MMM结合Semisync机制实现Mysql Master-Master高可用

**架构：**

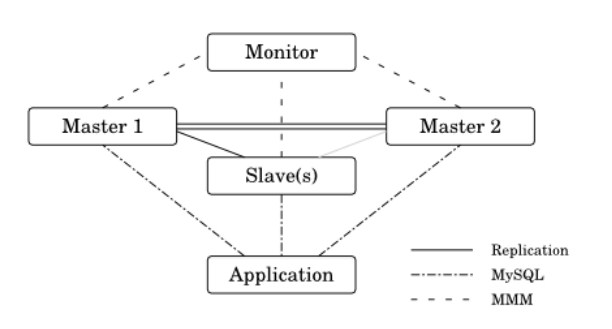
两个Master(主备模式)，一个或多个Slave(也可以没有Slave，只有主备Master)：

1、Monitor运行MMM Daemon程序，实现所有Mysql服务器的监控和故障切换工作；

2、Master1和Master2互为主备，同时只有一个主可用于写操作(也可同时分担读操作)，另一个作为备用，可以分担读操作，读写分离需要应用程序实现；

3、Slave机器与当前active Master同步，如当前active Master故障后，Master将切换到passive Master，同时MMM修改Slave与新的Master同步；

4、Application通过write和read ip进行读写操作；

[](http://static.oschina.net/uploads/img/201405/27170910_FPzu.jpg)

**环境：**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **主机名** | **服务器IP地址** | **Write IP** | **Read IP** | **备注** |
| mysql01 | 10.0.60.100 | 10.0.60.160 | 10.0.60.161 | 默认为active Master，运行mmm agent |
| mysql02 | 10.0.60.101 |  | 10.0.60.162 | 默认为passive  Master，运行mmm agent |
| mysql03 | 10.0.60.102 |  | 10.0.60.163 | Slave，由MMM维护，运行mmm agent |
| mysql04 | 10.0.60.103 |  |  | 监控机，运行MMM  Daemon程序 |

**软件信息：**

Mysql：5.6.17-log MySQL Community Server (GPL)

MMM：mysql-mmm-2.2.1

OS：CentOS release 6.4 (Final)，kernel 2.6.32-358.el6.x86\_64

**一、配置复制环境**

这里是全新配置，如果是已经存在了单master和slave环境，将配置不一样，可以结合xtrabackup工具实现数据的备份和恢复，配置主备master环境。

**前提要求：**

1、所有mysql实例开启read\_only=1；

2、主备master需要开启log\_bin；

3、所有mysql实例配置不同的server\_id以及不同的二进制日志、relay日志文件名；

**参考配置参数：**

mysql01的特殊配置参数：

mysql01>\! grep -E "log\_bin|server\_id|read\_only" my.cnf  
log\_bin = mysql01-bin  
server\_id = 1  
**read**\_**only**  
mysql01>

mysql02的特殊配置参数：

mysql02>\! grep -E "log\_bin|server\_id|read\_only" my.cnf  
log\_bin = mysql02-bin  
server\_id = 2  
**read**\_**only**  
mysql02>

mysql03的特殊配置参数：

mysql03>\! grep -E "log\_bin|server\_id|read\_only" my.cnf  
log\_bin = mysql\_bin  
server\_id = 3  
read\_only

**配置主从：**

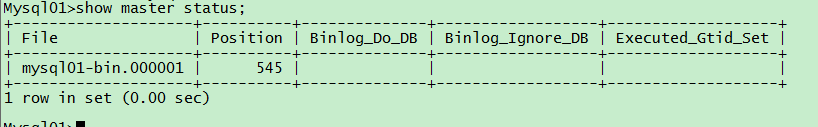
1、配置mysql01和mysql02互为主从：

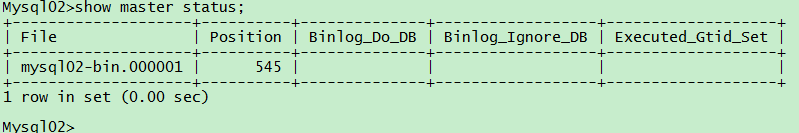
在mysql01和mysql02上创建同样的复制账号：

**grant** replication slave **on** \*.\* **to** 'repl'@'10.0.60.%' identified **by** 'repl';

查看master状态：

**show** master status;

[](http://static.oschina.net/uploads/img/201405/27170910_t7el.png)

[](http://static.oschina.net/uploads/img/201405/27170910_dTdC.png)

在每个节点执行CHANGE MASTER TO语句：

mysql01> change master **to** master\_host = '10.0.60.101',   
master\_user='repl',   
master\_password='repl',   
master\_log\_file='mysql02-bin.000001',   
master\_log\_pos=545;

mysql02> change master **to** master\_host = '10.0.60.100',   
master\_user='repl',   
master\_password='repl',   
master\_log\_file='mysql01-bin.000001',   
master\_log\_pos=545;

在两个节点开启slave：

**start** slave;

查看slave状态是否正常：

mysql02>**show** slave status\**G**;  
             Slave\_IO\_Running: Yes  
            Slave\_SQL\_Running: Yes

2、配置mysql03为mysql01的从服务器

mysql03> change master **to** master\_host = '10.0.60.100',   
master\_user='repl',   
master\_password='repl',   
master\_log\_file='mysql01-bin.000001',   
master\_log\_pos=545;

验证slave状态正常后，开始下面的步骤。

**二、配置半同步**

使用半同步机制，可以确保至少一台slave收到master的二进制日志，在一定程度上保证了数据的一致性，减少了当master当机时，造成数据丢失。

半同步机制由google贡献，从mysql 5.5开始原生支持该特性。

**前提要求：**

1、主备master都要安装并开启semisync master和slave，因mmm不能进行semisync配置和管理；

2、slave需要安装并开启semisync slave；

**配置步骤：**

1、mysql01和mysql02安装semisync master和slave插件：

mysql01>INSTALL PLUGIN rpl\_semi\_sync\_master SONAME 'semisync\_master.so';  
Query OK, 0 **rows** affected (0.01 sec)  
  
mysql01>INSTALL PLUGIN rpl\_semi\_sync\_slave SONAME 'semisync\_slave.so';  
Query OK, 0 **rows** affected (0.00 sec)

mysql02>INSTALL PLUGIN rpl\_semi\_sync\_master SONAME 'semisync\_master.so';  
Query OK, 0 **rows** affected (0.05 sec)  
  
mysql02>INSTALL PLUGIN rpl\_semi\_sync\_slave SONAME 'semisync\_slave.so';  
Query OK, 0 **rows** affected (0.01 sec)

2、mysql01和mysql02开启semisync master和slave：

mysql01>**SET** **GLOBAL** rpl\_semi\_sync\_master\_enabled = 1;  
Query OK, 0 **rows** affected (0.00 sec)  
  
mysql01>**SET** **GLOBAL** rpl\_semi\_sync\_slave\_enabled = 1;  
Query OK, 0 **rows** affected (0.00 sec)

mysql02>**SET** **GLOBAL** rpl\_semi\_sync\_master\_enabled = 1;  
Query OK, 0 **rows** affected (0.00 sec)  
  
mysql02>**SET** **GLOBAL** rpl\_semi\_sync\_slave\_enabled = 1;  
Query OK, 0 **rows** affected (0.00 sec)

同时将参数写入到配置文件，以mysql实例开启时自动开启半同步：

mysql02>\! cat my.cnf|grep semi  
rpl\_semi\_sync\_master\_enabled = 1  
rpl\_semi\_sync\_slave\_enabled = 1

3、mysql03安装并开启semisync slave插件：

mysql03>INSTALL PLUGIN rpl\_semi\_sync\_slave SONAME 'semisync\_slave.so';  
Query OK, 0 **rows** affected (0.01 sec)  
  
mysql03>**SET** **GLOBAL** rpl\_semi\_sync\_slave\_enabled = 1;  
Query OK, 0 **rows** affected (0.00 sec)

同时将参数写入到配置文件，以mysql实例开启时自动开启半同步：

mysql03>\! cat my.cnf|grep semi  
rpl\_semi\_sync\_slave\_enabled = 1  
mysql03>

4、所有mysql实例停止slave并开启slave，使半同步机制生效：

stop slave;**start** slave;

5、查看semisync状态

mysql01>**show** status **like** '%emi%';  
+--------------------------------------------+-------+  
| Variable\_name                              | **Value** |  
+--------------------------------------------+-------+  
| Rpl\_semi\_sync\_master\_clients               | 2     |  
| Rpl\_semi\_sync\_master\_net\_avg\_wait\_time     | 0     |  
| Rpl\_semi\_sync\_master\_net\_wait\_time         | 0     |  
| Rpl\_semi\_sync\_master\_net\_waits             | 0     |  
| Rpl\_semi\_sync\_master\_no\_times              | 0     |  
| Rpl\_semi\_sync\_master\_no\_tx                 | 0     |  
| Rpl\_semi\_sync\_master\_status                | **ON**    |  
| Rpl\_semi\_sync\_master\_timefunc\_failures     | 0     |  
| Rpl\_semi\_sync\_master\_tx\_avg\_wait\_time      | 0     |  
| Rpl\_semi\_sync\_master\_tx\_wait\_time          | 0     |  
| Rpl\_semi\_sync\_master\_tx\_waits              | 0     |  
| Rpl\_semi\_sync\_master\_wait\_pos\_backtraverse | 0     |  
| Rpl\_semi\_sync\_master\_wait\_sessions         | 0     |  
| Rpl\_semi\_sync\_master\_yes\_tx                | 0     |  
| Rpl\_semi\_sync\_slave\_status                 | **ON**    |  
+--------------------------------------------+-------+  
15 **rows** **in** **set** (0.00 sec)

**三、配置MMM**

Multi Master Replication Manager for Mysql(MMM)是一套开源的perl脚本，对Mysql Master-Master复制环境(在任何时刻只有一个节点可写)进行监控、故障恢复以及管理。同时能根据复制的延时情况管理读负载均衡，通过迁移read虚拟IP地址。同时也能用于数据备份，以及节点之间重同步。

主要由三个脚本组成：

1、mmm\_mod：监控daemon程序，进行监控工作，并决定读、写角色的迁移；最好运行在专用的监控服务器上，可以管理多套Master-Slave集群。

2、mmm\_agentd：客户端daemon程序，运行在所有mysql实例服务器，与监控节点进行简单的远程通信。

3、mmm\_control：用于管理mmm\_mond进程的命令行脚本。

**前提需求：**

1、支持环境：

两个节点的Master-Master环境，MMM需要5个IP地址(每个节点一个固定IP地址，一个write IP地址，两个read IP地址，write和read IP依据节点的可用性进行自动的迁移)，正常情况下，active master有一个write IP和一个read IP地址，standby master有一个read IP地址，如果当前active master故障，write和read IP地址将迁移到standby master；

两个节点的Master-Master，以及一个或多个slave的环境，同时也是大多数企业使用的方案(可以更好的扩展读，同时有冗余的Slave可用于备份等工作，防止阻塞正常的事务)。

2、n+1个主机：n个运行mysql实例的服务器，一个机器用于运行MMM监控daemon程序；

3、2\*(n+1) IP地址：每个主机一个固定IP地址，同时每台mysql实例一个read IP地址以及一个write IP地址；

4、monitor数据库用户：需要REPLICATION CLIENT权限，用于MMM监控(mmm\_mond)；

5、agent数据库用户：需要SUPER、REPLICATION CLIENT、PROCESS权限，用于MMM 客户端(mmm\_agentd)，可以只针对本机IP进行授权；

6、relication数据库用户：需要REPLICATION SLAVE权限，用于mysql复制；

7、tools数据库用户：需要SUPER、REPLICATION CLIENT、RELOAD权限，用于MMM tools(如mmm\_backup、mmm\_clone、mmm\_restore)

1、在mysql实例服务器安装依赖包和mmm

安装依赖包：

yum -y install perl iproute perl-Algorithm-Diff perl-DBI perl-Class-Singleton perl-DBD-MySQL perl-Log-Log4perl perl-Log-Dispatch perl-Proc-Daemon perl-MailTools perl-Time-HiRes perl-Mail-Sendmail perl-Mail-Sender perl-Email-Date-Format perl-MIME-Lite perl-Net-ARP

如果在标准软件仓库和EPEL软件仓库没有，需要单独下载，可以去以下网址下载：

[http://rpm.pbone.net](http://rpm.pbone.net/)

<http://search.cpan.org/>

<http://www.rpmfind.net/>

安装mmm：

tar xvf mysql-mmm-2.2.1.tar.gz  
cd mysql-mmm-2.2.1  
make install

2、在mysql实例服务器配置mmm agent

mmm\_agentd使用mmm\_agent.conf配置文件：

# cat /etc/mysql-mmm/mmm\_agent.conf     
include mmm\_common.conf #包含这个公用配置文件  
#Description: name of this host，可以不是主机名，每台mysql实例的host不同(如mysql01设置为db1，mysql02设置为db2，mysql03设置为db3)  
this db1  
#Description: Enable debug mode，设置1，打印日志到前台，按ctrl+c将结束进程  
debug 0  
#Description: Maximum number of retries when killing threads to prevent further  
#writes during the removal of the active\_master\_role.  
max\_kill\_retries 10

公用配置文件：mmm\_common.conf，每个实例一样，并要拷贝到监控服务器供mmm\_mond使用，进行网卡接口的定义，每个主机的描述，复制和mmm agent的用户名和密码配置，以及读写规则等

# cat /etc/mysql-mmm/mmm\_common.conf   
#Description: name of the role for which identifies the active master，定义活动master为可写  
active\_master\_role                              writer  
  
<host default> #默认段  
        #Description: network interface on which the IPs of the roles should be configured，用于绑定ip的网络接口  
        cluster\_interface                       eth0  
  
        pid\_path                                /var/run/mmm\_agentd.pid  
        bin\_path                                /usr/lib/mysql-mmm/  
        #Description: Port on which mmm agentd listens  
        agent\_port                              9989  
        #Description: Port on which mysqld is listening  
        mysql\_port                              3306  
  
        #Description: mysql user used for replication  
        replication\_user                        repl  
        #Description: mysql password used for replication  
        replication\_password                    repl  
        #Description: mysql user for MMM Agent  
        agent\_user                              mmm\_agent  
        #Description: mysql password for MMM Agent  
        agent\_password                          mmm\_agent  
</host>  
  
<host db1> #命名段，指定每个mysql实例主机  
        #Description: IP of host  
        ip                                      10.0.60.100  
        #Description: Mode of host. Either master or slave.  
        mode                                    master  
        #Description: Name of peer host (if mode is master)  
        peer                                    db2  
</host>  
  
<host db2>  
        ip                                      10.0.60.101  
        mode                                    master  
        peer                                    db1  
</host>  
  
<host db3>  
        ip                                      10.0.60.102  
        mode                                    slave  
</host>  
  
  
<role writer> #定义write角色  
        #Description: Hosts which may take over the role  
        hosts                                   db1, db2  
        #Description: One or multiple IPs associated with the role，指定浮动write IP地址  
        ips                                     10.0.60.160  
        #Description: Mode of role. Either balanced or exclusive  
        mode                                    exclusive  
        #Description: The preferred host for this role. Only allowed for exclusive roles.  
        #prefer                                 -  
</role>  
  
<role reader>  #定义read角色  
        hosts                                   db1, db2, db3  
        ips                                     10.0.60.161,10.0.60.162,10.0.60.163 #浮动read IP地址  
        mode                                    balanced  
</role>

3、启动mmm agent服务

/etc/init.d/mysql-mmm-agent start  
chkconfig --level 2345 mysql-mmm-agent on

4、在监控服务器(mysql04)安装依赖包和mmm

安装依赖包：

yum -y install perl iproute perl-Algorithm-Diff perl-DBI perl-Class-Singleton perl-DBD-MySQL perl-Log-Log4perl perl-Log-Dispatch perl-Proc-Daemon perl-MailTools perl-Time-HiRes perl-Mail-Sendmail perl-Mail-Sender perl-Email-Date-Format perl-MIME-Lite perl-Net-Ping

安装mmm：

tar xvf mysql-mmm-2.2.1.tar.gz  
cd mysql-mmm-2.2.1  
make install

5、配置mmm 监控配置文件

mmm\_mond和mmm\_control使用mmm\_mon.conf或mmm\_mon\_CLUSTER.conf配置文件

mmm\_mon.conf配置文件参考：

# cat /etc/mysql-mmm/mmm\_mon.conf   
include mmm\_common.conf  
  
#The monitor section is required by mmm\_mond and mmm\_control  
<monitor>  
        #Description: IP on which mmm\_mond listens  
        ip                                              127.0.0.1  
        #Description: Port on which mmm mond listens  
        port                                    9988  
        #Description: Location of pid-file  
        pid\_path                                /var/run/mmm\_mond.pid  
        #Description: Path to directory containing MMM binaries  
        bin\_path                                /usr/lib/mysql-mmm/  
        #Description: Location of of status file  
        status\_path                             /var/lib/misc/mmm\_mond.status  
        #Description: Break between network checks  
        ping\_interval                           1  
        #Description: IPs used for network checks，指定所有mysql服务器IP，write和read IP地址，用于进行ping检查  
        ping\_ips                                10.0.60.100, 10.0.60.101, 10.0.60.102, 10.0.60.160, 10.0.60.161, 10.0.60.162, 10.0.60.163  
        #Description: Duration in seconds for flap detection. See flap\_count  
        flap\_duration                           3600  
        #Description: Maximum number of downtimes within flap\_duration seconds after  
        #which a host is considered to be flapping.  
        flap\_count                              3  
        #Description: How many seconds to wait before switching node status from  
        #AWAITING\_RECOVERY to ONLINE. 0 = disabled.  
        auto\_set\_online                         0  
        #Description: Binary used to kill hosts if roles couldn’t be removed because the agent  
        #was not reachable. You have to provide a custom binary for this which  
        #takes the hostname as first argument and the state of check ping (1 -ok; 0 - not ok) as second argument.  
        kill\_host\_bin                           /usr/lib/mysql-mmm/monitor/kill\_host  
        #Description: Startup carefully i.e. switch into passive mode when writer role is  
        #configured on multiple hosts  
        careful\_startup                         0  
        #Description: Default mode of monitor.  
        mode                                    active  
        #Description: How many seconds to wait for other master to become ONLINE before  
        #switching from mode WAIT to mode ACTIVE. 0 = infinite.  
        wait\_for\_other\_master                   120  
</monitor>  
  
<host default>  
        #Description: mysql user for MMM Monitor  
        monitor\_user                    mmm\_agent  
        #Description: mysql password for MMM Monitor  
        monitor\_password                mmm\_agent  
</host>  
  
<check  mysql> #check段，mmm执行ping、mysql、rep\_threads、rep\_backlog四种检查，可以分别进行检查间隔等参数配置。  
        #Description: Perform check every 5 seconds  
        check\_period                    5  
        #Description: Check is considered as failed if it doesn’t succeed for at least  
        #trap period seconds.  
        trap\_period                     10  
        #Description: Check times out after timeout seconds  
        timeout                         2  
        #Description: Restart checker process after restart after checks  
        restart\_after                   10000  
        #Description: Maximum backlog for check rep\_backlog.  
        max\_backlog                     60  
</check>  
  
#设置为1,开启调试模式，打印日志到前台，ctrl+c将结束进程，对于调试有帮助  
debug 0

6、开启mmm monitor监控

/etc/init.d/mysql-mmm-monitor start  
chkconfig --level 2345 mysql-mmm-monitor on

7、使用mmm\_control查看状态

# mmm\_control show  
  db1(10.0.60.100) master/ONLINE. Roles: reader(10.0.60.163), writer(10.0.60.160)  
  db2(10.0.60.101) master/ONLINE. Roles: reader(10.0.60.161)  
  db3(10.0.60.102) slave/ONLINE. Roles: reader(10.0.60.162)

注：

当节点第一次开启，状态为等待恢复。

设置节点online：

# mmm\_control set\_online db1

**MMM如何工作：**

当故障发生时，mmm迅速的迁移故障节点的IP地址从一个节点到另一个节点，并使用Net::ARP Perl模块更新ARP表。

**处理过程：**

**在故障active master节点：**

1、mysql 设置为read\_only(set global read\_only=1)，防止写事务；

2、中断活动连接；

3、移除写ip；

**在新master节点：**

1、运行在passive master的mmm进程被通知即将成为active write；

2、slave将尝试从master的二进制日志抓取任何剩余事务；

3、关闭read\_only(set global read\_only=0)；

4、绑定write ip，并发生arp通告；

**四、测试**

**1、测试mysql01 mysql实例故障**

手动关闭mysql01服务器上的mysql实例，期望master将迁移到mysql02

停止mysql01的mysqld进程：也可以使用"killall -15 mysqld"结束mysqld进程

mysql01>\! sh stop.sh

查看mmm\_mond的日志：总共经过10s时间完成迁移

# tail -f /var/log/mysql-mmm/mmm\_mond.log   
2014/05/27 14:19:13  WARN Check 'rep\_backlog' on 'db1' is **in** unknown state! Message: UNKNOWN: Connect error (host = 10.0.60.100:3306, user = mmm\_agent)! Lost connection to MySQL server at 'reading initial communication packet', system error: 111  
2014/05/27 14:19:13  WARN Check 'rep\_threads' on 'db1' is **in** unknown state! Message: UNKNOWN: Connect error (host = 10.0.60.100:3306, user = mmm\_agent)! Lost connection to MySQL server at 'reading initial communication packet', system error: 111  
2014/05/27 14:19:22 ERROR Check 'mysql' on 'db1' has failed **for** 10 seconds! Message: ERROR: Connect error (host = 10.0.60.100:3306, user = mmm\_agent)! Lost connection to MySQL server at 'reading initial communication packet', system error: 111  
2014/05/27 14:19:23 FATAL State of host 'db1' changed from ONLINE to HARD\_OFFLINE (ping: OK, mysql: not OK)  
2014/05/27 14:19:23  INFO Removing all roles from host 'db1':  
2014/05/27 14:19:23  INFO     Removed role 'reader(10.0.60.163)' from host 'db1'  
2014/05/27 14:19:23  INFO     Removed role 'writer(10.0.60.160)' from host 'db1'  
2014/05/27 14:19:23  INFO Orphaned role 'writer(10.0.60.160)' has been assigned to 'db2' #可以看到写IP已经迁移到mysql02  
2014/05/27 14:19:23  INFO Orphaned role 'reader(10.0.60.163)' has been assigned to 'db3'

使用mmm\_control命令查看状态：

[root@mysql04 ~]# mmm\_control show  
  db1(10.0.60.100) master/HARD\_OFFLINE. Roles: #mysql01状态已经变为HARD\_OFFLINE，意外着ping错误或mysql故障  
  db2(10.0.60.101) master/ONLINE. Roles: reader(10.0.60.161), writer(10.0.60.160)  
  db3(10.0.60.102) slave/ONLINE. Roles: reader(10.0.60.162), reader(10.0.60.163)

检查mysql02的read\_only变量是否改变：

mysql02>**show** **global** variables **like** 'read\_only'; #默认在passive master时，read\_only为ON  
+---------------+-------+  
| Variable\_name | **Value** |  
+---------------+-------+  
| read\_only     | **OFF**   |  
+---------------+-------+  
1 **row** **in** **set** (0.00 sec)  
  
mysql02>

检查mysql03是否已经将mysql02作为主：

mysql03>**show** slave status\**G**;  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 1. **row** \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
               Slave\_IO\_State: Waiting **for** master **to** send event  
                  Master\_Host: 10.0.60.101 #已经从Mysql02同步  
                  Master\_User: repl  
                  Master\_Port: 3306  
                Connect\_Retry: 10  
              Master\_Log\_File: mysql02-bin.000014  
          Read\_Master\_Log\_Pos: 120  
               Relay\_Log\_File: mysql03-relay-bin.000002  
                Relay\_Log\_Pos: 285  
        Relay\_Master\_Log\_File: mysql02-bin.000014  
             Slave\_IO\_Running: Yes  
            Slave\_SQL\_Running: Yes

当再次启动mysql01的mysql实例，db1的状态将由HARD\_OFFLINE改变为AWAITING\_RECOVERY：

[root@mysql04 ~]# mmm\_control show  
  db1(10.0.60.100) master/AWAITING\_RECOVERY. Roles:   
  db2(10.0.60.101) master/ONLINE. Roles: reader(10.0.60.161), writer(10.0.60.160)  
  db3(10.0.60.102) slave/ONLINE. Roles: reader(10.0.60.162), reader(10.0.60.163)

需要手动设置为online，mmm才会分配read ip给mysql01，并与mysql02同步：

[root@mysql04 ~]# mmm\_control set\_online db1  
OK: State of 'db1' changed to ONLINE. Now you can wait some time and check its new roles!

mysql01>**show** slave status\**G**;  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 1. **row** \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
               Slave\_IO\_State: Waiting **for** master **to** send event  
                  Master\_Host: 10.0.60.101 #mysql01已经从Mysql02同步  
                  Master\_User: repl  
                  Master\_Port: 3306  
                Connect\_Retry: 10  
              Master\_Log\_File: mysql02-bin.000015  
          Read\_Master\_Log\_Pos: 120  
               Relay\_Log\_File: mysql01-relay.000027  
                Relay\_Log\_Pos: 285  
        Relay\_Master\_Log\_File: mysql02-bin.000015  
             Slave\_IO\_Running: Yes  
            Slave\_SQL\_Running: Yes

查看mysql02的semisync状态：

mysql02>**show** status **like** 'Rpl\_semi%';  
+--------------------------------------------+-------+  
| Variable\_name                              | **Value** |  
+--------------------------------------------+-------+  
| Rpl\_semi\_sync\_master\_clients               | 2     |  
| Rpl\_semi\_sync\_master\_net\_avg\_wait\_time     | 1053  |  
| Rpl\_semi\_sync\_master\_net\_wait\_time         | 2106  |  
| Rpl\_semi\_sync\_master\_net\_waits             | 2     |  
| Rpl\_semi\_sync\_master\_no\_times              | 0     |  
| Rpl\_semi\_sync\_master\_no\_tx                 | 0     |  
| Rpl\_semi\_sync\_master\_status                | **ON**    |  
| Rpl\_semi\_sync\_master\_timefunc\_failures     | 0     |  
| Rpl\_semi\_sync\_master\_tx\_avg\_wait\_time      | 1015  |  
| Rpl\_semi\_sync\_master\_tx\_wait\_time          | 1015  |  
| Rpl\_semi\_sync\_master\_tx\_waits              | 1     |  
| Rpl\_semi\_sync\_master\_wait\_pos\_backtraverse | 0     |  
| Rpl\_semi\_sync\_master\_wait\_sessions         | 0     |  
| Rpl\_semi\_sync\_master\_yes\_tx                | 1     |  
| Rpl\_semi\_sync\_slave\_status                 | **ON**    |  
+--------------------------------------------+-------+  
15 **rows** **in** **set** (0.00 sec)

**2、模拟mysql02(Active Master服务器) kernel panic，期望进行迁移**

执行上面的测试后，当前active master为mysql02，使用下面命令模拟kernel panic：

mysql02>\! **echo** "c" > /proc/sysrq-trigger

查看mmm\_mond日志：总共经过了20s的时间完成迁移

# tail -f /var/log/mysql-mmm/mmm\_mond.log  
2014/05/27 14:44:42  WARN Check 'rep\_threads' on 'db2' is **in** unknown state! Message: UNKNOWN: Connect error (host = 10.0.60.101:3306, user = mmm\_agent)! Can't connect to MySQL server on '10.0.60.101' (4)  
2014/05/27 14:44:42  WARN Check 'rep\_backlog' on 'db2' is in unknown state! Message: UNKNOWN: Connect error (host = 10.0.60.101:3306, user = mmm\_agent)! Can't connect to MySQL server on '10.0.60.101' (4)  
2014/05/27 14:44:46 FATAL Can't reach agent on host 'db2'  
2014/05/27 14:44:49 ERROR Check 'ping' on 'db2' has failed for 11 seconds! Message: ERROR: Could not ping 10.0.60.101 #ping检查错误  
2014/05/27 14:44:55 ERROR Check 'mysql' on 'db2' has failed for 14 seconds! Message: ERROR: Connect error (host = 10.0.60.101:3306, user = mmm\_agent)! Can't connect to MySQL server on '10.0.60.101' (4) #mysql检查错误，不能连接  
2014/05/27 14:44:59  INFO Check 'ping' on 'db2' is ok!  
2014/05/27 14:45:02 FATAL State of host 'db2' changed from ONLINE to HARD\_OFFLINE (ping: OK, mysql: not OK) #改变mysql02的状态为HARD\_OFFLINE  
2014/05/27 14:45:02  INFO Removing all roles from host 'db2':  #移除mysql02的角色  
2014/05/27 14:45:02  INFO     Removed role 'reader(10.0.60.161)' from host 'db2'  
2014/05/27 14:45:02  INFO     Removed role 'writer(10.0.60.160)' from host 'db2'  
2014/05/27 14:45:02 FATAL Agent on host 'db2' is reachable again  
2014/05/27 14:45:02  INFO Orphaned role 'writer(10.0.60.160)' has been assigned to 'db1' #分配角色到其他机器,write IP分配到mysql01，永远不会分配到mysql03  
2014/05/27 14:45:02  INFO Orphaned role 'reader(10.0.60.161)' has been assigned to 'db3'

使用mmm\_control查看状态：

[root@mysql04 ~]# mmm\_control show  
  db1(10.0.60.100) master/ONLINE. Roles: reader(10.0.60.163)  
  db2(10.0.60.101) master/HARD\_OFFLINE. Roles:   
  db3(10.0.60.102) slave/ONLINE. Roles: reader(10.0.60.162)

**3、模拟active master服务器网络不通，期望进行迁移，但是网络恢复后，将不会重启slave；**

当前active master为mysql01，在mysql01上禁用网卡：

mysql01>\! **cat** down\_net.sh  
ifdown eth0  
sleep 600  
ifup eth0  
mysql01>\! sh down\_net.sh

查看mmm\_mond日志：总共经过了9s完成迁移

2014/05/27 15:02:35  WARN Check 'rep\_threads' on 'db1' is **in** unknown state! Message: UNKNOWN: Connect error (host = 10.0.60.100:3306, user = mmm\_agent)! Can't connect to MySQL server on '10.0.60.100' (4)  
2014/05/27 15:02:35  WARN Check 'rep\_backlog' on 'db1' is in unknown state! Message: UNKNOWN: Connect error (host = 10.0.60.100:3306, user = mmm\_agent)! Can't connect to MySQL server on '10.0.60.100' (4)  
2014/05/27 15:02:41 FATAL Can't reach agent on host 'db1'  
2014/05/27 15:02:41 ERROR Check 'ping' on 'db1' has failed for 11 seconds! Message: ERROR: Could not ping 10.0.60.100  #ping检查错误  
2014/05/27 15:02:44 FATAL State of host 'db1' changed from ONLINE to HARD\_OFFLINE (ping: not OK, mysql: OK) #改变状态  
2014/05/27 15:02:44  INFO Removing all roles from host 'db1': #移除角色  
2014/05/27 15:02:44  INFO     Removed role 'reader(10.0.60.163)' from host 'db1'    
2014/05/27 15:02:44  INFO     Removed role 'writer(10.0.60.160)' from host 'db1'  
2014/05/27 15:02:44 ERROR Can't send offline status notification to 'db1' - killing it!  
2014/05/27 15:02:44 FATAL Could not kill host 'db1' - there may be some duplicate ips now! (There's no binary configured for killing hosts.)  
2014/05/27 15:02:44  INFO Orphaned role 'writer(10.0.60.160)' has been assigned to 'db2'  
2014/05/27 15:02:44  INFO Orphaned role 'reader(10.0.60.163)' has been assigned to 'db3'  
2014/05/27 15:02:48 ERROR Check 'mysql' on 'db1' has failed for 14 seconds! Message: ERROR: Connect error (host = 10.0.60.100:3306, user = mmm\_agent)! Can't connect to MySQL server on '10.0.60.100' (4)

使用mmm\_control查看状态：

[root@mysql04 ~]# mmm\_control show  
# Warning: agent on host db1 is not reachable  
  db1(10.0.60.100) master/HARD\_OFFLINE. Roles:   
  db2(10.0.60.101) master/ONLINE. Roles: reader(10.0.60.161), writer(10.0.60.160)  
  db3(10.0.60.102) slave/ONLINE. Roles: reader(10.0.60.162), reader(10.0.60.163)

**当网络恢复后，mmm会修改mysql01的slave配置，修改主为mysql02，但是没有重启slave，造成不能进行数据同步，需要手工重新开启slave。**

使用mmm\_control检查状态：

[root@mysql04 ~]# mmm\_control show  
  db1(10.0.60.100) master/AWAITING\_RECOVERY. Roles:   
  db2(10.0.60.101) master/ONLINE. Roles: reader(10.0.60.161), writer(10.0.60.160)  
  db3(10.0.60.102) slave/ONLINE. Roles: reader(10.0.60.162), reader(10.0.60.163)  
  
[root@mysql04 ~]# mmm\_control set\_online db1  #网络恢复后，手动设置为online  
OK: State of 'db1' changed to ONLINE. Now you can wait some time and check its new roles!  
[root@mysql04 ~]# mmm\_control show  
  db1(10.0.60.100) master/ONLINE. Roles: reader(10.0.60.163)  
  db2(10.0.60.101) master/ONLINE. Roles: reader(10.0.60.161), writer(10.0.60.160)  
  db3(10.0.60.102) slave/ONLINE. Roles: reader(10.0.60.162)

检查mysql01的slave状态：看上去正常的

mysql01>show slave status\G;  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 1. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
               Slave\_IO\_State: Waiting **for** master to send event  
                  Master\_Host: 10.0.60.101  
                  Master\_User: repl  
                  Master\_Port: 3306  
                Connect\_Retry: 10  
              Master\_Log\_File: mysql02-bin.000015  
          Read\_Master\_Log\_Pos: 328  
               Relay\_Log\_File: mysql01-relay.000027  
                Relay\_Log\_Pos: 493  
        Relay\_Master\_Log\_File: mysql02-bin.000015  
             Slave\_IO\_Running: Yes  
            Slave\_SQL\_Running: Yes

但是在mysql02(当前active master)插入数据，mysql01不能从mysql02同步：

mysql02> **insert** **into** t1 **values**(2);  
Query OK, 1 **row** affected (0.02 sec)  
mysql02>**select** \* **from** t1;  
+----+  
| id |  
+----+  
|  1 |  
|  2 |  
+----+  
2 **rows** **in** **set** (0.00 sec)

mysql03已经同步了数据：

mysql03>**select** \* **from** t1;  
+----+  
| id |  
+----+  
|  1 |  
|  2 |  
+----+  
2 **rows** **in** **set** (0.00 sec)

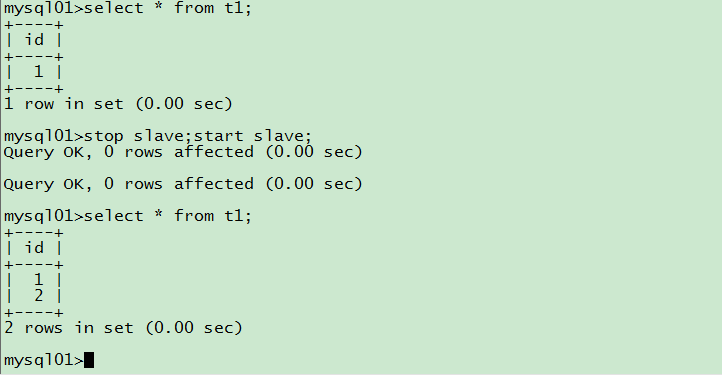
而mysql01没有同步数据：

mysql01>**select** \* **from** t1;  
+----+  
| id |  
+----+  
|  1 |  
+----+  
1 **row** **in** **set** (0.00 sec)

**解决方法：先停止slave，然后启动slave；**

mysql01>stop slave;**start** slave;   
Query OK, 0 **rows** affected (0.00 sec)  
  
Query OK, 0 **rows** affected (0.00 sec)  
  
mysql01>**select** \* **from** t1;  
+----+  
| id |  
+----+  
|  1 |  
|  2 |  
+----+  
2 **rows** **in** **set** (0.00 sec)

**截图：**

[](http://static.oschina.net/uploads/img/201405/27170911_1W7e.png)

**4、模拟slave线程故障，不管是io或sql线程故障，期望进行迁移，恢复时如果在flap\_duration时间内超过了flap\_count次数的故障，将不会自动恢复，状态由REPLICATION\_FAIL改为 AWAITING\_RECOVERY (because it's flapping)**

当前active master为mysql02。

停止active master(mysql02)的slave，不会造成迁移：

mmm\_mond的日志：已经检测到db2(mysql02)复制线程错误

2014/05/27 15:39:02 ERROR Check 'rep\_threads' on 'db2' has failed for 10 seconds! Message: ERROR: Replication is broken

使用mmm\_control查看状态：

[root@mysql04 ~]# mmm\_control show  
  db1(10.0.60.100) master/ONLINE. Roles: reader(10.0.60.161)  
  db2(10.0.60.101) master/ONLINE. Roles: reader(10.0.60.162), writer(10.0.60.160)  
  db3(10.0.60.102) slave/ONLINE. Roles: reader(10.0.60.163)

如果当其他slave(mysql01、mysql03)的线程(不管是io还是sql线程)故障将会发生迁移：

手工停止io线程：

mysql01>stop slave io\_thread;  
Query OK, 0 **rows** affected (0.01 sec)

查看mmm\_mond日志：

2014/05/27 15:43:28 ERROR Check 'rep\_threads' on 'db1' has failed **for** 10 seconds! Message: ERROR: Replication is broken  
2014/05/27 15:43:31 FATAL State of host 'db1' changed from ONLINE to REPLICATION\_FAIL  
2014/05/27 15:43:31  INFO Removing all roles from host 'db1':  
2014/05/27 15:43:31  INFO     Removed role 'reader(10.0.60.161)' from host 'db1'  #移除角色  
2014/05/27 15:43:31  INFO Orphaned role 'reader(10.0.60.161)' has been assigned to 'db3'

使用mmm\_control查看状态：

[root@mysql04 ~]# mmm\_control show  
  db1(10.0.60.100) master/REPLICATION\_FAIL. Roles:   
  db2(10.0.60.101) master/ONLINE. Roles: reader(10.0.60.162), writer(10.0.60.160)  
  db3(10.0.60.102) slave/ONLINE. Roles: reader(10.0.60.161), reader(10.0.60.163)

当重新开启io线程后，mmm将自动恢复db1，并重新迁移read IP到db1(mysql01)上，如果故障超过：

重新开启线程：

mysql01>**start** slave io\_thread;    
Query OK, 0 **rows** affected (0.00 sec)

查看mmm\_mond日志：

2014/05/27 15:45:23  INFO Check 'rep\_threads' on 'db1' is ok!  
2014/05/27 15:45:25 FATAL State of host 'db1' changed from REPLICATION\_FAIL to ONLINE  
2014/05/27 15:45:25  INFO Moving role 'reader(10.0.60.163)' from host 'db3' to host 'db1'

使用mmm\_control查看状态：

[root@mysql04 ~]# mmm\_control show  
  db1(10.0.60.100) master/ONLINE. Roles: reader(10.0.60.163)  
  db2(10.0.60.101) master/ONLINE. Roles: reader(10.0.60.162), writer(10.0.60.160)  
  db3(10.0.60.102) slave/ONLINE. Roles: reader(10.0.60.161)

**5、复制延时**

延时检查有max\_backlog控制，默认为60；

复制延时或错误，如果故障时间少于60s，状态为ONLINE，单会迁移，故障恢复后，mmm自动恢复read IP。如果rep\_backlog和rel\_threads同时错误，状态将为REPLICATION\_FAIL。

**6、mmm agent或monitor故障**

不会迁移角色，如果此时有master或slave故障，也将不能迁移角色

**参考：**

MMM官网：<http://mysql-mmm.org/>

MMM博客：<http://blog.mysql-mmm.org/>