## Mysql-mmm配置全自动切换主从关系

1、MySQL-MMM优缺点

优点：高可用性，扩展性好，出现故障自动切换，对于主主同步，在同一时间只提供一台数据库写操作，保证的数据的一致性。

缺点：Monitor节点是单点，可以结合Keepalived实现高可用。

2、MySQL-MMM工作原理

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

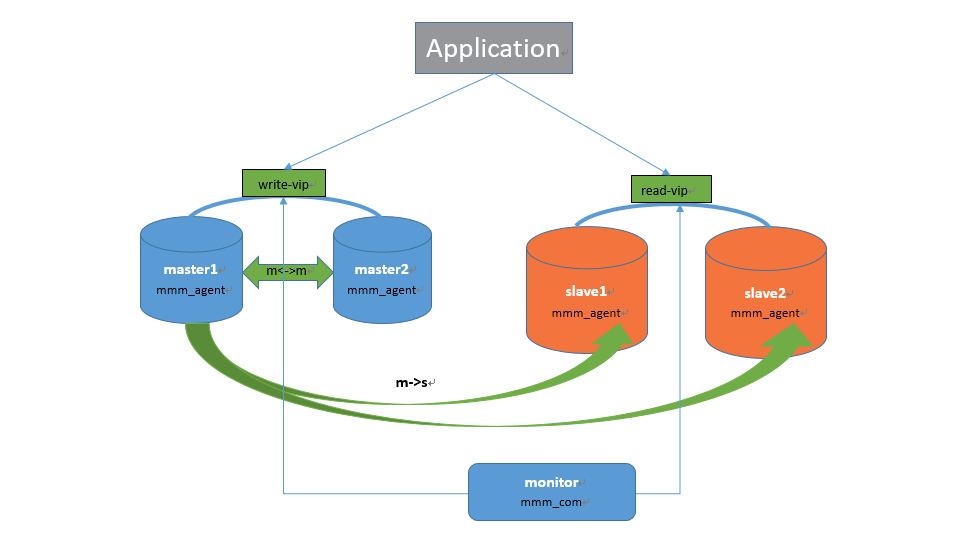
mmm\_mond：监控进程，负责所有的监控工作，决定和处理所有节点角色活动。此脚本需要在监管机上运行。

mmm\_agentd：运行在每个mysql服务器上的代理进程，完成监控的探针工作和执行简单的远端服务设置。此脚本需要在被监管机上运行。

mmm\_control：一个简单的脚本，提供管理mmm\_mond进程的命令。

mysql-mmm的监管端会提供多个虚拟IP（VIP），包括一个可写VIP，多个可读VIP，通过监管的管理，这些IP会绑定在可用mysql之上，当某一台mysql宕机时，监管会将VIP迁移至其他mysql。

在整个监管过程中，需要在mysql中添加相关授权用户，以便让mysql可以支持监理机的维护。授权的用户包括一个mmm\_monitor用户和一个mmm\_agent用户，如果想使用mmm的备份工具则还要添加一个mmm\_tools用户



# 一、概述

## 1.1配置结构说明

操作系统：CentOS release 6.5 (Final)

软件信息：mysql-5.5.3-m3.tar.gz

mysql-mmm-2.2.1.tar.gz

服务IP

db1：10.0.2.208 主

db2：10.0.2.209 主

db3：10.0.2.211 从（并提供心跳监听）

虚IP：

10.0.2.196 write

10.0.2.213 read

10.0.2.214 read

db1：10.0.2.208 主 db2：10.0.2.209 主

互为主从

安装mysql-5.5.3-m3.tar.gz 安装mysql-5.5.3-m3.tar.gz

安装mysql-mmm-2.2.1.tar.gz 安装mysql-mmm-2.2.1.tar.gz

主从

安装mysql-5.5.3-m3.tar.gz

安装mysql-mmm-2.2.1.tar.gz

db3：10.0.2.211 从（并提供心跳监听）

10.0.2.196 (write) 10.0.2.213 (read) 10.0.2.214 (read)

# 二、安装和配置mysql

## 2.1获得程序

Mysql:堡垒机211.100.97.116 的 /home/lushanlin/soft 目录中

wget http://mysql-mmm.org/\_media/:mmm2:mysql-mmm-2.2.1.tar.gz -O mysql-mmm-2.2.1.tar.gz

## 2.2 安装mysql程序

**创建用户：**

groupadd mysql

useradd -g mysql mysql

**解压：**

tar xvf mysql-5.5.3-m3.tar.gz -C /var/tmp/

**目录：**

cd /var/tmp/mysql-5.5.3/

**设置编译环境并指定目录：**

./configure --prefix=/usr/local/mysql --enable-thread-safe-client --enable-local-infile --with-charset=gbk --with-extra-charsets="gbk gb2312 big5 utf8" --with-low-memory

**编译和安装：**

make && make install

**基本设置：**

cp support-files/my-medium.cnf /etc/my.cnf

cd /usr/local/mysql/

chown -R mysql:mysql \*

/usr/local/mysql/bin/mysql\_install\_db --user=mysql --datadir=/usr/local/mysql/var

cp /usr/local/mysql/share/mysql/mysql.server /etc/init.d/mysqld

chmod 755 /etc/init.d/mysqld

chkconfig --add mysqld

chkconfig mysqld on

export PATH=$PATH:/usr/local/mysql/bin

echo "PATH=$PATH:/usr/local/mysql/bin" >> /etc/profile

service mysqld restart

/usr/local/mysql/bin/mysqladmin -u root password izptec

## 2.3 配置主主和主从关系

**配置主主：**

DB1：编辑/etc/my.cnf，加入以下内容：

server-id=1  
log-bin=mysql-bin  
binlog\_format=ROW  
log-slave-updates  
sync\_binlog=1  
auto\_increment\_increment=2  
auto\_increment\_offset=1  
skip\_slave\_start

DB2：编辑/etc/my.cnf，加入以下内容：

server-id=2  
log-bin=mysql-bin  
binlog\_format=ROW  
log-slave-updates  
sync\_binlog=1  
auto\_increment\_increment=2  
auto\_increment\_offset=2

**配置主从：**

DB3：编辑/etc/my.cnf，加入以下内容：

server-id=3  
log-bin=mysql-bin  
log-slave-updates

**重启mysql服务**

service mysqld restart

## 2.3.1 创建授权账号

**DB1授权：**

mysql -uroot -pizptec -e "GRANT REPLICATION SLAVE ON \*.\* TO 'slave'@'10.10.4.17' IDENTIFIED BY 'slave'; "

mysql -uroot -pizptec -e "GRANT REPLICATION SLAVE ON \*.\* TO 'slave'@'10.10.4.18' IDENTIFIED BY 'slave'; "

**DB2授权：**

mysql -uroot -pizptec -e "GRANT REPLICATION SLAVE ON \*.\* TO 'slave'@'10.10.4.18' IDENTIFIED BY 'slave'; "

mysql -uroot -pizptec -e "GRANT REPLICATION SLAVE ON \*.\* TO 'slave'@'10.10.4.32' IDENTIFIED BY 'slave'; "

DB3的话以谁为主都无所谓，当主出现故障的时候，mmm会自动切换主的，我这里就以DB2为主，配置方从的步骤：锁表——（主）导出数据——（主）查看指针位置——（主）解锁——（从）导入数据——（从）导入主服务器的指针位置——（双方）开启主从线程

## 2.3.2 创建主主和主从

**查看DB2的指针位置：**

mysql> flush tables with read lock;  
mysql> show master status;  
+------------------+----------+--------------+------------------+  
| File | Position | Binlog\_Do\_DB | Binlog\_Ignore\_DB |  
+------------------+----------+--------------+------------------+  
| mysql-bin.000004 | 586 | | |  
+------------------+----------+--------------+------------------+  
1 row in set (0.00 sec)  
mysql> unlock tables;

**得出指针位置之后在DB1和DB3上操作：**

mysql> change master to  
-> master\_host='10.10.4.17',  
-> master\_user='slave',  
-> master\_password='slave',  
-> master\_log\_file='mysql-bin.000005',  
-> master\_log\_pos=31422,  
-> master\_connect\_retry=10;  
Query OK, 0 rows affected (0.04 sec)

mysql> slave start;  
Query OK, 0 rows affected (0.00 sec)

mysql> show slave status\G;  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 1. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
Slave\_IO\_State: Waiting for master to send event  
Master\_Host: 10.0.2.209  
Master\_User: slave  
Master\_Port: 3306  
Connect\_Retry: 10  
Master\_Log\_File: mysql-bin.000004  
Read\_Master\_Log\_Pos: 586  
Relay\_Log\_File: localhost-relay-bin.000002  
Relay\_Log\_Pos: 251  
Relay\_Master\_Log\_File: mysql-bin.000004  
Slave\_IO\_Running: Yes  
Slave\_SQL\_Running: Yes  
Replicate\_Do\_DB:  
Replicate\_Ignore\_DB:  
Replicate\_Do\_Table:  
Replicate\_Ignore\_Table:  
Replicate\_Wild\_Do\_Table:  
Replicate\_Wild\_Ignore\_Table:  
Last\_Errno: 0  
Last\_Error:  
Skip\_Counter: 0  
Exec\_Master\_Log\_Pos: 586  
Relay\_Log\_Space: 410  
Until\_Condition: None  
Until\_Log\_File:  
Until\_Log\_Pos: 0  
Master\_SSL\_Allowed: No  
Master\_SSL\_CA\_File:  
Master\_SSL\_CA\_Path:  
Master\_SSL\_Cert:  
Master\_SSL\_Cipher:  
Master\_SSL\_Key:  
Seconds\_Behind\_Master: 0  
Master\_SSL\_Verify\_Server\_Cert: No  
Last\_IO\_Errno: 0  
Last\_IO\_Error:  
Last\_SQL\_Errno: 0  
Last\_SQL\_Error:  
1 row in set (0.01 sec)

看见蓝色那两行已经yes了，表示配置成功，刚才一口气把DB2配成DB1和DB3的主，现在要把DB1配成DB2的主，实现DB1和DB2互为主从关系

**查看DB1的指针位置：**

mysql> show master status;  
+------------------+----------+--------------+------------------+  
| File | Position | Binlog\_Do\_DB | Binlog\_Ignore\_DB |  
+------------------+----------+--------------+------------------+  
| mysql-bin.000005 | 586 | | |  
+------------------+----------+--------------+------------------+  
1 row in set (0.00 sec)

**然后在DB2上导入指针位置：**

mysql> change master to  
-> master\_host='10.0.2.208',  
-> master\_user='slave',  
-> master\_password='slave',  
-> master\_log\_file='mysql-bin.000005',  
-> master\_log\_pos=586,  
-> master\_connect\_retry=10;  
Query OK, 0 rows affected (0.01 sec)

mysql> slave start;  
Query OK, 0 rows affected (0.00 sec)

mysql> show slave status\G;  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 1. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
Slave\_IO\_State: Waiting for master to send event  
Master\_Host: 10.0.2.208  
Master\_User: slave  
Master\_Port: 3306  
Connect\_Retry: 10  
Master\_Log\_File: mysql-bin.000005  
Read\_Master\_Log\_Pos: 586  
Relay\_Log\_File: localhost-relay-bin.000002  
Relay\_Log\_Pos: 251  
Relay\_Master\_Log\_File: mysql-bin.000005  
Slave\_IO\_Running: Yes  
Slave\_SQL\_Running: Yes  
Replicate\_Do\_DB:  
Replicate\_Ignore\_DB:  
Replicate\_Do\_Table:  
Replicate\_Ignore\_Table:  
Replicate\_Wild\_Do\_Table:  
Replicate\_Wild\_Ignore\_Table:  
Last\_Errno: 0  
Last\_Error:  
Skip\_Counter: 0  
Exec\_Master\_Log\_Pos: 586  
Relay\_Log\_Space: 410  
Until\_Condition: None  
Until\_Log\_File:  
Until\_Log\_Pos: 0  
Master\_SSL\_Allowed: No  
Master\_SSL\_CA\_File:  
Master\_SSL\_CA\_Path:  
Master\_SSL\_Cert:  
Master\_SSL\_Cipher:  
Master\_SSL\_Key:  
Seconds\_Behind\_Master: 0  
Master\_SSL\_Verify\_Server\_Cert: No  
Last\_IO\_Errno: 0  
Last\_IO\_Error:  
Last\_SQL\_Errno: 0  
Last\_SQL\_Error:  
1 row in set (0.00 sec)

## 2.3.3 测试

现在三台DB的主从关系都配置好了，可以测试一下：

在DB1的test库建个表，在DB2插入一条数据，在DB3查看

**DB1：**

mysql> create table test.t (id int(11) not null auto\_increment,name varchar(30),primary key (id));  
Query OK, 0 rows affected (0.03 sec)

**DB2：**

mysql> use test  
Database changed  
mysql> show tables;  
+----------------+  
| Tables\_in\_test |  
+----------------+  
| t |  
+----------------+  
1 row in set (0.00 sec)

mysql> insert into t(name) values ('mysql-mmm');  
Query OK, 1 row affected (0.02 sec)

**DB3：**

mysql> use test  
Database changed  
mysql> select \* from t;  
+----+-----------+  
| id | name |  
+----+-----------+  
| 2 | mysql-mmm |  
+----+-----------+  
1 row in set (0.00 sec)

**DB1：**

mysql> select \* from test.t;  
+----+-----------+  
| id | name |  
+----+-----------+  
| 2 | mysql-mmm |  
+----+-----------+  
1 row in set (0.00 sec)

OK！三台DB的主从配置正常工作

# 三、安装和配置Mysql-mmm

## 3.1 安装perl模块

安装Mysql-mmm之前，需要先安装一些perl模块，安装方法用perl -MCPAN -e shell 直接在线安装，第一次使用perl -MCPAN -e shell 命令时会有一大堆选项的，我直接enter过去，比较关键的是选择地区、国家和源这三个，按的时候注意选择一下就没问题，

**重新配置CPAN**

#perl -MCPAN -e shell

cpan> o conf init

cpan> o conf init urllist

特别注意URL的选择

**DB1、DB2和DB3需要安装以下模块：**

# perl -MCPAN -e shell

cpan> install Algorithm::Diff

cpan> install DBI

cpan>install Log::Dispatch

cpan> install Log::Log4perl

cpan> install Mail::Send

cpan> install Net::ARP

cpan> install Proc::Daemon

cpan> install Time::HiRes

cpan>install DBD::mysql

cpan>install File::stat

cpan>install File::basename

**由于DB3还负责心跳监听，所有还要安装以下模块：**

# perl -MCPAN -e shell

cpan> install Algorithm::Diff

cpan> install Class::Singleton

cpan> install Log::Dispatch

cpan> install Log::Log4perl

cpan> install Mail::Send

cpan> install Proc::Daemon

cpan> install Thread::Queue

cpan> install Time::HiRes

cpan> install DBI

cpan>install DBD::mysql

## 3.2 安装Mysql-mmm

wget http://mysql-mmm.org/\_media/:mmm2:mysql-mmm-2.2.1.tar.gz -O mysql-mmm-2.2.1.tar.gz

tar zxvf mysql-mmm-2.2.1.tar.gz   
cd mysql-mmm-2.2.1

make  
make instal

**数据库授权一个Mysql-mmm专用用户（DB1、DB2和DB3都要授权）：**

mysql -uroot -e "grant super,replication client,process on \*.\* to 'mmm'@'192.168.%.%' identified by 'p@9865321';"

mysql -uroot -e "grant super,replication client,process on \*.\* to 'mmm\_agent'@'192.168.16.9' identified by ' p@9865321';"

mysql -uroot -e "grant super,replication client,process on \*.\* to 'mmm\_agent'@'192.168.16.8' identified by ' p@9865321';"

## 3.3 修改Mysql-mmm配置文件

**DB1：**

vi /etc/mysql-mmm/mmm\_agent.conf

配置文件内容：

include mmm\_common.conf  
this db1

蓝色的部分DB2和DB3则分别修改为db2和db3

**DB1、DB2和DB3都修改成下面内容并启动mysql-mmm-agant**

vi /etc/mysql-mmm/mmm\_common.conf

配置文件内容：

active\_master\_role writer

<host default>

cluster\_interface eth0

pid\_path /var/run/mmm\_agentd.pid

bin\_path /usr/lib/mysql-mmm/

replication\_user slave

replication\_password slave

agent\_user mmm\_agent

agent\_password mmm\_agent

</host>

<host db1>

ip 10.0.2.208

mode master

peer db2

</host>

<host db2>

ip 10.0.2.209

mode master

peer db1

</host>

<host db3>

ip 10.0.2.211

mode slave

</host>

<role writer>

hosts db1, db2

ips 10.0.2.196

mode exclusive

</role>

<role reader>

hosts db1, db2, db3

ips 10.0.2.213,10.0.2.214

mode balanced

</role>

## 3.3.1 启动mysql-mmm-agant

DB1、DB2和DB3启动mysql-mmm-agant

/etc/init.d/mysql-mmm-agent start

## 3.4 配置监听

由于监听部署在DB3上，修改DB3上的/etc/mysql-mmm/mmm\_mon.conf

vi /etc/mysql-mmm/mmm\_mon.conf

配置文件内容：

include mmm\_common.conf

<monitor>

ip 127.0.0.1

pid\_path /var/run/mmm\_mond.pid

bin\_path /usr/lib/mysql-mmm/

status\_path /var/lib/misc/mmm\_mond.status

ping\_ips 10.0.2.208, 10.0.2.209, 10.0.2.211

auto\_set\_online 60

</monitor>

<host default>

monitor\_user mmm\_agent

monitor\_password mmm\_agent

</host>

debug 0

## 3.4.1 启动mysql-mmm-monitor

**启动mysql-mmm-monitor：**

[root@localhost mysql-mmm-2.2.1]# /etc/init.d/mysql-mmm-monitor start  
Daemon bin: '/usr/sbin/mmm\_mond'  
Daemon pid: '/var/run/mmm\_mond.pid'  
Starting MMM Monitor daemon: Ok

[root@localhost mysql-mmm-2.2.1]# mmm\_control show  
db1(10.0.2.208) master/AWAITING\_RECOVERY. Roles:  
db2(10.0.2.209) master/AWAITING\_RECOVERY. Roles:  
db3(10.0.2.211) slave/AWAITING\_RECOVERY. Roles:

**在监听服务器上将所有DB服务器设为online状态：**

[root@localhost mysql-mmm-2.2.1]# mmm\_control set\_online db1  
OK: State of 'db1' changed to ONLINE. Now you can wait some time and check its new roles!  
[root@localhost mysql-mmm-2.2.1]# mmm\_control set\_online db2  
OK: State of 'db2' changed to ONLINE. Now you can wait some time and check its new roles!  
[root@localhost mysql-mmm-2.2.1]# mmm\_control set\_online db3  
OK: State of 'db3' changed to ONLINE. Now you can wait some time and check its new roles!  
[root@localhost mysql-mmm-2.2.1]# mmm\_control show  
 db1(10.0.2.208) master/ONLINE. Roles: reader(10.0.2.213)

db2(10.0.2.209) master/ONLINE. Roles: writer(10.0.2.196)

db3(10.0.2.211) slave/ONLINE. Roles: reader(10.0.2.214)

**设置成功，下面查看节点状态：**

[root@localhost mysql-mmm]# mmm\_control checks all

db2 ping [last change: 2012/06/21 17:48:08] OK

db2 mysql [last change: 2012/06/21 17:48:08] OK

db2 rep\_threads [last change: 2012/06/21 17:48:08] OK

db2 rep\_backlog [last change: 2012/06/21 17:48:08] OK: Backlog is null

db3 ping [last change: 2012/06/21 17:48:08] OK

db3 mysql [last change: 2012/06/21 17:48:08] OK

db3 rep\_threads [last change: 2012/06/21 17:48:08] OK

db3 rep\_backlog [last change: 2012/06/21 17:48:08] OK: Backlog is null

db1 ping [last change: 2012/06/21 17:48:08] OK

db1 mysql [last change: 2012/06/21 17:48:08] OK

db1 rep\_threads [last change: 2012/06/21 17:48:08] OK

db1 rep\_backlog [last change: 2012/06/21 17:48:08] OK: Backlog is null

## 3.5 测试

**停掉DB1上的mysql**

service mysqld stop

**查看监听状态：**

[root@localhost mysql-mmm]# mmm\_control show

db1(10.0.2.208) master/HARD\_OFFLINE. Roles:

db2(10.0.2.209) master/ONLINE. Roles: reader(10.0.2.213), writer(10.0.2.196)

db3(10.0.2.211) slave/ONLINE. Roles: reader(10.0.2.214)

**启动DB1上的mysql**

service mysqld start

**查看监听状态：**

[root@localhost mysql-mmm]# mmm\_control show

db1(10.0.2.208) master/ONLINE. Roles: reader(10.0.2.213)

db2(10.0.2.209) master/ONLINE. Roles: writer(10.0.2.196)

db3(10.0.2.211) slave/ONLINE. Roles: reader(10.0.2.214)

**停掉DB2上的mysql**

service mysqld stop

**查看监听状态：**

[root@localhost mysql-mmm]# mmm\_control show

db1(10.0.2.208) master/ONLINE. Roles: reader(10.0.2.213), writer(10.0.2.196)

db2(10.0.2.209) master/HARD\_OFFLINE. Roles:

db3(10.0.2.211) slave/ONLINE. Roles: reader(10.0.2.214)

**启动DB2上的mysql**

service mysqld start

**查看监听状态：**

[root@localhost mysql-mmm]# mmm\_control show

db1(10.0.2.208) master/ONLINE. Roles: writer(10.0.2.196)

db2(10.0.2.209) master/ONLINE. Roles: reader(10.0.2.213)

db3(10.0.2.211) slave/ONLINE. Roles: reader(10.0.2.214)