23 00:27:40 2014 - [info] Checking MHA is not monitoring or doing failover..

Wed Apr 23 00:27:40 2014 - [info] Checking replication health on 192.168.0.60..

Wed Apr 23 00:27:40 2014 - [info] ok.

Wed Apr 23 00:27:40 2014 - [info] Checking replication health on 192.168.0.70..

Wed Apr 23 00:27:40 2014 - [info] ok.

Wed Apr 23 00:27:40 2014 - [info] 192.168.0.60 can be new master.

Wed Apr 23 00:27:40 2014 - [info]

From:

192.168.0.50 (current master)

+--192.168.0.60

+--192.168.0.70

To:

192.168.0.60 (new master)

+--192.168.0.70

+--192.168.0.50

Starting master switch from 192.168.0.50(192.168.0.50:3306) to 192.168.0.60(192.168.0.60:3306)? (yes/NO): yes

Wed Apr 23 00:27:41 2014 - [info] Checking whether 192.168.0.60(192.168.0.60:3306) is ok for the new master..

Wed Apr 23 00:27:41 2014 - [info] ok.

Wed Apr 23 00:27:41 2014 - [info] \*\* Phase 1: Configuration Check Phase completed.

Wed Apr 23 00:27:41 2014 - [info]

Wed Apr 23 00:27:41 2014 - [info] \* Phase 2: Rejecting updates Phase..

Wed Apr 23 00:27:41 2014 - [info]

Wed Apr 23 00:27:41 2014 - [info] Executing master ip online change script to disable write on the current master:

Wed Apr 23 00:27:41 2014 - [info] /usr/local/bin/master\_ip\_online\_change.pl --command=stop --orig\_master\_host=192.168.0.50 --orig\_master\_ip=192.168.0.50 --orig\_master\_port=3306 --new\_master\_host=192.168.0.60 --new\_master\_ip=192.168.0.60 --new\_master\_port=3306

Wed Apr 23 00:27:41 2014 714804 Set read\_only on the new master.. ok.

Wed Apr 23 00:27:41 2014 719969 Set read\_only=1 on the orig master.. ok.

Disabling the VIP on old master: 192.168.0.50

reverse mapping checking getaddrinfo for bogon [192.168.0.50] failed - POSSIBLE BREAK-IN ATTEMPT!

Wed Apr 23 00:27:51 2014 963762 Killing all application threads..

Wed Apr 23 00:27:51 2014 963869 done.

Wed Apr 23 00:27:51 2014 - [info] ok.

Wed Apr 23 00:27:51 2014 - [info] Locking all tables on the orig master to reject updates from everybody (including root):

Wed Apr 23 00:27:51 2014 - [info] Executing FLUSH TABLES WITH READ LOCK..

Wed Apr 23 00:27:51 2014 - [info] ok.

Wed Apr 23 00:27:51 2014 - [info] Orig master binlog:pos is mysql-bin.000028:112.

Wed Apr 23 00:27:51 2014 - [info] Waiting to execute all relay logs on 192.168.0.60(192.168.0.60:3306)..

Wed Apr 23 00:27:51 2014 - [info] master\_pos\_wait(mysql-bin.000028:112) completed on 192.168.0.60(192.168.0.60:3306). Executed 0 events.

Wed Apr 23 00:27:51 2014 - [info] done.

Wed Apr 23 00:27:51 2014 - [info] Getting new master's binlog name and position..

Wed Apr 23 00:27:51 2014 - [info] mysql-bin.000023:1550

Wed Apr 23 00:27:51 2014 - [info] All other slaves should start replication from here. Statement should be: CHANGE MASTER TO MASTER\_HOST='192.168.0.60', MASTER\_PORT=3306, MASTER\_LOG\_FILE='mysql-bin.000023', MASTER\_LOG\_POS=1550, MASTER\_USER='repl', MASTER\_PASSWORD='xxx';

Wed Apr 23 00:27:51 2014 - [info] Executing master ip online change script to allow write on the new master:

Wed Apr 23 00:27:51 2014 - [info] /usr/local/bin/master\_ip\_online\_change.pl --command=start --orig\_master\_host=192.168.0.50 --orig\_master\_ip=192.168.0.50 --orig\_master\_port=3306 --new\_master\_host=192.168.0.60 --new\_master\_ip=192.168.0.60 --new\_master\_port=3306

Wed Apr 23 00:27:52 2014 077334 Set read\_only=0 on the new master.

Enabling the VIP - 192.168.0.88/24 on the new master - 192.168.0.60

reverse mapping checking getaddrinfo for bogon [192.168.0.60] failed - POSSIBLE BREAK-IN ATTEMPT!

Wed Apr 23 00:28:02 2014 - [info] ok.

Wed Apr 23 00:28:02 2014 - [info]

Wed Apr 23 00:28:02 2014 - [info] \* Switching slaves in parallel..

Wed Apr 23 00:28:02 2014 - [info]

Wed Apr 23 00:28:02 2014 - [info] -- Slave switch on host 192.168.0.70(192.168.0.70:3306) started, pid: 3036

Wed Apr 23 00:28:02 2014 - [info]

Wed Apr 23 00:28:02 2014 - [info] Log messages from 192.168.0.70 ...

Wed Apr 23 00:28:02 2014 - [info]

Wed Apr 23 00:28:02 2014 - [info] Waiting to execute all relay logs on 192.168.0.70(192.168.0.70:3306)..

Wed Apr 23 00:28:02 2014 - [info] master\_pos\_wait(mysql-bin.000028:112) completed on 192.168.0.70(192.168.0.70:3306). Executed 0 events.

Wed Apr 23 00:28:02 2014 - [info] done.

Wed Apr 23 00:28:02 2014 - [info] Resetting slave 192.168.0.70(192.168.0.70:3306) and starting replication from the new master 192.168.0.60(192.168.0.60:3306)..

Wed Apr 23 00:28:02 2014 - [info] Executed CHANGE MASTER.

Wed Apr 23 00:28:02 2014 - [info] Slave started.

Wed Apr 23 00:28:02 2014 - [info] End of log messages from 192.168.0.70 ...

Wed Apr 23 00:28:02 2014 - [info]

Wed Apr 23 00:28:02 2014 - [info] -- Slave switch on host 192.168.0.70(192.168.0.70:3306) succeeded.

Wed Apr 23 00:28:02 2014 - [info] Unlocking all tables on the orig master:

Wed Apr 23 00:28:02 2014 - [info] Executing UNLOCK TABLES..

Wed Apr 23 00:28:02 2014 - [info] ok.

Wed Apr 23 00:28:02 2014 - [info] Starting orig master as a new slave..

Wed Apr 23 00:28:02 2014 - [info] Resetting slave 192.168.0.50(192.168.0.50:3306) and starting replication from the new master 192.168.0.60(192.168.0.60:3306)..

Wed Apr 23 00:28:02 2014 - [info] Executed CHANGE MASTER.

Wed Apr 23 00:28:02 2014 - [info] Slave started.

Wed Apr 23 00:28:02 2014 - [info] All new slave servers switched successfully.

Wed Apr 23 00:28:02 2014 - [info]

Wed Apr 23 00:28:02 2014 - [info] \* Phase 5: New master cleanup phease..

Wed Apr 23 00:28:02 2014 - [info]

Wed Apr 23 00:28:02 2014 - [info] 192.168.0.60: Resetting slave info succeeded.

Wed Apr 23 00:28:02 2014 - [info] Switching master to 192.168.0.60(192.168.0.60:3306) completed successfully.

[复制代码](javascript:void(0);)

**其中参数的意思：**

--orig\_master\_is\_new\_slave 切换时加上此参数是将原 master 变为 slave 节点，如果不加此参数，原来的 master 将不启动

--running\_updates\_limit=10000,故障切换时,候选master 如果有延迟的话， mha 切换不能成功，加上此参数表示延迟在此时间范围内都可切换（单位为s），但是切换的时间长短是由recover 时relay 日志的大小决定

**注意：由于在线进行切换需要调用到master\_ip\_online\_change这个脚本，但是由于该脚本不完整，需要自己进行相应的修改，我google到后发现还是有问题，脚本中new\_master\_password这个变量获取不到，导致在线切换失败，所以进行了相关的硬编码，直接把mysql的root用户密码赋值给变量new\_master\_password，如果有哪位大牛知道原因，请指点指点。这个脚本还可以管理vip。下面贴出脚本：**

https://images.cnblogs.com/OutliningIndicators/ContractedBlock.gifhttps://images.cnblogs.com/OutliningIndicators/ExpandedBlockStart.gif

#!/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 script based on your environment.

use strict;

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

use Getopt::Long;

use MHA::DBHelper;

use MHA::NodeUtil;

use Time::HiRes qw( sleep gettimeofday tv\_interval );

use Data::Dumper;

my $\_tstart;

my $\_running\_interval = 0.1;

my (

$command, $orig\_master\_host, $orig\_master\_ip,

$orig\_master\_port, $orig\_master\_user,

$new\_master\_host, $new\_master\_ip, $new\_master\_port,

$new\_master\_user,

);

my $vip = **'192.168.0.88/24'**; # Virtual IP

my $key = **"1"**;

my $ssh\_start\_vip = **"/sbin/ifconfig eth1:$key $vip"**;

my $ssh\_stop\_vip = **"/sbin/ifconfig eth1:$key down"**;

my $ssh\_user = **"root"**;

my $new\_master\_password=**'123456'**;

my $orig\_master\_password=**'123456'**;

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,

**'orig\_master\_user=s'** => \$orig\_master\_user,

#'orig\_master\_password=s' => \$orig\_master\_password,

**'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 current\_time\_us {

my ( $sec, $microsec ) = gettimeofday();

my $curdate = localtime($sec);

return $curdate . **"** **"** . sprintf( **"%06d"**, $microsec );

}

sub sleep\_until {

my $elapsed = tv\_interval($\_tstart);

if ( $\_running\_interval > $elapsed ) {

sleep( $\_running\_interval - $elapsed );

}

}

sub get\_threads\_util {

my $dbh = shift;

my $my\_connection\_id = shift;

my $running\_time\_threshold = shift;

my $type = shift;

$running\_time\_threshold = 0 unless ($running\_time\_threshold);

$type = 0 unless ($type);

my @threads;

my $sth = $dbh->prepare(**"SHOW PROCESSLIST"**);

$sth->execute();

while ( my $ref = $sth->fetchrow\_hashref() ) {

my $id = $ref->{Id};

my $user = $ref->{User};

my $host = $ref->{Host};

my $command = $ref->{Command};

my $state = $ref->{State};

my $query\_time = $ref->{Time};

my $info = $ref->{Info};

$info =~ s/^\s\*(.\*?)\s\*$/$1/ if defined($info);

next if ( $my\_connection\_id == $id );

next if ( defined($query\_time) && $query\_time < $running\_time\_threshold );

next if ( defined($command) && $command eq **"Binlog Dump"** );

next if ( defined($user) && $user eq **"system user"** );

next

if ( defined($command)

&& $command eq **"Sleep"**

&& defined($query\_time)

&& $query\_time >= 1 );

if ( $type >= 1 ) {

next if ( defined($command) && $command eq **"Sleep"** );

next if ( defined($command) && $command eq **"Connect"** );

}

if ( $type >= 2 ) {

next if ( defined($info) && $info =~ m/^select/i );

next if ( defined($info) && $info =~ m/^show/i );

}

push @threads, $ref;

}

return @threads;

}

sub main {

if ( $command eq **"stop"** ) {

## Gracefully killing connections on the current master

# 1. Set read\_only= 1 on the new master

# 2. DROP USER so that no app user can establish new connections

# 3. Set read\_only= 1 on the current master

# 4. Kill current queries

# \* Any database access failure will result in script die.

my $exit\_code = 1;

eval {

## Setting read\_only=1 on the new master (to avoid accident)

my $new\_master\_handler = new MHA::DBHelper();

# args: hostname, port, user, password, raise\_error(die\_on\_error)\_or\_not

$new\_master\_handler->connect( $new\_master\_ip, $new\_master\_port,

$new\_master\_user, $new\_master\_password, 1 );

print current\_time\_us() . **" Set read\_only on the new master.. "**;

$new\_master\_handler->enable\_read\_only();

if ( $new\_master\_handler->is\_read\_only() ) {

print **"ok.\n"**;

}

else {

die **"Failed!\n"**;

}

$new\_master\_handler->disconnect();

# Connecting to the orig master, die if any database error happens

my $orig\_master\_handler = new MHA::DBHelper();

$orig\_master\_handler->connect( $orig\_master\_ip, $orig\_master\_port,

$orig\_master\_user, $orig\_master\_password, 1 );

## Drop application user so that nobody can connect. Disabling per-session binlog beforehand

#$orig\_master\_handler->disable\_log\_bin\_local();

#print current\_time\_us() . " Drpping app user on the orig master..\n";

#FIXME\_xxx\_drop\_app\_user($orig\_master\_handler);

## Waiting for N \* 100 milliseconds so that current connections can exit

my $time\_until\_read\_only = 15;

$\_tstart = [gettimeofday];

my @threads = get\_threads\_util( $orig\_master\_handler->{dbh},

$orig\_master\_handler->{connection\_id} );

while ( $time\_until\_read\_only > 0 && $#threads >= 0 ) {

if ( $time\_until\_read\_only % 5 == 0 ) {

printf

**"%s Waiting all running %d threads are disconnected.. (max %d milliseconds)\n"**,

current\_time\_us(), $#threads + 1, $time\_until\_read\_only \* 100;

if ( $#threads < 5 ) {

print Data::Dumper->new( [$\_] )->Indent(0)->Terse(1)->Dump . **"\n"**

foreach (@threads);

}

}

sleep\_until();

$\_tstart = [gettimeofday];

$time\_until\_read\_only--;

@threads = get\_threads\_util( $orig\_master\_handler->{dbh},

$orig\_master\_handler->{connection\_id} );

}

## Setting read\_only=1 on the current master so that nobody(except SUPER) can write

print current\_time\_us() . **" Set read\_only=1 on the orig master.. "**;

$orig\_master\_handler->enable\_read\_only();

if ( $orig\_master\_handler->is\_read\_only() ) {

print **"ok.\n"**;

}

else {

die **"Failed!\n"**;

}

## Waiting for M \* 100 milliseconds so that current update queries can complete

my $time\_until\_kill\_threads = 5;

@threads = get\_threads\_util( $orig\_master\_handler->{dbh},

$orig\_master\_handler->{connection\_id} );

while ( $time\_until\_kill\_threads > 0 && $#threads >= 0 ) {

if ( $time\_until\_kill\_threads % 5 == 0 ) {

printf

**"%s Waiting all running %d queries are disconnected.. (max %d milliseconds)\n"**,

current\_time\_us(), $#threads + 1, $time\_until\_kill\_threads \* 100;

if ( $#threads < 5 ) {

print Data::Dumper->new( [$\_] )->Indent(0)->Terse(1)->Dump . **"\n"**

foreach (@threads);

}

}

sleep\_until();

$\_tstart = [gettimeofday];

$time\_until\_kill\_threads--;

@threads = get\_threads\_util( $orig\_master\_handler->{dbh},

$orig\_master\_handler->{connection\_id} );

}

print **"Disabling the VIP on old master: $orig\_master\_host \n"**;

&stop\_vip();

## Terminating all threads

print current\_time\_us() . **" Killing all application threads..\n"**;

$orig\_master\_handler->kill\_threads(@threads) if ( $#threads >= 0 );

print current\_time\_us() . **" done.\n"**;

#$orig\_master\_handler->enable\_log\_bin\_local();

$orig\_master\_handler->disconnect();

## After finishing the script, MHA executes FLUSH TABLES WITH READ LOCK

$exit\_code = 0;

};

if ($@) {

warn **"Got Error: $@\n"**;

exit $exit\_code;

}

exit $exit\_code;

}

elsif ( $command eq **"start"** ) {

## Activating master ip on the new master

# 1. Create app user with write privileges

# 2. Moving backup script if needed

# 3. Register new master's ip to the catalog database

# We don't return error even though activating updatable accounts/ip failed so that we don't interrupt slaves' recovery.

# If exit code is 0 or 10, MHA does not abort

my $exit\_code = 10;

eval {

my $new\_master\_handler = new MHA::DBHelper();

# args: hostname, port, user, password, raise\_error\_or\_not

$new\_master\_handler->connect( $new\_master\_ip, $new\_master\_port,

$new\_master\_user, $new\_master\_password, 1 );

## Set read\_only=0 on the new master

#$new\_master\_handler->disable\_log\_bin\_local();

print current\_time\_us() . **" Set read\_only=0 on the new master.\n"**;

$new\_master\_handler->disable\_read\_only();

## Creating an app user on the new master

#print current\_time\_us() . " Creating app user on the new master..\n";

#FIXME\_xxx\_create\_app\_user($new\_master\_handler);

#$new\_master\_handler->enable\_log\_bin\_local();

$new\_master\_handler->disconnect();

## Update master ip on the catalog database, etc

print **"Enabling the VIP - $vip on the new master - $new\_master\_host \n"**;

&start\_vip();

$exit\_code = 0;

};

if ($@) {

warn **"Got Error: $@\n"**;

exit $exit\_code;

}

exit $exit\_code;

}

elsif ( $command eq **"status"** ) {

# do nothing

exit 0;

}

else {

&usage();

exit 1;

}

}

# A simple system call that enable the VIP on the new master

sub start\_vip() {

`ssh $ssh\_user\@$new\_master\_host \**" $ssh\_start\_vip \"`;**

**}**

**# A simple system call that disable the VIP on the old\_master**

**sub stop\_vip() {**

**`ssh $ssh\_user\@$orig\_master\_host \" $ssh\_stop\_vip \"`;**

**}**

**sub usage {**

**print**

**"**Usage: master\_ip\_online\_change --command=start|stop|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**";**

**die;**

**}**

[复制代码](javascript:void(0);)

#!/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 script based on your environment.

use strict;

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

use Getopt::Long;

use MHA::DBHelper;

use MHA::NodeUtil;

use Time::HiRes qw( sleep gettimeofday tv\_interval );

use Data::Dumper;

my $\_tstart;

my $\_running\_interval = 0.1;

my (

$command, $orig\_master\_host, $orig\_master\_ip,

$orig\_master\_port, $orig\_master\_user,

$new\_master\_host, $new\_master\_ip, $new\_master\_port,

$new\_master\_user,

);

my $vip = **'192.168.0.88/24'**; # Virtual IP

my $key = **"1"**;

my $ssh\_start\_vip = **"/sbin/ifconfig eth1:$key $vip"**;

my $ssh\_stop\_vip = **"/sbin/ifconfig eth1:$key down"**;

my $ssh\_user = **"root"**;

my $new\_master\_password=**'123456'**;

my $orig\_master\_password=**'123456'**;

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,

**'orig\_master\_user=s'** => \$orig\_master\_user,

#'orig\_master\_password=s' => \$orig\_master\_password,

**'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 current\_time\_us {

my ( $sec, $microsec ) = gettimeofday();

my $curdate = localtime($sec);

return $curdate . **"** **"** . sprintf( **"%06d"**, $microsec );

}

sub sleep\_until {

my $elapsed = tv\_interval($\_tstart);

if ( $\_running\_interval > $elapsed ) {

sleep( $\_running\_interval - $elapsed );

}

}

sub get\_threads\_util {

my $dbh = shift;

my $my\_connection\_id = shift;

my $running\_time\_threshold = shift;

my $type = shift;

$running\_time\_threshold = 0 unless ($running\_time\_threshold);

$type = 0 unless ($type);

my @threads;

my $sth = $dbh->prepare(**"SHOW PROCESSLIST"**);

$sth->execute();

while ( my $ref = $sth->fetchrow\_hashref() ) {

my $id = $ref->{Id};

my $user = $ref->{User};

my $host = $ref->{Host};

my $command = $ref->{Command};

my $state = $ref->{State};

my $query\_time = $ref->{Time};

my $info = $ref->{Info};

$info =~ s/^\s\*(.\*?)\s\*$/$1/ if defined($info);

next if ( $my\_connection\_id == $id );

next if ( defined($query\_time) && $query\_time < $running\_time\_threshold );

next if ( defined($command) && $command eq **"Binlog Dump"** );

next if ( defined($user) && $user eq **"system user"** );

next

if ( defined($command)

&& $command eq **"Sleep"**

&& defined($query\_time)

&& $query\_time >= 1 );

if ( $type >= 1 ) {

next if ( defined($command) && $command eq **"Sleep"** );

next if ( defined($command) && $command eq **"Connect"** );

}

if ( $type >= 2 ) {

next if ( defined($info) && $info =~ m/^select/i );

next if ( defined($info) && $info =~ m/^show/i );

}

push @threads, $ref;

}

return @threads;

}

sub main {

if ( $command eq **"stop"** ) {

## Gracefully killing connections on the current master

# 1. Set read\_only= 1 on the new master

# 2. DROP USER so that no app user can establish new connections

# 3. Set read\_only= 1 on the current master

# 4. Kill current queries

# \* Any database access failure will result in script die.

my $exit\_code = 1;

eval {

## Setting read\_only=1 on the new master (to avoid accident)

my $new\_master\_handler = new MHA::DBHelper();

# args: hostname, port, user, password, raise\_error(die\_on\_error)\_or\_not

$new\_master\_handler->connect( $new\_master\_ip, $new\_master\_port,

$new\_master\_user, $new\_master\_password, 1 );

print current\_time\_us() . **" Set read\_only on the new master.. "**;

$new\_master\_handler->enable\_read\_only();

if ( $new\_master\_handler->is\_read\_only() ) {

print **"ok.\n"**;

}

else {

die **"Failed!\n"**;

}

$new\_master\_handler->disconnect();

# Connecting to the orig master, die if any database error happens

my $orig\_master\_handler = new MHA::DBHelper();

$orig\_master\_handler->connect( $orig\_master\_ip, $orig\_master\_port,

$orig\_master\_user, $orig\_master\_password, 1 );

## Drop application user so that nobody can connect. Disabling per-session binlog beforehand

#$orig\_master\_handler->disable\_log\_bin\_local();

#print current\_time\_us() . " Drpping app user on the orig master..\n";

#FIXME\_xxx\_drop\_app\_user($orig\_master\_handler);

## Waiting for N \* 100 milliseconds so that current connections can exit

my $time\_until\_read\_only = 15;

$\_tstart = [gettimeofday];

my @threads = get\_threads\_util( $orig\_master\_handler->{dbh},

$orig\_master\_handler->{connection\_id} );

while ( $time\_until\_read\_only > 0 && $#threads >= 0 ) {

if ( $time\_until\_read\_only % 5 == 0 ) {

printf

**"%s Waiting all running %d threads are disconnected.. (max %d milliseconds)\n"**,

current\_time\_us(), $#threads + 1, $time\_until\_read\_only \* 100;

if ( $#threads < 5 ) {

print Data::Dumper->new( [$\_] )->Indent(0)->Terse(1)->Dump . **"\n"**

foreach (@threads);

}

}

sleep\_until();

$\_tstart = [gettimeofday];

$time\_until\_read\_only--;

@threads = get\_threads\_util( $orig\_master\_handler->{dbh},

$orig\_master\_handler->{connection\_id} );

}

## Setting read\_only=1 on the current master so that nobody(except SUPER) can write

print current\_time\_us() . **" Set read\_only=1 on the orig master.. "**;

$orig\_master\_handler->enable\_read\_only();

if ( $orig\_master\_handler->is\_read\_only() ) {

print **"ok.\n"**;

}

else {

die **"Failed!\n"**;

}

## Waiting for M \* 100 milliseconds so that current update queries can complete

my $time\_until\_kill\_threads = 5;

@threads = get\_threads\_util( $orig\_master\_handler->{dbh},

$orig\_master\_handler->{connection\_id} );

while ( $time\_until\_kill\_threads > 0 && $#threads >= 0 ) {

if ( $time\_until\_kill\_threads % 5 == 0 ) {

printf

**"%s Waiting all running %d queries are disconnected.. (max %d milliseconds)\n"**,

current\_time\_us(), $#threads + 1, $time\_until\_kill\_threads \* 100;

if ( $#threads < 5 ) {

print Data::Dumper->new( [$\_] )->Indent(0)->Terse(1)->Dump . **"\n"**

foreach (@threads);

}

}

sleep\_until();

$\_tstart = [gettimeofday];

$time\_until\_kill\_threads--;

@threads = get\_threads\_util( $orig\_master\_handler->{dbh},

$orig\_master\_handler->{connection\_id} );

}

print **"Disabling the VIP on old master: $orig\_master\_host \n"**;

&stop\_vip();

## Terminating all threads

print current\_time\_us() . **" Killing all application threads..\n"**;

$orig\_master\_handler->kill\_threads(@threads) if ( $#threads >= 0 );

print current\_time\_us() . **" done.\n"**;

#$orig\_master\_handler->enable\_log\_bin\_local();

$orig\_master\_handler->disconnect();

## After finishing the script, MHA executes FLUSH TABLES WITH READ LOCK

$exit\_code = 0;

};

if ($@) {

warn **"Got Error: $@\n"**;

exit $exit\_code;

}

exit $exit\_code;

}

elsif ( $command eq **"start"** ) {

## Activating master ip on the new master

# 1. Create app user with write privileges

# 2. Moving backup script if needed

# 3. Register new master's ip to the catalog database

# We don't return error even though activating updatable accounts/ip failed so that we don't interrupt slaves' recovery.

# If exit code is 0 or 10, MHA does not abort

my $exit\_code = 10;

eval {

my $new\_master\_handler = new MHA::DBHelper();

# args: hostname, port, user, password, raise\_error\_or\_not

$new\_master\_handler->connect( $new\_master\_ip, $new\_master\_port,

$new\_master\_user, $new\_master\_password, 1 );

## Set read\_only=0 on the new master

#$new\_master\_handler->disable\_log\_bin\_local();

print current\_time\_us() . **" Set read\_only=0 on the new master.\n"**;

$new\_master\_handler->disable\_read\_only();

## Creating an app user on the new master

#print current\_time\_us() . " Creating app user on the new master..\n";

#FIXME\_xxx\_create\_app\_user($new\_master\_handler);

#$new\_master\_handler->enable\_log\_bin\_local();

$new\_master\_handler->disconnect();

## Update master ip on the catalog database, etc

print **"Enabling the VIP - $vip on the new master - $new\_master\_host \n"**;

&start\_vip();

$exit\_code = 0;

};

if ($@) {

warn **"Got Error: $@\n"**;

exit $exit\_code;

}

exit $exit\_code;

}

elsif ( $command eq **"status"** ) {

# do nothing

exit 0;

}

else {

&usage();

exit 1;

}

}

# A simple system call that enable the VIP on the new master

sub start\_vip() {

`ssh $ssh\_user\@$new\_master\_host \**" $ssh\_start\_vip \"`;**

**}**

**# A simple system call that disable the VIP on the old\_master**

**sub stop\_vip() {**

**`ssh $ssh\_user\@$orig\_master\_host \" $ssh\_stop\_vip \"`;**

**}**

**sub usage {**

**print**

**"**Usage: master\_ip\_online\_change --command=start|stop|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**";**

**die;**

**}**

[复制代码](javascript:void(0);)

**四.修复宕机的Master**

通常情况下自动切换以后，原master可能已经废弃掉，待原master主机修复后，如果数据完整的情况下，可能想把原来master重新作为新主库的slave，这时我们可以借助当时自动切换时刻的MHA日志来完成对原master的修复。下面是提取相关日志的命令：

[root@192.168.0.20 app1]# grep -i "All other slaves should start" manager.log

Mon Apr 21 22:28:33 2014 - [info] All other slaves should start replication from here. Statement should be: CHANGE MASTER TO MASTER\_HOST='192.168.0.60', MASTER\_PORT=3306, MASTER\_LOG\_FILE='mysql-bin.000022', MASTER\_LOG\_POS=506716, MASTER\_USER='repl', MASTER\_PASSWORD='xxx';

[root@192.168.0.20 app1]#

获取上述信息以后，就可以直接在修复后的master上执行change master to相关操作，重新作为从库了。

最后补充一下邮件发送脚本send\_report ，这个脚本在询问一位朋友后可以使用，如下：

#!/usr/bin/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 script based on your environment.

use strict;

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

use Mail::Sender;

use Getopt::Long;

#new\_master\_host and new\_slave\_hosts are set only when recovering master succeeded

my ( $dead\_master\_host, $new\_master\_host, $new\_slave\_hosts, $subject, $body );

my $smtp=**'smtp.163.com'**;

my $mail\_from=**'xxxx'**;

my $mail\_user=**'xxxxx'**;

my $mail\_pass=**'xxxxx'**;

my $mail\_to=[**'xxxx'**,**'xxxx'**];

GetOptions(

**'orig\_master\_host=s'** => \$dead\_master\_host,

**'new\_master\_host=s'** => \$new\_master\_host,

**'new\_slave\_hosts=s'** => \$new\_slave\_hosts,

**'subject=s'** => \$subject,

**'body=s'** => \$body,

);

mailToContacts($smtp,$mail\_from,$mail\_user,$mail\_pass,$mail\_to,$subject,$body);

sub mailToContacts {

my ( $smtp, $mail\_from, $user, $passwd, $mail\_to, $subject, $msg ) = @\_;

open my $DEBUG, **"> /tmp/monitormail.log"**

or die **"Can't open the debug file:$!\n"**;

my $sender = new Mail::Sender {

ctype => **'text/plain; charset=utf-8'**,

encoding => **'utf-8'**,

smtp => $smtp,

from => $mail\_from,

auth => **'LOGIN'**,

TLS\_allowed => **'0'**,

authid => $user,

authpwd => $passwd,

to => $mail\_to,

subject => $subject,

debug => $DEBUG

};

$sender->MailMsg(

{ msg => $msg,

debug => $DEBUG

}

) or print $Mail::Sender::Error;

return 1;

}

# Do whatever you want here

exit 0;

[复制代码](javascript:void(0);)

[复制代码](javascript:void(0);)

#!/usr/bin/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 script based on your environment.

use strict;

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

use Mail::Sender;

use Getopt::Long;

#new\_master\_host and new\_slave\_hosts are set only when recovering master succeeded

my ( $dead\_master\_host, $new\_master\_host, $new\_slave\_hosts, $subject, $body );

my $smtp=**'smtp.163.com'**;

my $mail\_from=**'xxxx'**;

my $mail\_user=**'xxxxx'**;

my $mail\_pass=**'xxxxx'**;

my $mail\_to=[**'xxxx'**,**'xxxx'**];

GetOptions(

**'orig\_master\_host=s'** => \$dead\_master\_host,

**'new\_master\_host=s'** => \$new\_master\_host,

**'new\_slave\_hosts=s'** => \$new\_slave\_hosts,

**'subject=s'** => \$subject,

**'body=s'** => \$body,

);

mailToContacts($smtp,$mail\_from,$mail\_user,$mail\_pass,$mail\_to,$subject,$body);

sub mailToContacts {

my ( $smtp, $mail\_from, $user, $passwd, $mail\_to, $subject, $msg ) = @\_;

open my $DEBUG, **"> /tmp/monitormail.log"**

or die **"Can't open the debug file:$!\n"**;

my $sender = new Mail::Sender {

ctype => **'text/plain; charset=utf-8'**,

encoding => **'utf-8'**,

smtp => $smtp,

from => $mail\_from,

auth => **'LOGIN'**,

TLS\_allowed => **'0'**,

authid => $user,

authpwd => $passwd,

to => $mail\_to,

subject => $subject,

debug => $DEBUG

};

$sender->MailMsg(

{ msg => $msg,

debug => $DEBUG

}

) or print $Mail::Sender::Error;

return 1;

}

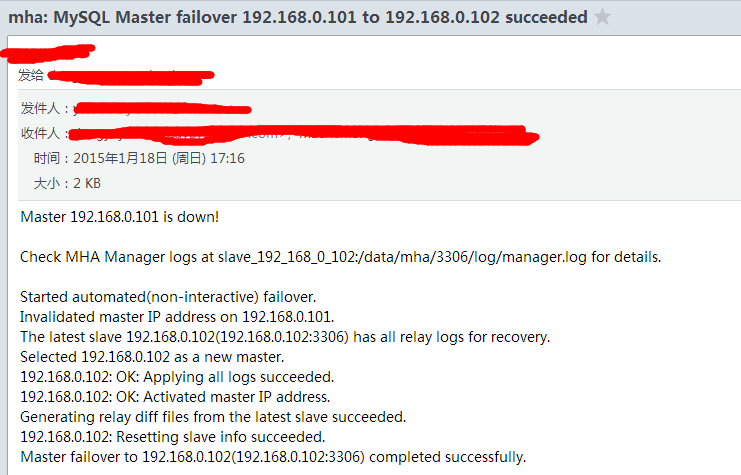
# Do whatever you want here

exit 0;

[复制代码](javascript:void(0);)

[复制代码](javascript:void(0);)

最后切换以后发送告警的邮件示例，注意，这个是我后续的测试，和上面环境出现的ip不一致不要在意。



## 总结

目前高可用方案可以一定程度上实现数据库的高可用，比如前面文章介绍的[MMM](http://www.cnblogs.com/gomysql/p/3671896.html)，[heartbeat+drbd](http://www.cnblogs.com/gomysql/p/3674030.html)，[Cluster](http://www.cnblogs.com/gomysql/p/3664783.html)等。还有percona的Galera Cluster等。这些高可用软件各有优劣。在进行高可用方案选择时，主要是看业务还有对数据一致性方面的要求。最后出于对数据库的高可用和数据一致性的要求，推荐使用MHA架构。

# [MMM](https://www.cnblogs.com/gomysql/p/3671896.html)

## **简介**

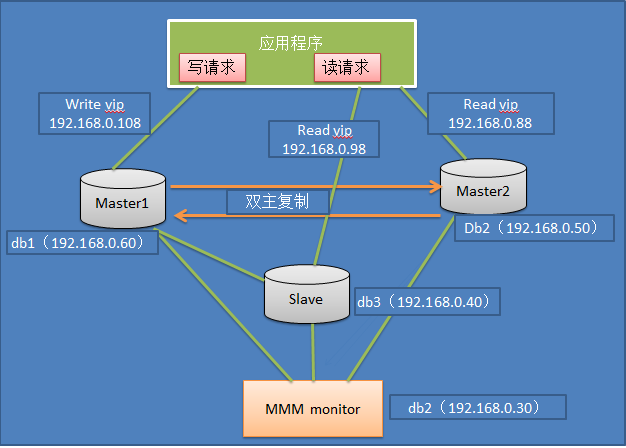
MMM（Master-Master replication manager for MySQL）是一套支持双主故障切换和双主日常管理的脚本程序。MMM使用Perl语言开发，主要用来监控和管理MySQL Master-Master（双主）复制，虽然叫做双主复制，但是业务上同一时刻只允许对一个主进行写入，另一台备选主上提供部分读服务，以加速在主主切换时刻备选主的预热，可以说MMM这套脚本程序一方面实现了故障切换的功能，另一方面其内部附加的工具脚本也可以实现多个slave的read负载均衡。

MMM提供了自动和手动两种方式移除一组服务器中复制延迟较高的服务器的虚拟ip，同时它还可以备份数据，实现两节点之间的数据同步等。由于MMM无法完全的保证数据一致性，所以MMM适用于对数据的一致性要求不是很高，但是又想最大程度的保证业务可用性的场景。对于那些对数据的一致性要求很高的业务，非常不建议采用MMM这种高可用架构。

MMM项目来自 Google：<http://code.google.com/p/mysql-master-master>

官方网站为：[http://mysql-mmm.org](http://mysql-mmm.org/)

下面我们通过一个实际案例来充分了解MMM的内部架构，如下图所示。



具体的配置信息如下所示：

角色 ip地址 主机名字 server-id

monitoring 192.168.0.30 db2 -

master1 192.168.0.60 db1 1

master2 192.168.0.50 db2 2

slave1 192.168.0.40 db3 3

业务中的服务ip信息如下所示：

ip地址 角色 描述

192.168.0.108 write 应用程序连接该ip对主库进行写请求

192.168.0.88 read 应用程序连接该ip进行读请求

192.168.0.98 read 应用程序连接该ip进行读请求

## 搭建

具体的配置步骤如下：

### （1）主机配置

配置/etc/hosts,在所有主机中，添加所有的主机信息：

[root@192.168.0.30 ~]# cat /etc/hosts

192.168.0.60 db1

192.168.0.50 db2

192.168.0.40 db3

[root@192.168.0.30 ~]#

### （2）首先在3台主机上安装mysql和搭建复制

（192.168.0.60和192.168.0.50互为主从，192.168.0.40为192.168.0.60的从）具体的复制搭建这里就省略，要是这都不会，那么该文章对你就没意思了。然后在每个mysql的配置文件中加入以下内容，**注意server\_id 不能重复。**

db1（192.168.0.60）上：

server-id = 1

log\_slave\_updates = 1

auto-increment-increment = 2

auto-increment-offset = 1

db2（192.168.0.50）上：

server-id = 2

log\_slave\_updates = 1

auto-increment-increment = 2

auto-increment-offset = 2

db3（192.168.0.40）上：

server-id = 3

log\_slave\_updates = 1

上面的id不一定要按顺序来，只要没有重复即可。

### （3）安装MMM所需要的Perl模块

（所有服务器）执行该脚本，也可以安装epel源，然后yum -y install mysql-mmm\*来安装MMM：

rpm -ivh http://dl.fedoraproject.org/pub/epel/6/x86\_64/epel-release-6-8.noarch.rpm

yum -y install mysql-mmm\*

[复制代码](javascript:void(0);)

[root@192.168.0.60 ~]# cat install.sh

#!/bin/bash

wget http://xrl.us/cpanm --no-check-certificate

mv cpanm /usr/bin

chmod 755 /usr/bin/cpanm

cat > /root/list << EOF

install Algorithm::Diff

install Class::Singleton

install DBI

install DBD::mysql

install File::Basename

install File::stat

install File::Temp

install Log::Dispatch

install Log::Log4perl

install Mail::Send

install Net::ARP

install Net::Ping

install Proc::Daemon

install Thread::Queue

install Time::HiRes

EOF

for package in `cat /root/list`

do

cpanm $package

done

[root@192.168.0.60 ~]#

[复制代码](javascript:void(0);)

### （4）下载mysql-mmm软件，在所有服务器上安装

[root@192.168.0.60 ~]# wget http://mysql-mmm.org/\_media/:mmm2:mysql-mmm-2.2.1.tar.gz

[root@192.168.0.60 ~]# mv :mmm2:mysql-mmm-2.2.1.tar.gz mysql-mmm-2.2.1.tar.gz

[root@192.168.0.60 ~]# tar xf mysql-mmm-2.2.1.tar.gz

[root@192.168.0.60 ~]# cd mysql-mmm-2.2.1

[root@192.168.0.60 mysql-mmm-2.2.1]# make install

mysql-mmm安装后的主要拓扑结构如下所示（注意：yum安装的和源码安装的路径有所区别）：

[复制代码](javascript:void(0);)

目录 介绍

/usr/lib/perl5/vendor\_perl/5.8.8/MMM MMM使用的主要perl模块

/usr/lib/mysql-mmm MMM使用的主要脚本

/usr/sbin MMM使用的主要命令的路径

/etc/init.d/ MMM的agent和monitor启动服务的目录

/etc/mysql-mmm MMM配置文件的路径，默认所以的配置文件位于该目录下

/var/log/mysql-mmm 默认的MMM保存日志的位置

[复制代码](javascript:void(0);)

到这里已经完成了MMM的基本需求，接下来需要配置具体的配置文件，其中mmm\_common.conf，mmm\_agent.conf为agent端的配置文件，mmm\_mon.conf为monitor端的配置文件。

### （5）配置agent端的配置文件，需要在db1，db2，db3上分别配置。

在db1主机上配置agent配置文件：

[复制代码](javascript:void(0);)

[root@192.168.0.60 ~]# cd /etc/mysql-mmm/

[root@192.168.0.60 mysql-mmm]# cat mmm\_common.conf

active\_master\_role writer

<host default>

cluster\_interface eth1

pid\_path /var/run/mmm\_agentd.pid

bin\_path /usr/lib/mysql-mmm/

**replication\_user repl**

**replication\_password 123456**

**agent\_user mmm\_agent**

**agent\_password mmm\_agent**

</host>

<host db1>

ip 192.168.0.60

**mode master**

peer db2

</host>

<host db2>

ip 192.168.0.50

**mode master**

peer db1

</host>

<host db3>

ip 192.168.0.40

**mode slave**

</host>

<role writer>

hosts db1, db2

**ips 192.168.0.108**

**mode exclusive**

</role>

<role reader>

hosts db2, db3

**ips 192.168.0.88, 192.168.0.98**

mode balanced

</role>

[root@192.168.0.60 mysql-mmm]#

[复制代码](javascript:void(0);)

其中**replication\_user**用于检查复制的用户，**agent\_user**为agent的用户，**mode**标明是否为主或者备选主，或者从库。**mode exclusive**主为独占模式，同一时刻只能有一个主，**<role write>**中hosts表示目前的主库和备选主的真实主机ip或者主机名，**ips**为对外提供的虚拟机ip地址**，<role readr>**中hosts代表从库真实的ip和主机名，**ips**代表从库的虚拟ip地址。

由于db2和db3两台主机也要配置agent配置文件，我们直接把mmm\_common.conf从db1拷贝到db2和db3两台主机的/etc/mysql-mmm下。

**注意：monitor主机要需要：**

scp /etc/mysql-mmm/mmm\_common.conf db2:/etc/mysql-mmm/

scp /etc/mysql-mmm/mmm\_common.conf db3:/etc/mysql-mmm/

分别在db1，db2，db3三台主机的/etc/mysql-mmm配置mmm\_agent.conf文件，分别用不同的字符标识，注意这三台机器的this db1这块要想，比如本环境中，db1要配置this db1，db2要配置为this db2，而db3要配置为this db3。

在db1（192.168.0.60）上：

[root@192.168.0.60 ~]# cat /etc/mysql-mmm/mmm\_agent.conf

include mmm\_common.conf

this db1

[root@192.168.0.60 ~]#

在db2（192.168.0.50）上：

[root@192.168.0.50 ~]# cat /etc/mysql-mmm/mmm\_agent.conf

include mmm\_common.conf

this db2

[root@192.168.0.50 ~]#

在db3（192.168.0.40）上：

[root@192.168.0.40 ~]# cat /etc/mysql-mmm/mmm\_agent.conf

include mmm\_common.conf

this db3

[root@192.168.0.40 ~]#

在db2（192.168.0.30）配置monitor的配置文件：

[复制代码](javascript:void(0);)

[root@192.168.0.30 ~]# cat /etc/mysql-mmm/mmm\_mon.conf

include mmm\_common.conf

<monitor>

ip 127.0.0.1

pid\_path /var/run/mysql-mmm/mmm\_mond.pid

bin\_path /usr/libexec/mysql-mmm

status\_path /var/lib/mysql-mmm/mmm\_mond.status

ping\_ips 192.168.0.40,192.168.0.50,192.168.0.60

auto\_set\_online 60

</monitor>

<host default>

monitor\_user mmm\_monitor

monitor\_password mmm\_monitor

</host>

debug 0

[root@192.168.0.30 ~]#

[复制代码](javascript:void(0);)

这里只在原有配置文件中的ping\_ips添加了整个架构被监控主机的ip地址，而在<host default>中配置了用于监控的用户。

### （6）创建监控用户，这里需要创建3个监控用户

具体描述如下：

[复制代码](javascript:void(0);)

用户名 描述 权限

monitor user MMM的monitor端监控所有的mysql数据库的状态用户 REPLICATION CLIENT

agent user 主要是MMM客户端用于改变的master的read\_only状态用户 SUPER,REPLICATION CLIENT,PROCESS

repl 用于复制的用户 REPLICATION SLAVE

[复制代码](javascript:void(0);)

在3台服务器(db1,db2,db3）进行授权，因为我之前的主主复制，以及主从已经是ok的，所以我在其中一台服务器执行就ok了。用于复制的账号之前已经有了，所以这里就授权两个账号。

[复制代码](javascript:void(0);)

mysql> GRANT SUPER, REPLICATION CLIENT, PROCESS ON \*.\* TO 'mmm\_agent'@'192.168.0.%' IDENTIFIED BY 'mmm\_agent';

Query OK, **0** rows affected (**0.08** sec)

mysql> GRANT REPLICATION CLIENT ON \*.\* TO 'mmm\_monitor'@'192.168.0.%' IDENTIFIED BY 'mmm\_monitor';

Query OK, **0** rows affected (**0.00** sec)

mysql> flush privileges;

Query OK, **0** rows affected (**0.03** sec)

mysql>

[复制代码](javascript:void(0);)

如果是从头到尾从新搭建，则加上另外一个复制账户（分别在3台服务器都需要执行这3条SQL）：

GRANT REPLICATION SLAVE ON \*.\* TO 'repl'@'192.168.0.%' IDENTIFIED BY '123456';

### （7）启动agent服务

最后分别在db1，db2，db3上启动agent，并在db2（192.168.0.30）上启动monitor程序：

[root@192.168.0.60 ~]# /etc/init.d/mysql-mmm-agent start

Daemon bin: '/usr/sbin/mmm\_agentd'

Daemon pid: '/var/run/mmm\_agentd.pid'

Starting MMM Agent daemon... Ok

[root@192.168.0.60 ~]#

[root@192.168.0.50 ~]# /etc/init.d/mysql-mmm-agent start

Starting MMM Agent Daemon: [ OK ]

[root@192.168.0.50 ~]#

因为我有些使用yum安装的，所以启动信息有些不一样。^\_^

[root@192.168.0.40 ~]# /etc/init.d/mysql-mmm-agent start

Starting MMM Agent Daemon: [ OK ]

[root@192.168.0.40 ~]#

启动monitor：

[root@192.168.0.30 ~]# /etc/init.d/mysql-mmm-monitor start

Starting MMM Monitor Daemon: [ OK ]

[root@192.168.0.30 ~]#

其中agent的日志存放在/var/log/mysql-mmm/mmm\_agentd.log，monitor日志放在/var/log/mysql-mmm/mmm\_mond.log，启动过程中有什么问题，通常日志都会有详细的记录。

### （8）在monitor主机上检查集群主机的状态

[复制代码](javascript:void(0);)

[root@192.168.0.30 ~]# mmm\_control checks all

db2 ping [last change: 2014/04/18 00:29:01] OK

db2 mysql [last change: 2014/04/18 00:29:01] OK

db2 rep\_threads [last change: 2014/04/18 00:29:01] OK

db2 rep\_backlog [last change: 2014/04/18 00:29:01] OK: Backlog is null

db3 ping [last change: 2014/04/18 00:29:01] OK

db3 mysql [last change: 2014/04/18 00:29:01] OK

db3 rep\_threads [last change: 2014/04/18 00:29:01] OK

db3 rep\_backlog [last change: 2014/04/18 00:29:01] OK: Backlog is null

db1 ping [last change: 2014/04/18 00:29:01] OK

db1 mysql [last change: 2014/04/18 00:29:01] OK

db1 rep\_threads [last change: 2014/04/18 00:29:01] OK

db1 rep\_backlog [last change: 2014/04/18 00:29:01] OK: Backlog is null

[root@192.168.0.30 ~]#

[复制代码](javascript:void(0);)

### （9)在monitor主机上检查集群环境在线状况

[root@192.168.0.30 ~]# mmm\_control show

db1(192.168.0.60) master/ONLINE. Roles: writer(192.168.0.108)

db2(192.168.0.50) master/ONLINE. Roles: reader(192.168.0.88)

db3(192.168.0.40) slave/ONLINE. Roles: reader(192.168.0.98)

[root@192.168.0.30 ~]#

### （10）online（上线）所有主机

我这里主机已经在线了，如果没有在线，可以使用下面的命令将相关主机online

[root@192.168.0.30 ~]# mmm\_control set\_online db1

OK: This host is already ONLINE. Skipping command.

[root@192.168.0.30 ~]#

提示主机已经在线，已经跳过命令执行了。

到这里整个集群就配置完成了。从输出中可以看到虚拟ip 192.168.0.108已经顺利添加到主机192.168.0.60上作为主对外提供写服务，虚拟ip 192.168.0.88添加到主机192.168.0.50上对外提供读服务，而虚拟ip 192.168.0.98添加到192.168.0.40上对外提供读服务。

## **MMM高可用测试**

我们已经完成高可用环境的搭建了，下面我们就可以做MMM的HA测试咯。首先查看整个集群的状态，可以看到整个集群状态正常。

[root@192.168.0.30 ~]# mmm\_control show

db1(192.168.0.60) master/ONLINE. Roles: writer(192.168.0.108)

db2(192.168.0.50) master/ONLINE. Roles: reader(192.168.0.88)

db3(192.168.0.40) slave/ONLINE. Roles: reader(192.168.0.98)

[root@192.168.0.30 ~]#

模拟db2（192.168.0.50 ）宕机，手动停止mysql服务，观察monitor日志：

[root@192.168.0.30 ~]# tail -f /var/log/mysql-mmm/mmm\_mond.log

2014/04/18 00:55:53 FATAL State of host 'db2' changed from ONLINE to HARD\_OFFLINE (ping: OK, mysql: not OK)

从日志发现db2的状态有ONLINE转换为HARD\_OFFLINE

重新查看集群的最新状态：

[root@192.168.0.30 ~]# mmm\_control show

db1(192.168.0.60) master/ONLINE. Roles: writer(192.168.0.108)

db2(192.168.0.50) master/HARD\_OFFLINE. Roles:

db3(192.168.0.40) slave/ONLINE. Roles: reader(192.168.0.88), reader(192.168.0.98)

[root@192.168.0.30 ~]#

重启db2，可以看到db2由HARD\_OFFLINE转到AWAITING\_RECOVERY。这里db2再次接管读请求。

[root@192.168.0.30 ~]# mmm\_control show

db1(192.168.0.60) master/ONLINE. Roles: writer(192.168.0.108)

db2(192.168.0.50) master/ONLINE. Roles: reader(192.168.0.88)

db3(192.168.0.40) slave/ONLINE. Roles: reader(192.168.0.98)

[root@192.168.0.30 ~]#

模拟db1主库宕机：

查看集群状态：

[复制代码](javascript:void(0);)

[root@192.168.0.30 ~]# mmm\_control show

db1(192.168.0.60) master/HARD\_OFFLINE. Roles:

db2(192.168.0.50) master/ONLINE. Roles: reader(192.168.0.88), writer(192.168.0.108)

db3(192.168.0.40) slave/ONLINE. Roles: reader(192.168.0.98)

[root@192.168.0.30 ~]#

[复制代码](javascript:void(0);)

查看MMM日志：

[root@192.168.0.30 ~]# tail -f /var/log/mysql-mmm/mmm\_mond.log

2014/04/18 01:09:20 FATAL State of host 'db1' changed from ONLINE to HARD\_OFFLINE (ping: OK, mysql: not OK)

从上面可以发现，db1由以前的ONLINE转化为HARD\_OFFLINE，移除了写角色，因为db2是备选主，所以接管了写角色，db3指向新的主库db2，应该说db3实际上找到了db2的sql现在的位置，即db2 show master返回的值，然后直接在db3上change master to到db2。

db1，db2，db3之间为一主两从的复制关系，一旦发生db2，db3延时于db1时，这个时刻db1 mysql宕机，db3将会等待数据追上db1后，再重新指向新的主db2，进行change master to db2操作，在db1宕机的过程中，一旦db2落后于db1，这时发生切换，db2变成了可写状态，数据的一致性将会无法保证。

## **总结**

**MMM不适用于对数据一致性要求很高的环境。但是高可用完全做到了。**

# [Heartbeat+DRBD+MySQL高可用方案](https://www.cnblogs.com/gomysql/p/3674030.html)

## **1.方案简介**

本方案采用Heartbeat双机热备软件来保证数据库的高稳定性和连续性，数据的一致性由DRBD这个工具来保证。默认情况下只有一台mysql在工作，当主mysql服务器出现问题后，系统将自动切换到备机上继续提供服务，当主数据库修复完毕，又将服务切回继续由主mysql提供服务。

## **2.方案优缺点**

优点：安全性高、稳定性高、可用性高，出现故障自动切换。

缺点：只有一台服务器提供服务，成本相对较高，不方便扩展，可能会发生脑裂。

## **3.软件介绍**

Heartbeat介绍

官方站点：<http://linux-ha.org/wiki/Main_Page>

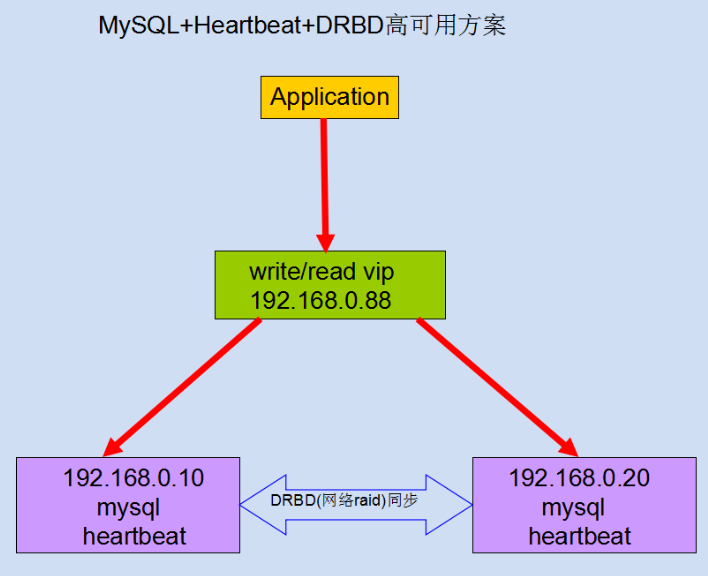
heartbeat可以资源(VIP地址及程序服务)从一台有故障的服务器快速的转移到另一台正常的服务器提供服务，heartbeat和keepalived相似，heartbeat可以实现failover功能，但不能实现对后端的健康检查

DRBD介绍

官方站点：<http://www.drbd.org/>

DRBD(DistributedReplicatedBlockDevice)是一个基于块设备级别在远程服务器直接同步和镜像数据的软件，用软件实现的、无共享的、服务器之间镜像块设备内容的存储复制解决方案。它可以实现在网络中两台服务器之间基于块设备级别的实时镜像或同步复制(两台服务器都写入成功)/异步复制(本地服务器写入成功)，相当于网络的RAID1，由于是基于块设备(磁盘，LVM逻辑卷)，在文件系统的底层，所以数据复制要比cp命令更快。DRBD已经被MySQL官方写入文档手册作为推荐的高可用的方案之一

## **4.方案拓扑**



## **5.方案适用场景：**

适用于数据库访问量不太大，短期内访问量增长不会太快，对数据库可用性要求非常高的场景。

## **6.测试环境介绍**

**（如下所示，均已关闭防火墙及selinux，生产环境自行开放端口）**

主机名 ip 系统 DRBD磁盘 heartbeat版本

db-server-01 192.168.0.10 centos6.2 64bit /dev/sda5 3.0.4

db-server-02 192.168.0.20 centos6.2 64bit /dev/sda5 3.0.4

## **7.软件安装以及环境配置**

### （1）安装drbd依赖组件

（两台机器，安装以后重启系统，因为会升级内核版本，不重启会对不上内核版本，有知道不用重启的童鞋请给我留言^\_^）：

yum install -y kernel kernel-devel kernel-headers flex

### （2）下载软件安装（两台机器操作一样）

wget http://oss.linbit.com/drbd/8.4/drbd-8.4.2.tar.gz

[复制代码](javascript:void(0);)

tar xf drbd-8.4.2.tar.gz

cd drbd-8.4.2

./configure --prefix=/usr/local/drbd --with-km

make KDIR=/usr/src/kernels/2.6.32-431.11.2.el6.x86\_64/ #很多童鞋无法加载drbd模块，多半是正在运行的内核版本和新安装的不相符

make install

mkdir -p /usr/local/drbd/var/run/drbd

cp /usr/local/drbd/etc/rc.d/init.d/drbd /etc/rc.d/init.d

chmod 755 /etc/init.d/drbd

cd drbd

make clean

make KDIR=/usr/src/kernels/2.6.32-431.11.2.el6.x86\_64/

cp drbd.ko /lib/modules/`uname -r`/kernel/lib/

modprobe drbd

[复制代码](javascript:void(0);)

检查是否加载了drbd模块

[root@192.168.0.10 ~]# lsmod | grep drbd

drbd 314246 0

libcrc32c 1246 1 drbd

[root@192.168.0.10 ~]#

### （3）DRBD配置

（配置之前需要先使用fdisk对 /dev/sda进行分区）

[复制代码](javascript:void(0);)

[root@192.168.0.10 ~]# df -HT

Filesystem Type Size Used Avail Use% Mounted on

/dev/sda2 ext4 19G 2.6G 16G 15% /

tmpfs tmpfs 121M 0 121M 0% /dev/shm

/dev/sda1 ext4 204M 52M 141M 27% /boot

/dev/sda5 ext4 34G 185M 32G 1% /data

[root@192.168.0.10 ~]#

[复制代码](javascript:void(0);)

我这里两台机器之前都已经分区了，由于是自己笔记本上的虚拟机，所以懒得加磁盘了，我直接把 /data/卸载，然后格式化/dev/sda5，我两台机器都这样操作，如果你有空的磁盘，照样需要进行分区，比如可以将一个1T的盘分一个区就行了。

[复制代码](javascript:void(0);)

[root@192.168.0.10 ~]# umount /data/

[root@192.168.0.10 ~]# mkfs.ext4 /dev/sda5

mke2fs 1.41.12 (17-May-2010)

Filesystem label=

OS type: Linux

Block size=4096 (log=2)

Fragment size=4096 (log=2)

Stride=0 blocks, Stripe width=0 blocks

2048000 inodes, 8185344 blocks

409267 blocks (5.00%) reserved for the super user

First data block=0

Maximum filesystem blocks=4294967296

250 block groups

32768 blocks per group, 32768 fragments per group

8192 inodes per group

Superblock backups stored on blocks:

32768, 98304, 163840, 229376, 294912, 819200, 884736, 1605632, 2654208,

4096000, 7962624

Writing inode tables: done

Creating journal (32768 blocks): done

Writing superblocks and filesystem accounting information: done

This filesystem will be automatically checked every 28 mounts or

180 days, whichever comes first. Use tune2fs -c or -i to override.

[root@192.168.0.10 ~]#

[复制代码](javascript:void(0);)

[复制代码](javascript:void(0);)

[root@192.168.0.10 ~]# fdisk -l

Disk /dev/sda: 53.7 GB, 53687091200 bytes

255 heads, 63 sectors/track, 6527 cylinders

Units = cylinders of 16065 \* 512 = 8225280 bytes

Sector size (logical/physical): 512 bytes / 512 bytes

I/O size (minimum/optimal): 512 bytes / 512 bytes

Disk identifier: 0x000eb0ff

Device Boot Start End Blocks Id System

/dev/sda1 \* 1 26 204800 83 Linux

Partition 1 does not end on cylinder boundary.

/dev/sda2 26 2321 18432000 83 Linux

/dev/sda3 2321 2451 1048576 82 Linux swap / Solaris

/dev/sda4 2451 6528 32742400 5 Extended

/dev/sda5 2451 6528 32741376 83 Linux

[root@192.168.0.10 ~]#

[复制代码](javascript:void(0);)

我这里还要在/etc/fstab里面注释一项：

#UUID=33958004-e8a7-4135-844f-707a5537e86a /data ext4 defaults 1 2

否则重启机器的时候提示无法挂载，会无法启动的。

修改/etc/hosts文件，两台服务器操作一样。

192.168.0.10 db-server-01

192.168.0.20 db-server-02

drbd配置只需要修改/usr/local/drbd/etc/drbd.d/global\_common.conf配置文件即可，修改后如下（两台服务器配置一样）：

[复制代码](javascript:void(0);)

[root@192.168.0.10 ~]# cat /usr/local/drbd/etc/drbd.d/global\_common.conf

global { usage-count yes; }

common { syncer { rate 30M; } } #同步速率，视带宽而定

resource r0 { #创建一个资源，名字叫"r0"

protocol C; #选择的是drbd的C 协议（数据同步协议，C为收到数据并写入后返回，确认成功）

startup {

}

disk {

on-io-error detach;

}

net {

}

on db-server-01 { #设定一个节点，分别以各自的主机名命名

device /dev/drbd0; #设定资源设备/dev/drbd0 指向实际的物理分区 /dev/sda5

disk /dev/sda5;

address 192.168.0.10:7888; #设定监听地址以及端口

meta-disk internal;

}

on db-server-02 {

device /dev/drbd0;

disk /dev/sda5;

address 192.168.0.20:7888;

meta-disk internal; #internal表示是在同一个局域网内

}

}

[root@192.168.0.10 ~]#

[复制代码](javascript:void(0);)

### （4）DRBD的管理与维护

创建DRBD资源

配置好drbd以后，就需要使用命令创建配置的drbd资源，使用如下命令（两台服务器操作一样）：

[root@192.168.0.10 ~]# dd if=/dev/zero of=/dev/sda5 bs=1M count=100 #不这样做的话，在创建资源的时候报错

100+0 records in

100+0 records out

104857600 bytes (105 MB) copied, 3.34339 s, 31.4 MB/s

[root@192.168.0.10 ~]#

[复制代码](javascript:void(0);)

[root@192.168.0.10 ~]# drbdadm create-md r0

Writing meta data...

initializing activity log

NOT initializing bitmap

New drbd meta data block successfully created.

success

[root@192.168.0.10 ~]#

[复制代码](javascript:void(0);)

### （5）DRBD的启动与状态查看（分别在两台服务器启动）

[复制代码](javascript:void(0);)

[root@192.168.0.10 ~]# /etc/init.d/drbd start

Starting DRBD resources: [

create res: r0

prepare disk: r0

adjust disk: r0

adjust net: r0

]

.....

[root@192.168.0.10 ~]#

[复制代码](javascript:void(0);)

[复制代码](javascript:void(0);)

[root@192.168.0.20 ~]# /etc/init.d/drbd start

Starting DRBD resources: [

create res: r0

prepare disk: r0

adjust disk: r0

adjust net: r0

]

.

[root@192.168.0.20 ~]#

[复制代码](javascript:void(0);)

查看drbd的状态：

[复制代码](javascript:void(0);)

[root@192.168.0.10 ~]# /etc/init.d/drbd status

drbd driver loaded OK; device status:

version: 8.4.2 (api:1/proto:86-101)

GIT-hash: 7ad5f850d711223713d6dcadc3dd48860321070c build by root@db-server-01, 2014-04-18 21:15:57

m:res cs ro ds p mounted fstype

0:r0 Connected Secondary/Secondary Inconsistent/Inconsistent C

[root@192.168.0.10 ~]#

[复制代码](javascript:void(0);)

可以看见都还没有主节点。设置当前节点（192.168.0.10）为主节点，并进行格式化和挂载 。

drbdadm -- --overwrite-data-of-peer primary all

mkfs.ext4 /dev/drbd0

mkdir /data

mount /dev/drbd0 /data/

在另外一台服务器创建挂载目录，也创建/data

[root@192.168.0.20 ~]# mkdir /data

查看一下drbd的状态（可以看见还在同步）：

[复制代码](javascript:void(0);)

[root@192.168.0.10 ~]# /etc/init.d/drbd status

drbd driver loaded OK; device status:

version: 8.4.2 (api:1/proto:86-101)

GIT-hash: 7ad5f850d711223713d6dcadc3dd48860321070c build by root@db-server-01, 2014-04-18 21:15:57

m:res cs ro ds p mounted fstype

... sync'ed: 13.7% (27596/31972)M

0:r0 SyncSource Primary/Secondary UpToDate/Inconsistent C /data ext4

[root@192.168.0.10 ~]#

[复制代码](javascript:void(0);)

### （6）mysql安装

我这里为了简单直接安装编译好的二进制软件包（两台服务器都需要安装，操作一样，只是第二台mysql不需要初始化数据）

**注意：两台服务器上的mysql用户的uid和gid要一样。不然切换后会导致mysql数据目录的属主不正确而启动失败。**

[root@192.168.0.10 ~]# wget http://cdn.mysql.com/Downloads/MySQL-5.5/mysql-5.5.37-linux2.6-x86\_64.tar.gz

[复制代码](javascript:void(0);)

[root@192.168.0.10 ~]# tar xf mysql-5.5.37-linux2.6-x86\_64.tar.gz -C /usr/local/

[root@192.168.0.10 ~]# cd /usr/local/

[root@192.168.0.10 local]# ln -s mysql-5.5.37-linux2.6-x86\_64/ mysql

[root@192.168.0.10 local]# groupadd mysql

[root@192.168.0.10 local]# useradd -r -g mysql mysql

[root@192.168.0.10 local]# cd mysql

[root@192.168.0.10 mysql]# chown -R mysql .

[root@192.168.0.10 mysql]# chgrp -R mysql .

[root@192.168.0.10 mysql]# mkdir /data/mysql

[root@192.168.0.10 mysql]# chown -R mysql.mysql /data/mysql/

[root@192.168.0.10 mysql]# /usr/local/mysql/scripts/mysql\_install\_db --user=mysql --datadir=/data/mysql/ --basedir=/usr/local/mysql

[复制代码](javascript:void(0);)

[root@192.168.0.10 mysql]# chown -R root .

[root@192.168.0.10 mysql]# cp support-files/my-medium.cnf /etc/my.cnf

[root@192.168.0.10 mysql]# cp support-files/mysql.server /etc/init.d/mysqld

[root@192.168.0.10 mysql]# chmod 755 /etc/init.d/mysqld

[root@192.168.0.10 mysql]# egrep 'datadir|basedir' /etc/my.cnf #两台服务器上的mysql配置文件都加入这里的配置

datadir=/data/mysql

basedir=/usr/local/mysql

[root@192.168.0.10 mysql]#

### （7）手动切换drbd的主从

看另外一台服务器是否有数据（自动切换需要使用heartbeat，后面介绍）：

[root@192.168.0.10 ~]# ll /data/

total 20

drwx------ 2 root root 16384 Apr 18 22:16 lost+found

drwxr-xr-x 5 mysql mysql 4096 Apr 18 23:01 mysql

[root@192.168.0.10 ~]#

[root@192.168.0.20 ~]# ll /data/

total 0

[root@192.168.0.20 ~]#

[复制代码](javascript:void(0);)

[root@192.168.0.10 ~]# /etc/init.d/drbd status

drbd driver loaded OK; device status:

version: 8.4.2 (api:1/proto:86-101)

GIT-hash: 7ad5f850d711223713d6dcadc3dd48860321070c build by root@db-server-01, 2014-04-18 21:15:57

m:res cs ro ds p mounted fstype

0:r0 Connected Primary/Secondary UpToDate/UpToDate C /data ext4

[root@192.168.0.10 ~]#

[复制代码](javascript:void(0);)

可以看见当前服务器是主，也就是数据在这台服务器上，另外一台服务器是没有数据的。下面进行手动切换

主切换成从，需要先卸载文件系统，再执行降级为从的命令：

[root@192.168.0.10 ~]# umount /data/

[root@192.168.0.10 ~]# drbdadm secondary all

从切换成主，要先执行升级成主的命令然后挂在文件系统：

[复制代码](javascript:void(0);)

[root@192.168.0.20 ~]# drbdadm primary all

[root@192.168.0.20 ~]# mount /dev/drbd0 /data/

[root@192.168.0.20 ~]# ll /data/

total 20

drwx------ 2 root root 16384 Apr 18 22:16 lost+found

drwxr-xr-x 5 mysql mysql 4096 Apr 18 23:01 mysql

[root@192.168.0.20 ~]# /etc/init.d/drbd status

drbd driver loaded OK; device status:

version: 8.4.2 (api:1/proto:86-101)

GIT-hash: 7ad5f850d711223713d6dcadc3dd48860321070c build by root@db-server-02, 2014-04-18 21:22:55

m:res cs ro ds p mounted fstype

0:r0 Connected Primary/Secondary UpToDate/UpToDate C /data ext4

[root@192.168.0.20 ~]#

[复制代码](javascript:void(0);)

可以看见已经成功切换成主，并且mysql初始化数据也存在了。

**DRBD脑裂后的处理**

**当DRBD出现脑裂后，会导致drbd两边的磁盘数据不一致，在确定要作为从的节点上切换成secondary，并放弃该资源的数据：**

drbdadm secondary r0

drbdadm -- --discard-my-data connect r0

**在要作为primary的节点重新连接secondary（如果这个节点当前的连接状态为WFConnection的话，可以省略），使用如下命令连接：**

drbdadm connect r0

### （8）Heartbeat安装（两台服务器）

需要添加epel源，centos默认自己没有该软件包，当然你可以自己源码编译。

rpm -ivh http://dl.fedoraproject.org/pub/epel/6/x86\_64/epel-release-6-8.noarch.rpm

yum install heartbeat -y

创建DRBD脚本文件drbddisk:(两台服务器)

**注意：**

**此处是一个大坑，因为默认yum安装Heartbeat，不会在/etc/ha.d/resource.d/创建drbddisk脚本，估计是版本太新了吧。记得前两年都不会这样的。囧。而且也无法在安装后从本地其他路径找到该文件。此处也是因为启动Heartbeat后无法PING通虚IP，最后通过查看/var/log/ha-log日志，找到一行ERROR: Cannot locate resource script drbddisk，然后进而到/etc/ha.d/resource.d/路径下发现竟然没有drbddisk脚本，最后在google上找到该代码，创建该脚本，终于测试通过：**

[复制代码](javascript:void(0);)

[root@192.168.0.20 ~]# chmod 755 /etc/ha.d/resource.d/drbddisk

[root@192.168.0.20 ~]# cat /etc/ha.d/resource.d/drbddisk

#!/bin/bash

#

# This script is inteded to be used as resource script by heartbeat

#

# Copright 2003-2008 LINBIT Information Technologies

# Philipp Reisner, Lars Ellenberg

#

###

DEFAULTFILE="/etc/default/drbd"

DRBDADM="/sbin/drbdadm"

if [ -f $DEFAULTFILE ]; then

. $DEFAULTFILE

fi

if [ "$#" -eq 2 ]; then

RES="$1"

CMD="$2"

else

RES="all"

CMD="$1"

fi

## EXIT CODES

# since this is a "legacy heartbeat R1 resource agent" script,

# exit codes actually do not matter that much as long as we conform to

# http://wiki.linux-ha.org/HeartbeatResourceAgent

# but it does not hurt to conform to lsb init-script exit codes,

# where we can.

# http://refspecs.linux-foundation.org/LSB\_3.1.0/

#LSB-Core-generic/LSB-Core-generic/iniscrptact.html

####

drbd\_set\_role\_from\_proc\_drbd()

{

local out

if ! test -e /proc/drbd; then

ROLE="Unconfigured"

return

fi

dev=$( $DRBDADM sh-dev $RES )

minor=${dev#/dev/drbd}

if [[ $minor = \*[!0-9]\* ]] ; then

# sh-minor is only supported since drbd 8.3.1

minor=$( $DRBDADM sh-minor $RES )

fi

if [[ -z $minor ]] || [[ $minor = \*[!0-9]\* ]] ; then

ROLE=Unknown

return

fi

if out=$(sed -ne "/^ \*$minor: cs:/ { s/:/ /g; p; q; }" /proc/drbd); then

set -- $out

ROLE=${5%/\*\*}

: ${ROLE:=Unconfigured} # if it does not show up

else

ROLE=Unknown

fi

}

case "$CMD" in

start)

# try several times, in case heartbeat deadtime

# was smaller than drbd ping time

try=6

while true; do

$DRBDADM primary $RES && break

let "--try" || exit 1 # LSB generic error

sleep 1

done

;;

stop)

# heartbeat (haresources mode) will retry failed stop

# for a number of times in addition to this internal retry.

try=3

while true; do

$DRBDADM secondary $RES && break

# We used to lie here, and pretend success for anything != 11,

# to avoid the reboot on failed stop recovery for "simple

# config errors" and such. But that is incorrect.

# Don't lie to your cluster manager.

# And don't do config errors...

let --try || exit 1 # LSB generic error

sleep 1

done

;;

status)

if [ "$RES" = "all" ]; then

echo "A resource name is required for status inquiries."

exit 10

fi

ST=$( $DRBDADM role $RES )

ROLE=${ST%/\*\*}

case $ROLE in

Primary|Secondary|Unconfigured)

# expected

;;

\*)

# unexpected. whatever...

# If we are unsure about the state of a resource, we need to

# report it as possibly running, so heartbeat can, after failed

# stop, do a recovery by reboot.

# drbdsetup may fail for obscure reasons, e.g. if /var/lock/ is

# suddenly readonly. So we retry by parsing /proc/drbd.

drbd\_set\_role\_from\_proc\_drbd

esac

case $ROLE in

Primary)

echo "running (Primary)"

exit 0 # LSB status "service is OK"

;;

Secondary|Unconfigured)

echo "stopped ($ROLE)"

exit 3 # LSB status "service is not running"

;;

\*)

# NOTE the "running" in below message.

# this is a "heartbeat" resource script,

# the exit code is \_ignored\_.

echo "cannot determine status, may be running ($ROLE)"

exit 4 # LSB status "service status is unknown"

;;

esac

;;

\*)

echo "Usage: drbddisk [resource] {start|stop|status}"

exit 1

;;

esac

exit 0

[root@192.168.0.20 ~]#

[复制代码](javascript:void(0);)

### （9）heartbeat配置

Hearbeat的配置主要包括三个配置文件，authkeys，ha.cf和haresources的配置，下面就分别来看看：

Authkerys的配置（两台服务器配置一样）

这个文件用来配置密码认证方式，支持3种认证方式，crc，md5和sha1，从左到右安全性越来越高，消耗的资源也越多。因此如果heartbeat运行在安全的网路之上，比如私网，那么可以将验证方式设置成crc，master和backup的authkeys配置一样。我的authkeys文件配置如下：

[root@192.168.0.10 ~]# cat /etc/ha.d/authkeys

auth 1

1 crc

[root@192.168.0.10 ~]# chmod 600 /etc/ha.d/authkeys

**注意：该文件权限必须是600**

ha.cf的配置（两台机器稍微有点区别），Primary（192.168.0.10）如下：

[复制代码](javascript:void(0);)

[root@192.168.0.10 ~]# cat /etc/ha.d/ha.cf

logfile /var/log/ha-log

#定义Heartbeat的日志名字及位置

logfacility local0

keepalive 2

#设定心跳(监测)时间为2秒

deadtime 15

#设定死亡时间为15秒

ucast eth1 192.168.0.20

#采用单播的方式，IP地址指定为对方IP

auto\_failback off

#当Primary机器发生故障切换到Secondary机器后Primary恢复后是否进行切回操作 （最好是我们有需求手动进行切换）

node db-server-01

node db-server-02

[root@192.168.0.10 ~]#

[复制代码](javascript:void(0);)

Secondary（192.168.0.20）如下：

[复制代码](javascript:void(0);)

[root@192.168.0.20 ~]# cat /etc/ha.d/ha.cf

logfile /var/log/ha-log

#定义Heartbeat的日志名字及位置

logfacility local0

keepalive 2

#设定心跳(监测)时间为2秒

deadtime 15

#设定死亡时间为15秒

ucast eth1 192.168.0.10

#采用单播的方式，IP地址指定为对方IP

auto\_failback off

#当Primary机器发生故障切换到Secondary机器后Primary恢复后是否进行切回操作（一般我们可以看需求，否则不用自动切换）

node db-server-01

node db-server-02

[root@192.168.0.20 ~]#

[复制代码](javascript:void(0);)

haresources的配置（两台机器配置一样）：

[root@192.168.0.10 ~]# cat /etc/ha.d/haresources

db-server-01 IPaddr::192.168.0.88/24/eth1 drbddisk::r0 Filesystem::/dev/drbd0::/data::ext4 mysqld

[root@192.168.0.10 ~]#

**注：该文件内IPaddr,Filesystem等脚本存放路径在/etc/ha.d/resource.d/下,也可在该目录下存放服务启动脚本（例如：mysqld）,将相同脚本名称添到/etc/ha.d/haresources内容中，从而跟随heartbeat启动而启动该脚本。**

**IPaddr::192.168.0.88/24/eth1：用IPaddr脚本配置浮动VIP**

**drbddisk::r0：用drbddisk脚本实现DRBD主从节点资源组的挂载和卸载**

**Filesystem::/dev/drbd0::/data::ext4：用Filesystem脚本实现磁盘挂载和卸载**

### （10）heartbeat的管理

配置好heartbeat之后，需要将mysql从自启动服务器中去掉，因为主heartbeat启动的时候会挂载drdb文件系统以及启动mysql，切换的时候会将主上的mysql停止并卸载文件系统，从上会挂载文件系统，并启动mysql。因此需要做如下操作（两台服务器）：

[root@192.168.0.10 ~]# chkconfig mysqld off

[root@192.168.0.10 ~]# chkconfig heartbeat off

[root@192.168.0.10 ~]# chkconfig drbd off

[复制代码](javascript:void(0);)

[root@192.168.0.10 ~]# cat /etc/rc.local

#!/bin/sh

#

# This script will be executed \*after\* all the other init scripts.

# You can put your own initialization stuff in here if you don't

# want to do the full Sys V style init stuff.

touch /var/lock/subsys/local

**modprobe drbd #必须先加载模块，这也是因为将启动命令放在这里的原因**

**/etc/init.d/drbd start**

**/etc/init.d/heartbeat start**

[root@192.168.0.10 ~]#

[复制代码](javascript:void(0);)

到这里heartbeat+drbd+mysql高可用环境就搭建结束了。接下来进行测试。

## **8.高可用测试**

（1）在第一台服务器上面启动mysql服务。（192.168.0.10）

[root@192.168.0.10 ~]# /etc/init.d/mysqld start

Starting MySQL.The server quit without updating PID file (/[FAILED]ql/db-server-01.pid).

[root@192.168.0.10 ~]# ll /data/

total 0

[root@192.168.0.10 ~]#

怎么回事？/data/下面为空。这里是因为我们在前面已经把这个节点变为Secondary

[复制代码](javascript:void(0);)

[root@192.168.0.10 ~]# /etc/init.d/drbd status

drbd driver loaded OK; device status:

version: 8.4.2 (api:1/proto:86-101)

GIT-hash: 7ad5f850d711223713d6dcadc3dd48860321070c build by root@db-server-01, 2014-04-18 21:15:57

m:res cs ro ds p mounted fstype

0:r0 Connected Secondary/Primary UpToDate/UpToDate C

[root@192.168.0.10 ~]#

[复制代码](javascript:void(0);)

我们现在需要手动切换回来。才能启动mysql

[root@192.168.0.20 ~]# umount /data/

[root@192.168.0.20 ~]# drbdadm secondary all

[root@192.168.0.20 ~]#

[复制代码](javascript:void(0);)

[root@192.168.0.10 ~]# drbdadm primary all

[root@192.168.0.10 ~]# mount /dev/drbd0 /data/

[root@192.168.0.10 ~]# ll /data/

total 20

drwx------ 2 root root 16384 Apr 18 22:16 lost+found

drwxr-xr-x 5 mysql mysql 4096 Apr 18 23:01 mysql

[root@192.168.0.10 ~]# /etc/init.d/drbd status

drbd driver loaded OK; device status:

version: 8.4.2 (api:1/proto:86-101)

GIT-hash: 7ad5f850d711223713d6dcadc3dd48860321070c build by root@db-server-01, 2014-04-18 21:15:57

m:res cs ro ds p mounted fstype

0:r0 Connected Primary/Secondary UpToDate/UpToDate C /data ext4

[root@192.168.0.10 ~]#

[复制代码](javascript:void(0);)

可以看见已经切换回来了，我们现在可以启动mysql了。

[复制代码](javascript:void(0);)

[root@192.168.0.10 ~]# /etc/init.d/mysqld start

Starting MySQL....... [ OK ]

[root@192.168.0.10 ~]# mysql

Welcome to the MySQL monitor. Commands end with ; or \g.

Your MySQL connection id is 1

Server version: 5.5.37-log MySQL Community Server (GPL)

Copyright (c) 2000, 2014, 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>

[复制代码](javascript:void(0);)

（2）在两台服务器上面启动heartbeat

[root@192.168.0.10 ~]# /etc/init.d/heartbeat start

Starting High-Availability services: INFO: Resource is stopped

Done.

[root@192.168.0.10 ~]#

[root@192.168.0.20 ~]# /etc/init.d/heartbeat start

Starting High-Availability services: INFO: Resource is stopped

Done.

[root@192.168.0.20 ~]#

[root@192.168.0.10 ~]# ip addr | grep eth1

3: eth1: <BROADCAST,MULTICAST,UP,LOWER\_UP> mtu 1500 qdisc pfifo\_fast state UP qlen 1000

inet 192.168.0.10/24 brd 192.168.0.255 scope global eth1

inet 192.168.0.88/24 brd 192.168.0.255 scope global secondary eth1

[root@192.168.0.10 ~]#

可以看见虚拟ip192.168.0.88已经存在了。说明成功了。我们看看heartbeat的日志就能发现。

[复制代码](javascript:void(0);)

[root@192.168.0.10 ~]# tail -n 20 /var/log/ha-log

harc(default)[5598]: 2014/04/19\_00:25:21 info: Running /etc/ha.d//rc.d/status status

Apr 19 00:25:22 db-server-01 heartbeat: [5591]: info: Comm\_now\_up(): updating status to active

Apr 19 00:25:22 db-server-01 heartbeat: [5591]: info: Local status now set to: 'active'

Apr 19 00:25:22 db-server-01 heartbeat: [5591]: info: Status update for node db-server-02: status active

harc(default)[5618]: 2014/04/19\_00:25:22 info: Running /etc/ha.d//rc.d/status status

Apr 19 00:25:33 db-server-01 heartbeat: [5591]: info: remote resource transition completed.

Apr 19 00:25:33 db-server-01 heartbeat: [5591]: info: remote resource transition completed.

Apr 19 00:25:33 db-server-01 heartbeat: [5591]: info: Initial resource acquisition complete (T\_RESOURCES(us))

/usr/lib/ocf/resource.d//heartbeat/IPaddr(IPaddr\_192.168.0.88)[5671]: 2014/04/19\_00:25:33 INFO: Resource is stopped

Apr 19 00:25:33 db-server-01 heartbeat: [5635]: info: Local Resource acquisition completed.

harc(default)[5752]: 2014/04/19\_00:25:33 info: Running /etc/ha.d//rc.d/ip-request-resp ip-request-resp

ip-request-resp(default)[5752]: 2014/04/19\_00:25:33 received ip-request-resp IPaddr::192.168.0.88/24/eth1 OK yes

ResourceManager(default)[5775]: 2014/04/19\_00:25:33 info: Acquiring resource group: db-server-01 IPaddr::192.168.0.88/24/eth1 drbddisk::r0 Filesystem::/dev/drbd0::/data::ext4 mysqld

/usr/lib/ocf/resource.d//heartbeat/IPaddr(IPaddr\_192.168.0.88)[5803]: 2014/04/19\_00:25:33 INFO: Resource is stopped

ResourceManager(default)[5775]: 2014/04/19\_00:25:33 info: Running /etc/ha.d/resource.d/IPaddr 192.168.0.88/24/eth1 start

IPaddr(IPaddr\_192.168.0.88)[5926]: 2014/04/19\_00:25:34 INFO: Adding inet address 192.168.0.88/24 with broadcast address 192.168.0.255 to device eth1

IPaddr(IPaddr\_192.168.0.88)[5926]: 2014/04/19\_00:25:34 INFO: Bringing device eth1 up

IPaddr(IPaddr\_192.168.0.88)[5926]: 2014/04/19\_00:25:34 INFO: /usr/libexec/heartbeat/send\_arp -i 200 -r 5 -p /var/run/resource-agents/send\_arp-192.168.0.88 eth1 192.168.0.88 auto not\_used not\_used

/usr/lib/ocf/resource.d//heartbeat/IPaddr(IPaddr\_192.168.0.88)[5900]: 2014/04/19\_00:25:34 INFO: Success

/usr/lib/ocf/resource.d//heartbeat/Filesystem(Filesystem\_/dev/drbd0)[6030]: 2014/04/19\_00:25:34 INFO: Running OK

[root@192.168.0.10 ~]#

[复制代码](javascript:void(0);)

激动的时刻到了，我们测试一下自动切换。我们先看看两台服务器的状态：

[复制代码](javascript:void(0);)

[root@192.168.0.10 ~]# df -HT

Filesystem Type Size Used Avail Use% Mounted on

/dev/sda2 ext4 19G 3.5G 15G 20% /

tmpfs tmpfs 121M 0 121M 0% /dev/shm

/dev/sda1 ext4 204M 52M 141M 27% /boot

/dev/drbd0 ext4 33G 216M 32G 1% /data

[root@192.168.0.10 ~]#

[复制代码](javascript:void(0);)

[root@192.168.0.20 ~]# df -HT

Filesystem Type Size Used Avail Use% Mounted on

/dev/sda2 ext4 19G 4.9G 13G 28% /

tmpfs tmpfs 121M 0 121M 0% /dev/shm

/dev/sda1 ext4 204M 52M 141M 27% /boot

[root@192.168.0.20 ~]#

可以看见挂载在第一台服务器。

测试方法：

**1.停掉master上的mysqld，看看是否切换(因为heartheat不检查服务的可用性，因此需要通过而外的脚本来实现)。**  
2.停掉master的heartheat看看是否能正常切换。   
3.停掉master的网络或者直接将master系统shutdown，看看能否正常切换。   
4.启动master的heartbeat看看是否能正常切换回来。   
5.重新启动master看看能否切换过程是否OK。   
注意：这里说的切换是不是已经将mysql停掉、是否卸载了文件系统等等。

我就停止master（192.168.0.10）上的heartbeat来测试是否会自动切换，这里除了第一条无法实现，其他的都可以切换：

[root@192.168.0.10 ~]# /etc/init.d/heartbeat stop

Stopping High-Availability services: Done.

[复制代码](javascript:void(0);)

[root@192.168.0.10 ~]# df -HT

Filesystem Type Size Used Avail Use% Mounted on

/dev/sda2 ext4 19G 3.5G 15G 20% /

tmpfs tmpfs 121M 0 121M 0% /dev/shm

/dev/sda1 ext4 204M 52M 141M 27% /boot

[root@192.168.0.10 ~]# /etc/init.d/drbd status

drbd driver loaded OK; device status:

version: 8.4.2 (api:1/proto:86-101)

GIT-hash: 7ad5f850d711223713d6dcadc3dd48860321070c build by root@db-server-01, 2014-04-18 21:15:57

m:res cs ro ds p mounted fstype

0:r0 Connected Secondary/Primary UpToDate/UpToDate C

[root@192.168.0.10 ~]#

[复制代码](javascript:void(0);)

可以看见已经切换了，我们看另外一台机器的情况：

[复制代码](javascript:void(0);)

[root@192.168.0.20 ~]# df -HT

Filesystem Type Size Used Avail Use% Mounted on

/dev/sda2 ext4 19G 4.9G 13G 28% /

tmpfs tmpfs 121M 0 121M 0% /dev/shm

/dev/sda1 ext4 204M 52M 141M 27% /boot

/dev/drbd0 ext4 33G 216M 32G 1% /data

[root@192.168.0.20 ~]# netstat -nltp | grep 3306 | grep -v grep

tcp 0 0 0.0.0.0:3306 0.0.0.0:\* LISTEN 5542/mysqld

[root@192.168.0.20 ~]#

[复制代码](javascript:void(0);)

可以发现已经切换过来，mysql也自动启动了。之前是没有启动的。

[复制代码](javascript:void(0);)

[root@192.168.0.20 ~]# ip addr | grep eth1

3: eth1: <BROADCAST,MULTICAST,UP,LOWER\_UP> mtu 1500 qdisc pfifo\_fast state UP qlen 1000

inet 192.168.0.20/24 brd 192.168.0.255 scope global eth1

inet 192.168.0.88/24 brd 192.168.0.255 scope global secondary eth1

[root@192.168.0.20 ~]# mysql

Welcome to the MySQL monitor. Commands end with ; or \g.

Your MySQL connection id is 1

Server version: 5.5.37-log MySQL Community Server (GPL)

Copyright (c) 2000, 2014, 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>

[复制代码](javascript:void(0);)

可以看见，一切正常呢。如果我们查看日志，就可以看见到底发生了什么。

[复制代码](javascript:void(0);)

[root@192.168.0.20 ~]# tail -n 10 /var/log/ha-log

ResourceManager(default)[4768]: 2014/04/19\_00:36:42 info: Running /etc/ha.d/resource.d/Filesystem /dev/drbd0 /data ext4 start

Filesystem(Filesystem\_/dev/drbd0)[5131]: 2014/04/19\_00:36:42 INFO: Running start for /dev/drbd0 on /data

/usr/lib/ocf/resource.d//heartbeat/Filesystem(Filesystem\_/dev/drbd0)[5122]: 2014/04/19\_00:36:42 INFO: Success

ResourceManager(default)[4768]: 2014/04/19\_00:36:43 info: Running /etc/init.d/mysqld start

mach\_down(default)[4741]: 2014/04/19\_00:36:46 info: /usr/share/heartbeat/mach\_down: nice\_failback: foreign resources acquired

mach\_down(default)[4741]: 2014/04/19\_00:36:46 info: mach\_down takeover complete for node db-server-01.

Apr 19 00:36:46 db-server-02 heartbeat: [4637]: info: mach\_down takeover complete.

Apr 19 00:36:58 db-server-02 heartbeat: [4637]: WARN: node db-server-01: is dead

Apr 19 00:36:58 db-server-02 heartbeat: [4637]: info: Dead node db-server-01 gave up resources.

Apr 19 00:36:58 db-server-02 heartbeat: [4637]: info: Link db-server-01:eth1 dead.

[root@192.168.0.20 ~]#

[复制代码](javascript:void(0);)

对于mysqld服务挂掉的情况无法实现自动切换，所以需要一个脚本来帮助我们完成，我这里有个简单的脚本，能实现当mysqld服务不可用时进行自动切换，当进行切换时发送邮件等。该脚本放在主服务器执行，也就是运行mysqld服务的服务器上执行。

[复制代码](javascript:void(0);)

[root@192.168.0.20 ~]# cat mysqlmon.sh

#!/bin/bash

trap 'echo PROGRAM INTERRUPTED; exit 1' INT

username=root

password=123456

n=0

log='/var/log/mysqlmon.log'

while true

do

if /usr/local/mysql/bin/mysql -u${username} -p${password} -e "use test" >&/dev/null

then

echo `date +"%Y-%m-%d %H:%M:%S"` mysqld is alive! >> ${log}

n=0

else

echo "`date +"%Y-%m-%d %H:%M:%S"` mysqld cannot be connected!" >> ${log}

n=$[n + 1]

if [ $n -eq 3 ]

then

/etc/init.d/heartbeat stop

echo "`date +"%Y-%m-%d %H:%M:%S"` mysqld switched to backup!" >> ${log}

echo "`date +"%Y-%m-%d %H:%M:%S"` mysqld switched to backup" | mutt -s "mysqld switched to backup" saltstack@163.com

break

fi

fi

sleep 10

done

[root@192.168.0.20 ~]#

[复制代码](javascript:void(0);)

挂在后台执行：

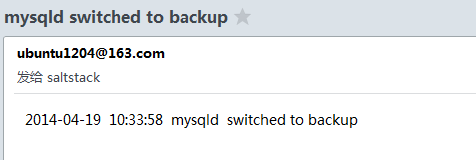
[root@192.168.0.10 ~]# nohup mysqlmon.sh &

停止mysqld服务，看是否进行切换以及发送邮件：

[root@192.168.0.10 ~]# /etc/init.d/mysqld stop

Shutting down MySQL. [ OK ]

[root@192.168.0.10 ~]#



[复制代码](javascript:void(0);)

[root@192.168.0.20 ~]# df -HT

Filesystem Type Size Used Avail Use% Mounted on

/dev/sda2 ext4 19G 4.9G 13G 28% /

tmpfs tmpfs 121M 0 121M 0% /dev/shm

/dev/sda1 ext4 204M 52M 141M 27% /boot

/dev/drbd0 ext4 33G 216M 32G 1% /data

[root@192.168.0.20 ~]# netstat -nltp | grep 3306

tcp 0 0 0.0.0.0:3306 0.0.0.0:\* LISTEN 13771/mysqld

[root@192.168.0.20 ~]#

[复制代码](javascript:void(0);)

## 总结

**搭建还不算复杂，但是也踩了不少坑，比如yum安装的heartbeat没有drbddisk脚本。该方案的优点是安全性高、稳定性高、可用性高，出现故障自动切换，但是缺点也很明显，只有一台服务器提供服务，成本相对较高。不方便扩展。可能会发生脑裂。当mysql服务挂掉或者不可用的情况下不能进行自动切换，需要通过crm模式实现或者额外的脚本实现(比如shell脚本监测到master的mysql不可用就将主上的heartbeat停掉，这样就会切换到backup中去)。监控也特别重要，可以使用nagios或者zabbix监控。**

# Mysql Cluster

MySQL Cluster是一个基于NDB Cluster存储引擎的完整的分布式数据库系统。不仅仅具有高可用性，而且可以自动切分数据，冗余数据等高级功能。和Oracle Real Cluster Application不太一样的是，MySQL Cluster 是一个Share Nothing的架构，各个MySQL Server之间并不共享任何数据，高度可扩展以及高度可用方面的突出表现是其最大的特色。虽然目前还只是MySQL家族中的一个新兴产品，但是已经有不少企业正在积极的尝试使用了。但是好像还不是很多，我常常听见人家问这玩意，包括一些企业，虽然该产品还不是很成熟，还有很多缺陷，但是我还是打算学习学习^\_^

## **MySQL Cluster 介绍**

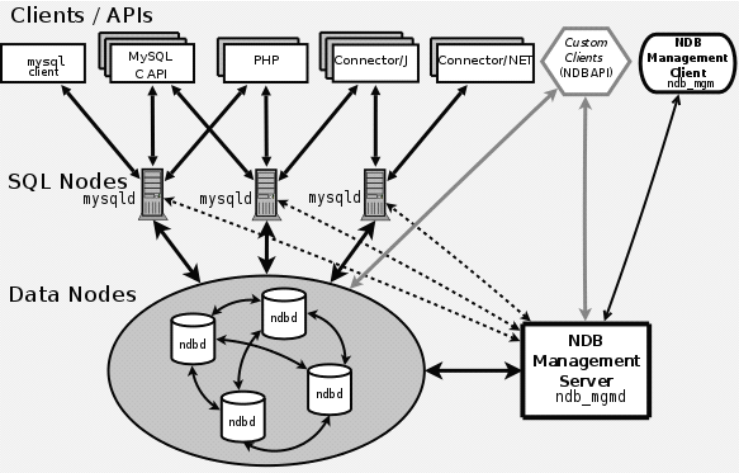
简单的说，MySQL Cluster 实际上是在无共享存储设备的情况下实现的一种完全分布式数据库系统，其主要通过 NDB Cluster（简称 NDB）存储引擎来实现。MySQL Cluster 刚刚诞生的时候可以说是一个可以对数据进行持久化的内存数据库，所有数据和索引都必须装载在内存中才能够正常运行，但是最新的 MySQL Cluster 版本已经可以做到仅仅将所有索引装载在内存中即可，实际的数据可以不用全部装载到内存中。

**一个 MySQL Cluster 的环境主要由以下三部分组成：**  
**（1）SQL 层的 SQL 服务器节点（后面简称为 SQL 节点）；**也就是我们常说的MySQL Server。主要负责实现一个数据库在存储层之上的所有事情，比如连接管理，Query 优化和响应 ，Cache 管理等等，只有存储层的工作交给了NDB 数据节点去处理了。也就是说，在纯粹的MySQL Cluster 环境中的SQL 节点，可以被认为是一个不需要提供任何存储引擎的MySQL服务器，因为他的存储引擎有Cluster 环境中的NDB 节点来担任。所以，SQL 层各MySQL服务器的启动与普通的MySQL Server 启动也有一定的区别，必须要添加ndbcluster参数选项才行。我们可以添加在my.cnf配置文件中，也可以通过启动命令行来指定。

**（2）Storage 层的 NDB 数据节点；**也就是上面说的NDB Cluster。最初的NDB是一个内存式存储引擎，当然也会将数据持久化到存储设备上。但是最新的NDB Cluster存储引擎已经改进了这一点，可以选择数据是全部加载到内存中还是仅仅加载索引数据。NDB 节点主要是实现底层数据存储功能，来保存Cluster 的数据。每一个Cluster节点保存完整数据的一个fragment，也就是一个数据分片（或者一份完整的数据，视节点数目和配置而定），所以只要配置得当，MySQL Cluster在存储层不会出现单点的问题。一般来说，NDB 节点被组织成一个一个的NDB Group，一个 NDB Group实际上就是一组存有完全相同的物理数据的NDB节点群。

上面提到了NDB 各个节点对数据的组织，可能每个节点都存有全部的数据也可能只保存一部分数据，主要是受节点数目和参数来控制的。首先在 MySQL Cluster主配置文件（在管理节点上面，一般为 config.ini）中，有一个非常重要的参数叫NoOfReplicas，这个参数指定了每一份数据被冗余存储在不同节点上面的份数，该参数一般至少应该被设置成2，也只需要设置成2就可以了。因为正常来说，两个互为冗余的节点同时出现故障的概率还是非常小的，当然如果机器和内存足够多的话，也可以继续增大来更进一步减小出现故障的概率。此外，一个节点上面是保存所有的数据还是一部分数据还受到存储节点数目的限制。NDB 存储引擎首先保证NoOfReplicas参数配置的要求来使用存储节点，对数据进行冗余，然后再根据节点数目将数据分段来继续使用多余的NDB节点。分段的数目为节点总数除以NoOfReplicas 所得。

**（3）负责管理各个节点的 Manage 节点主机；**管理节点负责整个Cluster集群中各个节点的管理工作,包括集群的配置,启动关闭各节点,对各个节点进行常规维护，以及实施数据的备份恢复等。管理节点会获取整个Cluster环境中各节点的状态和错误信息，并且将各 Cluster 集群中各个节点的信息反馈给整个集群中其他的所有节点。由于管理节点上保存了整个Cluster 环境的配置，同时担任了集群中各节点的基本沟通工作，所以他必须是最先被启动的节点。  
下面是一幅 MySQL Cluster 的基本架构图（[出自 MySQL 官方文档手册](http://dev.mysql.com/doc/refman/5.0/en/mysql-cluster-overview.html)）：



通过图中我们可以更清晰的了解整个 MySQL Cluster 环境各个节点以及客户端应用之间的关系。

## **MySQL Cluster 环境搭建**

搭建 MySQL Cluster首先需要至少一个管理节点主机来实现管理功能，一个SQL节点主机来实现MySQL server功能和两个ndb节点主机实现NDB Cluster的功能。我在这里测试使用双SQL节点来搭建测试环境，具体信息如下：

### 1、服务器准备

a) MySQL节点1 192.168.0.70

b) MySQL节点2 192.168.0.60

c) ndb节点1 192.168.0.50

d) ndb节点2 192.168.0.40

e) 管理节点 192.168.0.30

### 2、软件安装

测试环境（5台服务器均一样，不是必须的，所以服务器均已关闭iptables和selinux，生产环境请自行开放相关端口）

[root@localhost ~]# uname -a

Linux localhost.localdomain 2.6.32-220.el6.x86\_64 #1 SMP Tue Dec 6 19:48:22 GMT 2011 x86\_64 x86\_64 x86\_64 GNU/Linux

#### **安装 MySQL 节点：**

sql节点1： 192.168.0.70  
sql节点2： 192.168.0.60

下载安装包：mysql-cluster-gpl-7.2.4-linux2.6-x86\_64.tar.gz，我这里使用二进制编译好了的，同学们可以自己下载源码包编译。这里操作一台SQL节点服务器，另外一台SQL节点服务器也是相同的，都执行如下安装步骤。

[root@192.168.0.70 ~]# wget https://downloads.skysql.com/archives/mysql-cluster-gpl-7.2/mysql-cluster-gpl-7.2.4-linux2.6-x86\_64.tar.gz

[复制代码](javascript:void(0);)

[root@192.168.0.70 ~]# groupadd mysql

[root@192.168.0.70 ~]# useradd -r -g mysql mysql

[root@192.168.0.70 ~]# tar xf mysql-cluster-gpl-7.2.4-linux2.6-x86\_64.tar.gz -C /usr/local/

[root@192.168.0.70 ~]# cd /usr/local/

[root@192.168.0.70 local]# ln -s mysql-cluster-gpl-7.2.4-linux2.6-x86\_64 mysql

[root@192.168.0.70 local]# cd mysql

[root@192.168.0.70 mysql]# chown -R mysql .

[root@192.168.0.70 mysql]# chgrp -R mysql .

[root@192.168.0.70 mysql]# mkdir /data/mysql

[root@192.168.0.70 mysql]# chown -R mysql.mysql /data/mysql/

[root@192.168.0.70 mysql]# \cp support-files/my-large.cnf /etc/my.cnf

[root@192.168.0.70 mysql]# /usr/local/mysql/scripts/mysql\_install\_db --user=mysql --datadir=/data/mysql/ --basedir=/usr/local/mysql

[复制代码](javascript:void(0);)

[root@192.168.0.70 mysql]# chown -R root .

[root@192.168.0.70 mysql]# cp support-files/mysql.server /etc/init.d/mysqld

[root@192.168.0.70 mysql]# chmod 755 /etc/init.d/mysqld

[root@192.168.0.70 mysql]# echo "export PATH=$PATH:/usr/local/mysql/bin" >> /root/.bash\_profile #添加环境变量，执行命令方便一点

[root@192.168.0.70 mysql]# source /root/.bash\_profile

#### **SQL节点配置**

修改/etc/my.cnf配置文件，追加如下配置

[复制代码](javascript:void(0);)

[mysqld]

datadir=/data/mysql

basedir= /usr/local/mysql

ndbcluster # 运行NDB存储引擎

ndb-connectstring=192.168.0.30 # 管理节点

[MYSQL\_CLUSTER]

ndb-connectstring=192.168.0.30 #管理节点

[复制代码](javascript:void(0);)

#### **NDB节点安装（数据节点）**

数据节点1： 192.168.0.50  
数据节点2： 192.168.0.40

如果希望尽可能的各环境保持一致，建议在NDB节点也和SQL节点一样安装整个带有 NDB Cluster 存储引擎的MySQL Server。（NDB节点可以不用初始化数据，自己已经测试，但是我依然会初始化）安装细节和上面的SQL节点完全一样。两台NDB节点操作一样，如下：

[复制代码](javascript:void(0);)

[root@192.168.0.40 ~]# groupadd mysql

[root@192.168.0.40 ~]# useradd -r -g mysql mysql

[root@192.168.0.40 ~]# tar xf mysql-cluster-gpl-7.2.4-linux2.6-x86\_64.tar.gz -C /usr/local/

[root@192.168.0.40 ~]# cd /usr/local/

[root@192.168.0.40 local]# ln -s mysql-cluster-gpl-7.2.4-linux2.6-x86\_64 mysql

[root@192.168.0.40 local]# cd mysql

[root@192.168.0.40 mysql]# chown -R mysql .

[root@192.168.0.40 mysql]# chgrp -R mysql .

[root@192.168.0.40 mysql]# mkdir /data/mysql

[root@192.168.0.40 mysql]# chown -R mysql.mysql /data/mysql/

[root@192.168.0.40 mysql]# /usr/local/mysql/scripts/mysql\_install\_db --user=mysql --datadir=/data/mysql/ --basedir=/usr/local/mysql

[复制代码](javascript:void(0);)

[复制代码](javascript:void(0);)

[root@192.168.0.40 mysql]# chown -R root .

[root@192.168.0.40 mysql]# \cp support-files/my-large.cnf /etc/my.cnf

[root@192.168.0.40 mysql]# cp support-files/mysql.server /etc/init.d/mysqld

[root@192.168.0.40 mysql]# chmod 755 /etc/init.d/mysqld

[root@192.168.0.40 mysql]# echo "export PATH=$PATH:/usr/local/mysql/bin" >> /root/.bash\_profile

[root@192.168.0.40 mysql]# source /root/.bash\_profile

[root@192.168.0.40 mysql]#

[复制代码](javascript:void(0);)

[root@192.168.0.40 mysql]# mkdir /data/mysql-cluster/data -p

**上面这个目录用来存放NDB节点的数据，在管理节点里面也要配置为该目录，配置其他的目录会报错，说无法创建各种日志文件。上面提到的/data/mysql是用来在没使用NDB时存放的数据，和平时我们使用的mysql没有区别。**

#### **NDB节点配置(数据节点)：**

和SQL节点是一样的，修改/etc/my.cnf，追加如下内容：

[复制代码](javascript:void(0);)

[mysqld]

datadir=/data/mysql

basedir= /usr/local/mysql

ndbcluster # 运行NDB存储引擎

ndb-connectstring=192.168.0.30 # 管理节点

[MYSQL\_CLUSTER]

ndb-connectstring=192.168.0.30 #管理节点

[复制代码](javascript:void(0);)

#### **安装管理节点**

管理节点所需要的安装更简单，实际上只需要 ndb\_mgm 和ndb\_mgmd两个程序即可，这两个可执行程序可以在上面的MySQL节点的MySQL安装目录中的bin目录下面找到。将这两个程序copy到管理节点上面合适的位置（自行考虑，我一般会放在/usr/local/mysql/bin下面）并且添加环境变量就可以了。

[复制代码](javascript:void(0);)

[root@192.168.0.30 ~]# mkdir /usr/local/mysql/bin -p

[root@192.168.0.30 ~]# scp 192.168.0.70:/usr/local/mysql/bin/ndb\_mgm /usr/local/mysql/bin/

root@192.168.0.70's password:

ndb\_mgm 100% 6213KB 6.1MB/s 00:00

[root@192.168.0.30 ~]# scp 192.168.0.70:/usr/local/mysql/bin/ndb\_mgmd /usr/local/mysql/bin/

root@192.168.0.70's password:

ndb\_mgmd 100% 14MB 6.9MB/s 00:02

[root@192.168.0.30 ~]# echo "export PATH=$PATH:/usr/local/mysql/bin" >> /root/.bash\_profile

[root@192.168.0.30 ~]# source /root/.bash\_profile

[root@192.168.0.30 ~]#

[复制代码](javascript:void(0);)

#### **管理节点配置：**

1.在/data/创建目录mysql-cluster，该目录会存放相关日志文件，以及pid号。并在目录中创建配置文件config.ini

[root@192.168.0.30 ~]# mkdir /data/mysql-cluster

[root@192.168.0.30 ~]# cd /data/mysql-cluster/

[root@192.168.0.30 mysql-cluster]# touch config.ini

2.根据我们上面提供的环境，config.ini文件配置如下（在安装目录下面也有样例配置文件可以参考），详细的配置参数请阅读这里[MySQL Cluster配置详解](http://www.cnblogs.com/gomysql/p/3668287.html)

[复制代码](javascript:void(0);)

[root@192.168.0.30 ~]# cat /data/mysql-cluster/config.ini

[NDBD DEFAULT]

NoOfReplicas=1 #每个数据节点的镜像数量，通常最低设置为2，否则就没有意义了，这里是方便后面测试。

DataMemory=64M #每个数据节点中给数据分配的内存

IndexMemory=16M #每个数据节点中给索引分配的内存

#管理节点

[NDB\_MGMD]

nodeid=1

hostname=192.168.0.30 #管理节点ip

datadir=/data/mysql-cluster #管理节点数据目录,存放相关日志，以及pid文件

#第一个 ndbd 节点:

[NDBD]

nodeid=2

hostname=192.168.0.50 #数据节点ip地址

datadir=/data/mysql-cluster/data

#第二个 ndbd 节点:

[NDBD]

nodeid=3

hostname=192.168.0.40

datadir=/data/mysql-cluster/data #NDB点数据存放目录

# SQL node options:

[MySQLD]

nodeid=4

hostname=192.168.0.70 #SQL节点ip地址

[MySQLD]

nodeid=5

hostname=192.168.0.60

[MySQLD] 这里保留了一个空节点。否则停止NDB会报错：No free node id found for ndbd(NDB).

[root@192.168.0.30 ~]#

[复制代码](javascript:void(0);)

在上面的配置文件中，包括很多的组，组名用"[]"括起来，这里我们最关心的是3类节点组的配置，分别定义如下：

**[NDB\_MGMD] 表示管理节点的配置，只能有一个。**

**[NDBD DEFAULT] 表示每个数据节点的默认配置，在每个节点的[NDBD]中不用再写这些选项，只能有一个。**

**[NDBD] 表示每个数据节点的配置，可以有多个。**

**[MYSQLD] 表示SQL节点的配置，可以有多个，分别写上不同的SQL节点的ip地址；也可以不用写，只保留一个空节点，表示任意一个ip地址都可以进行访问。此节点的个数表明了可以用来连接数据节点的SQL节点总数。**

每个节点都有一个独立的id号，可以填写，比如nodeid=2，（老版本使用id，新版本已经不使用id标识了）也可以不用填写，系统会按照配置文件的填写顺序自动分配。

## **开始使用Cluster**

上面我们都已经配置完毕了，下面说明启动，关闭和使用方法

**启动顺序为：管理节点->数据节点->SQL节点（很重要）**

**（1）启动管理节点：**

[复制代码](javascript:void(0);)

[root@192.168.0.30 ~]# ndb\_mgmd -f /data/mysql-cluster/config.ini

MySQL Cluster Management Server mysql-5.5.19 ndb-7.2.4

[root@192.168.0.30 ~]# netstat -ntlp | grep 1186

tcp 0 0 0.0.0.0:1186 0.0.0.0:\* LISTEN 1329/ndb\_mgmd

[root@192.168.0.30 ~]# ps -ef | grep ndb\_mgmd | grep -v grep

root 1329 1 0 21:50 ? 00:00:00 ndb\_mgmd -f /data/mysql-cluster/config.ini

[root@192.168.0.30 ~]#

[复制代码](javascript:void(0);)

**（2）启动NDB（数据节点）**

**注意：只是在第一次启动或在备份/恢复或配置变化后重启ndbd时，才加–initial参数！原因在于，该参数会使节点删除由早期ndbd实例创建的，用于恢复的任何文件，包括用于恢复的日志文件。**

[root@192.168.0.40 ~]# ndbd --initial

2014-04-15 21:51:51 [ndbd] INFO -- Angel connected to '192.168.0.30:1186'

2014-04-15 21:51:51 [ndbd] INFO -- Angel allocated nodeid: 3

[root@192.168.0.40 ~]#

[root@192.168.0.50 ~]# ndbd --initial

2014-04-15 21:52:29 [ndbd] INFO -- Angel connected to '192.168.0.30:1186'

2014-04-15 21:52:29 [ndbd] INFO -- Angel allocated nodeid: 2

[root@192.168.0.50 ~]#

查看是否有相关进程：

[root@192.168.0.50 ~]# ps -ef | grep ndbd | grep -v grep

root 1879 1 0 21:52 ? 00:00:00 ndbd --initial

root 1880 1879 2 21:52 ? 00:00:03 ndbd --initial

[root@192.168.0.50 ~]#

[root@192.168.0.40 ~]# ps -ef | grep ndbd | grep -v grep

root 2266 1 0 21:51 ? 00:00:00 ndbd --initial

root 2267 2266 1 21:51 ? 00:00:04 ndbd --initial

[root@192.168.0.40 ~]#

ndbd进程是使用NDB存储引擎处理表中数据的进程。通过该进程，存储节点能够实现分布式事务管理，节点恢复，在线备份相关任务。

**（3）启动SQL节点（启动mysql服务）**

本文中是192.168.0.60，192.168.0.70两个节点

[root@192.168.0.60 ~]# /etc/init.d/mysqld start

Starting MySQL [ OK ]

[root@192.168.0.60 ~]#

[root@192.168.0.70 ~]# /etc/init.d/mysqld start

Starting MySQL..... [ OK ]

[root@192.168.0.70 ~]#

**（4）节点全部启动成功后，在管理节点使用ndb\_mgm工具的show命令查看集群状态：**

[复制代码](javascript:void(0);)

[root@192.168.0.30 ~]# ndb\_mgm

-- NDB Cluster -- Management Client --

ndb\_mgm> show

Connected to Management Server at: localhost:1186

Cluster Configuration

---------------------

[ndbd(NDB)] 2 node(s)

id=2 @192.168.0.50 (mysql-5.5.19 ndb-7.2.4, Nodegroup: 0, Master)

id=3 @192.168.0.40 (mysql-5.5.19 ndb-7.2.4, Nodegroup: 1)

[ndb\_mgmd(MGM)] 1 node(s)

id=1 @192.168.0.30 (mysql-5.5.19 ndb-7.2.4)

[mysqld(API)] 3 node(s)

id=4 @192.168.0.70 (mysql-5.5.19 ndb-7.2.4)

id=5 @192.168.0.60 (mysql-5.5.19 ndb-7.2.4)

id=6 (not connected, accepting connect from any host)

ndb\_mgm>

[复制代码](javascript:void(0);)

ndb\_mgm工具是ndb\_mgmd（MySQL Cluster Server）的客户端管理工具，通过该工具可以方便的检查Cluster的状态，启动备份，关闭等功能。更详细的方法可以通过ndb\_mgm --help命令来进行查看。

从上面显示的状态可以看出如下信息。

（1）集群目前的管理服务器端口是1186

Connected to Management Server at: localhost:1186

（2）集群的数据节点（NDB）有2个，详细信息：

[ndbd(NDB)] 2 node(s)

id=2 @192.168.0.50 (mysql-5.5.19 ndb-7.2.4, Nodegroup: 0, Master)

id=3 @192.168.0.40 (mysql-5.5.19 ndb-7.2.4, Nodegroup: 1)

（3）集群的管理节点有一个，详细信息：

[ndb\_mgmd(MGM)] 1 node(s)

id=1 @192.168.0.30 (mysql-5.5.19 ndb-7.2.4)

（4）SQL节点有3个，目前处于连接状态的有2个，详细信息：

[mysqld(API)] 3 node(s)

id=4 @192.168.0.70 (mysql-5.5.19 ndb-7.2.4)

id=5 @192.168.0.60 (mysql-5.5.19 ndb-7.2.4)

id=6 (not connected, accepting connect from any host)

到这里MySQL Cluster就已经搭建完成了。接下来就到测试时间咯。^\_^

## **MySQL Cluster 高可用测试**

成功启动后，下面来测试一下Cluster的功能。如果要使用cluster，则表的引擎必须为NDB，其他类型存储引擎的数据不会保存到数据节点中。对于cluster的一个重要功能就是防止单点故障。我们下面对这些问题分别来进行测试。

**1.NDB存储引擎测试**

（1）在任意一个SQL节点（我这里选择192.168.0.70）的test库中创建测试表t1，设置存储引擎为NDB，并插入两条测试数据：

[复制代码](javascript:void(0);)

mysql> create table t1 (

-> id int,

-> name varchar(**20**)

-> )

-> engine=ndb

-> ;

Query OK, **0** rows affected (**0.44** sec)

mysql> insert into t1 select **1**,'yayun';

Query OK, **1** row affected (**0.11** sec)

Records: **1** Duplicates: **0** Warnings: **0**

mysql> insert into t1 select **1**,'atlas';

Query OK, **1** row affected (**0.03** sec)

Records: **1** Duplicates: **0** Warnings: **0**

mysql>

[复制代码](javascript:void(0);)

（2）在另外一个SQL节点（192.168.0.60）查询test库中t1表，结果如下：

[复制代码](javascript:void(0);)

mysql> select \* from test.t1;

+------+-------+

| id | name |

+------+-------+

| **1** | atlas |

| **1** | yayun |

+------+-------+

**2** rows in set (**0.08** sec)

mysql>

[复制代码](javascript:void(0);)

显然，两个SQL节点查询的数据时一致的。

（3）在SQL节点192.168.0.70上把测试表t1引擎改为MyISAM，再次插入测试数据：

[复制代码](javascript:void(0);)

mysql> alter table t1 engine=myisam;

Query OK, **2** rows affected (**0.50** sec)

Records: **2** Duplicates: **0** Warnings: **0**

mysql> insert into t1 select **2**,'good boy';

Query OK, **1** row affected (**0.00** sec)

Records: **1** Duplicates: **0** Warnings: **0**

mysql>

[复制代码](javascript:void(0);)

（4）在SQL节点192.168.0.60上再次查询表t1，结果如下：

mysql> select \* from t1;

ERROR **1146** (42S02): Table 'test.t1' doesn't exist

mysql> show tables;

Empty set (0.04 sec)

mysql>

直接报错，说表不存在了。（老版本是报ERROR 1412:Table definition ha  changed,please retry transaction）

（5）我们再次改回NDB引擎。

mysql> alter table t1 engine=ndb;

Query OK, **3** rows affected (**0.25** sec)

Records: **3** Duplicates: **0** Warnings: **0**

mysql>

（6）再次进行查询如下：

[复制代码](javascript:void(0);)

mysql> select \* from t1;

+------+----------+

| id | name |

+------+----------+

| **1** | atlas |

| **2** | good boy |

| **1** | yayun |

+------+----------+

**3** rows in set (**0.02** sec)

mysql>

[复制代码](javascript:void(0);)

发现表t1的数据再次同步到了数据节点。所有SQL节点又都可以正常查询数据。

**2.单点故障测试**

对于任意一种节点，都存在单点故障的可能性。在cluster的设置过程中，应该尽量对每一类节点设置冗余，以防止单点故障发生时造成的应用终端。对于管理节点，一般不需要特殊的配置，只需要将管理工具和配置文件防止多台服务器上即可。下面我们测试一下SQL节点和NDB（数据节点）的单点故障。

**SQL节点发生单点故障**

对于上面的测试环境中，我们设置了两个SQL节点，应用从两个节点对数据访问都可以得到一致的结果。如果有一个节点故障，系统会正常运行吗？我们测试便知。

（1）将SQL节点192.168.0.60上的MySQL服务停止：

[root@192.168.0.60 ~]# /etc/init.d/mysqld stop

Shutting down MySQL.. [ OK ]

[root@192.168.0.60 ~]#

（2）查看cluster状态：

[复制代码](javascript:void(0);)

ndb\_mgm> show

Cluster Configuration

---------------------

[ndbd(NDB)] **2** node(s)

id=**2** @192.**168.0**.**50** (mysql-**5.5**.**19** ndb-**7.2**.**4**, Nodegroup: **0**, Master)

id=**3** @192.**168.0**.**40** (mysql-**5.5**.**19** ndb-**7.2**.**4**, Nodegroup: **1**)

[ndb\_mgmd(MGM)] **1** node(s)

id=**1** @192.**168.0**.**30** (mysql-**5.5**.**19** ndb-**7.2**.**4**)

[mysqld(API)] **3** node(s)

id=**4** @192.**168.0**.**70** (mysql-**5.5**.**19** ndb-**7.2**.**4**)

**id=5 (not connected, accepting connect from 192.168.0.60)**

id=**6** (not connected, accepting connect from any host)

ndb\_mgm>

[复制代码](javascript:void(0);)

可以发现SQL节点192.168.0.60已经断开，但是另外一个SQL节点192.168.0.70仍然处于正常状态。

（3）从SQL节点192.168.0.70上查看表t1，结果如下：

[复制代码](javascript:void(0);)

mysql> select \* from t1;

+------+----------+

| id | name |

+------+----------+

| **2** | good boy |

| **1** | yayun |

| **1** | atlas |

+------+----------+

**3** rows in set (**0.01** sec)

mysql>

[复制代码](javascript:void(0);)

显然，SQL节点的单点故障并没有引起数据查询的故障。对于应用来说，需要改变的就是将以前对故障节点的访问改为对非故障节点的访问（SQL节点前面可以挂LVS，然后做各种检测）

**NDB（数据节点）的单点故障**

在这个测试环境中，数据节点也是两个，那么他们对数据的存储是互相镜像还是一份数据分成几块存储呢？（类似磁盘阵列RAID1还是RAID0）？这个答案关键在于配置文件中[NDBD DEFAULT]组中的NoOfReplicas参数，如果这个参数等于1，表示只有一份数据，但是分成N块分别存储在N个数据节点上，如果该值等于2，则表示数据被分成N/2,每块数据都有两个备份，这样即使有任意一个节点发生故障，只要它的备份节点正常，数据就可以正常查询。

在下面的例子中，先将两个数据节点之一停止，访问表t1，看能否正常访问；然后将NoOfReplicas配置改为2，这时，数据节点实际上已经互为镜像，保存了两份。这时再停止任意一个数据节点，看是否能访问表t1。

（1）将数据节点192.168.0.40上的NDB进程停止：

[复制代码](javascript:void(0);)

[root@192.168.0.40 ~]# ps -ef | grep ndbd

root **2266** **1** **0** **21**:**51** ? **00**:**00**:**00** ndbd --initial

root **2267** **2266** **1** **21**:**51** ? **00**:**01**:**03** ndbd --initial

root **2368** **1300** **0** **23**:**06** pts/**0** **00**:**00**:**00** grep ndbd

[root@192.168.0.40 ~]# pkill -**9** ndbd

[root@192.168.0.40 ~]# ps -ef | grep ndbd

root **2371** **1300** **0** **23**:**07** pts/**0** **00**:**00**:**00** grep ndbd

[root@192.168.0.40 ~]#

[复制代码](javascript:void(0);)

（2）在任意节点，这里是192.168.0.70查看表t1的数据：

mysql> select \* from t1;

ERROR **1296** (HY000): Got error **157** 'Unknown error code' from NDBCLUSTER

mysql>

显然无法访问表t1的数据了。

（3）将配置文件中的NoOfReplicas改为2，按照前面的步骤重新启动集群：

[root@192.168.0.30 ~]# grep 'NoOfReplicas' /data/mysql-cluster/config.ini

NoOfReplicas=2 #每个数据节点的镜像数量

[root@192.168.0.30 ~]#

**最后发现无法启动数据节点，查看错误日志如下：**

[复制代码](javascript:void(0);)

Time: Tuesday 15 April 2014 - 23:22:51

Status: Permanent error, external action needed

Message: Invalid configuration received from Management Server (Configuration error)

Error: 2350

Error data: **Illegal configuration change. Initial start needs to be performed when changing no of replicas (1 != 2)**

Error object: DBDIH (Line: 4820) 0x00000002

Program: ndbd

[复制代码](javascript:void(0);)

**看来NoOfReplicas参数无法临时更改，我们开始就需要设置好，不要到后面才想到更改，那时就悲剧了。**

**如果重新ndbd --initial,将会丢失所有数据，好吧，记住这个坑，下次就不会踩进去了。**

[root@192.168.0.40 ~]# ndbd --initial

2014-04-15 23:36:08 [ndbd] INFO -- Angel connected to '192.168.0.30:1186'

2014-04-15 23:36:08 [ndbd] INFO -- Angel allocated nodeid: 3

[root@192.168.0.40 ~]#

[root@192.168.0.50 ~]# ndbd --initial

2014-04-15 23:38:50 [ndbd] INFO -- Angel connected to '192.168.0.30:1186'

2014-04-15 23:38:50 [ndbd] INFO -- Angel allocated nodeid: 2

[root@192.168.0.50 ~]#

重新建表插入数据再测试吧，囧..............

[复制代码](javascript:void(0);)

mysql> create table t1 (name varchar(**20**))engine=ndb;

Query OK, **0** rows affected (**0.31** sec)

mysql> insert into t1 select 'yayun';

Query OK, **1** row affected (**0.04** sec)

Records: **1** Duplicates: **0** Warnings: **0**

mysql> insert into t1 select 'atlas';

Query OK, **1** row affected (**0.06** sec)

Records: **1** Duplicates: **0** Warnings: **0**

mysql> select \* from t1;

+-------+

| name |

+-------+

| atlas |

| yayun |

+-------+

**2** rows in set (**0.01** sec)

mysql>

[复制代码](javascript:void(0);)

查看cluster状态：

[复制代码](javascript:void(0);)

[root@192.168.0.30 ~]# ndb\_mgm

-- NDB Cluster -- Management Client --

ndb\_mgm> show

Connected to Management Server at: localhost:**1186**

Cluster Configuration

---------------------

[ndbd(NDB)] **2** node(s)

id=**2** @192.**168.0**.**50** (mysql-**5.5**.**19** ndb-**7.2**.**4**, Nodegroup: **0**, Master)

id=**3** @192.**168.0**.**40** (mysql-**5.5**.**19** ndb-**7.2**.**4**, Nodegroup: **0**)

[ndb\_mgmd(MGM)] **1** node(s)

id=**1** @192.**168.0**.**30** (mysql-**5.5**.**19** ndb-**7.2**.**4**)

[mysqld(API)] **3** node(s)

id=**4** @192.**168.0**.**70** (mysql-**5.5**.**19** ndb-**7.2**.**4**)

id=**5** @192.**168.0**.**60** (mysql-**5.5**.**19** ndb-**7.2**.**4**)

id=**6** (not connected, accepting connect from any host)

ndb\_mgm>

[复制代码](javascript:void(0);)

一切正常。我们现在停止NDB节点192.168.0.50，看是否还能访问数据：

[复制代码](javascript:void(0);)

[root@192.168.0.50 ~]# ps -ef | grep ndbd

root **2119** **1** **0** **23**:**38** ? **00**:**00**:**00** ndbd --initial

root **2120** **2119** **2** **23**:**38** ? **00**:**00**:**10** ndbd --initial

root **2161** **1275** **0** **23**:**45** pts/**0** **00**:**00**:**00** grep ndbd

[root@192.168.0.50 ~]# pkill -**9** ndbd

[root@192.168.0.50 ~]# ps -ef | grep ndbd

root **2164** **1275** **0** **23**:**45** pts/**0** **00**:**00**:**00** grep ndbd

[root@192.168.0.50 ~]#

[复制代码](javascript:void(0);)

[复制代码](javascript:void(0);)

ndb\_mgm> show

Cluster Configuration

---------------------

[ndbd(NDB)] **2** node(s)

id=**2** (not connected, accepting connect from **192.168**.**0.50**)

id=**3** @192.**168.0**.**40** (mysql-**5.5**.**19** ndb-**7.2**.**4**, Nodegroup: **0**, Master)

[ndb\_mgmd(MGM)] **1** node(s)

id=**1** @192.**168.0**.**30** (mysql-**5.5**.**19** ndb-**7.2**.**4**)

[mysqld(API)] **3** node(s)

id=**4** @192.**168.0**.**70** (mysql-**5.5**.**19** ndb-**7.2**.**4**)

id=**5** @192.**168.0**.**60** (mysql-**5.5**.**19** ndb-**7.2**.**4**)

id=**6** (not connected, accepting connect from any host)

ndb\_mgm>

[复制代码](javascript:void(0);)

看见只有一个NDB节点在运行。

访问任意SQL节点查询数据,我这里用192.168.0.70，查询结果如下：

[复制代码](javascript:void(0);)

mysql> select \* from t1;

+-------+

| name |

+-------+

| atlas |

| yayun |

+-------+

**2** rows in set (**0.01** sec)

mysql>

[复制代码](javascript:void(0);)

显然挂掉一个NDB节点不影响我们正常的数据查询，数据节点的冗余同样防止了单点故障。

## **MySQL Cluster 集群的关闭**

**关闭顺序：SQL节点->数据节点->管理节点（在MySQL Cluster环境中，NDB节点和管理节点的关闭都可以在管理节点的管理程序中完成，也可以分节点关闭，但是SQL节点却没办法。所以，在关闭整个MySQL Cluster环境或者关闭某个SQL节点的时候，首先必须到SQL节点主机上来关闭SQL节点程序。关闭方法和MySQL Server的关闭一样。）**

（1）SQL节点关闭

[root@192.168.0.70 ~]# /etc/init.d/mysqld stop

Shutting down MySQL.. [ OK ]

[root@192.168.0.70 ~]#

（2）（NDB）数据节点关闭

[root@192.168.0.50 ~]# ndbd stop

2014-04-15 23:54:36 [ndbd] INFO -- Angel connected to '192.168.0.30:1186'

2014-04-15 23:54:36 [ndbd] INFO -- Angel allocated nodeid: 2

[root@192.168.0.50 ~]#

（3）管理节点关闭

[复制代码](javascript:void(0);)

ndb\_mgm> shutdown

Node 2: Cluster shutdown initiated

Node 3: Cluster shutdown initiated

3 NDB Cluster node(s) have shutdown.

Disconnecting to allow management server to shutdown.

Node 3: Node shutdown completed.

ndb\_mgm>

[复制代码](javascript:void(0);)

## **总结**

慢慢的学习过程中，踩了不少坑，比如参数NoOfReplicas无法临时更改，管理节点配置文件中如果不多预留一个[MySQLD]，在停止NDB节点时会报错，以及配置文件中的[NDBD]段落中的datadir指定的目录在数据节点的服务器上面要存在。以及selinux，iptables等相关问题。总之收获满满。后续的文章中将会介绍mysql cluster的日常维护，包括数据备份，数据恢复，日志管理等。MySQL Cluster的核心在于NDB Cluster存储引擎，不仅对数据进行了水平切分，还对数据进行了跨节点冗余。既解决了数据库的扩展问题，同时也在很大程度上提高了数据库整体可用性。