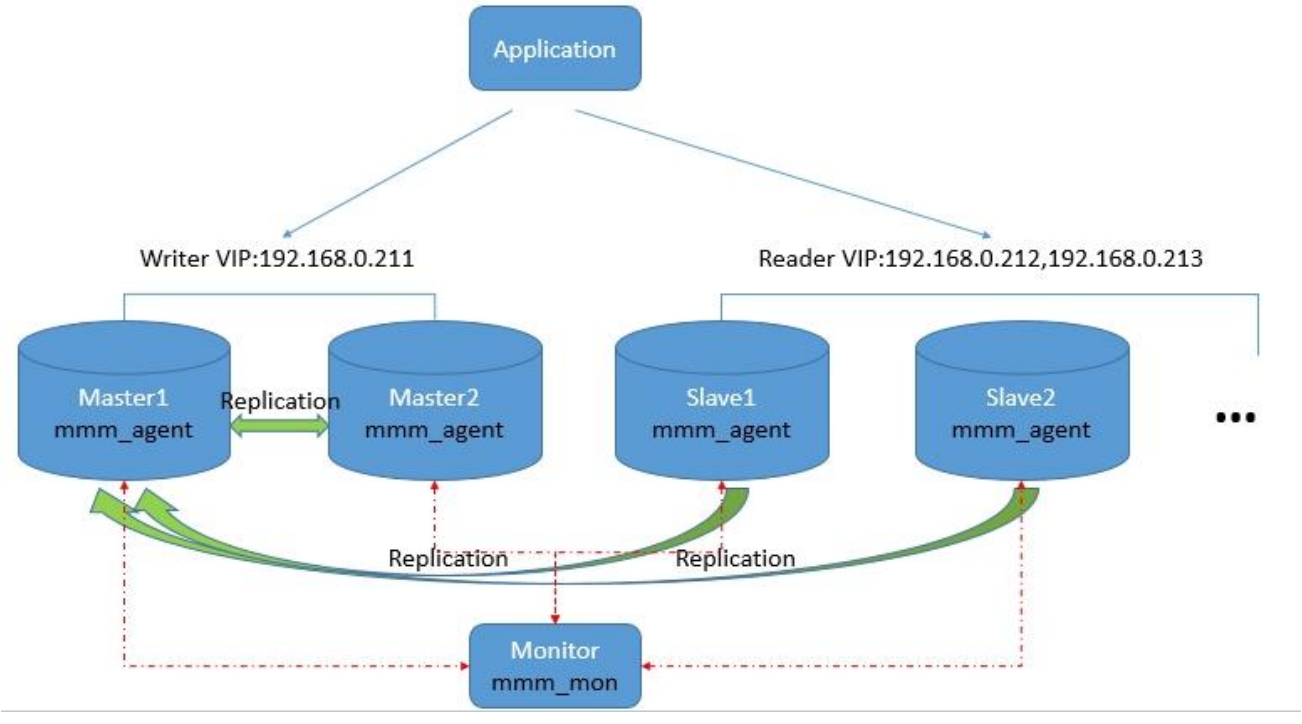
MySQL高可用集群之MySQL-MMM

一、环境简述

1、工作逻辑图



2、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。

3、MySQL-MMM 工作原理

MMM(Master-Master replication managerfor 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。

4、需求描述

操作系统：CentOS 6.5_X64

数据库：MySQL 5.1

MMM：MySQL-MMM 2.2.1

数据库分配：

function	ip	hostname	server id
monitoring host	192.168.0.201	monitor	无
master 1	192.168.0.202	db1	1
master 2	192.168.0.203	db2	2
slave 1	192.168.0.204	db3	3
slave 2	192.168.0.205	db4	4

虚拟 IP 地址（VIP）：

ip	role
192.168.0.211	writer
192.168.0.212	reader
192.168.0.213	reader

数据库同步需要的用户：

function	description	privileges
monitor user	mmm 监控用于对 mysql 服务器进程健康检查	REPLICATION CLIENT
agent user	mmm 代理用来更改只读模式，复制的主服务器等	SUPER, REPLICATION CLIENT, PROCESS
replication user	用于复制	REPLICATION SLAVE

二、db1，db2，db3 和 db4 安装数据库并配置

```
[root@db1 ~]# yum install mysql-server mysql
[root@db1 ~]# service mysqld start
[root@db1 ~]# mysqladmin -u root password 123.com

[root@db1 ~]# vi /etc/my.cnf      #添加如下
[mysqld]
binlog-do-db=test      #需要记录二进制日志的数据库，多个用逗号隔开
binlog-ignore-db=mysql，information_schema    #不需要记录二进制日志的数据库，多个用逗号隔开
auto_increment_increment=2    #字段一次递增多少
auto_increment_offset=1          #自增字段的起始值，值设置不同
replicate-do-db=test    #同步的数据库，多个写多行
replicate-ignore-db = information_schema  #不同步的数据库，多个写多行
server_id = 1                              #每台设置不同
log_bin = mysql-bin
log_slave_updates                        #当一个主故障，另一个立即接管
sync-binlog=1                              #每条自动更新，安全性高，默认是0
[root@db1 ~]# service mysqld restart
```

三、配置 db1 和 db2 主主同步

#先查看下 log bin 日志和 pos 值位置

```
mysql> show master status;
+------------------+----------+--------------+--------------------------+
| File             | Position | Binlog_Do_DB | Binlog_Ignore_DB         |
+------------------+----------+--------------+--------------------------+
| mysql-bin.000002 |      106 | test         | mysql, information_schema |
+------------------+----------+--------------+--------------------------+
1 row in set (0.00 sec)
```

db1 配置如下：

```
[root@db1 ~]# mysql -u root -p123.com
mysql> GRANT REPLICATION SLAVE ON *.* TO 'replication'@'192.168.0.%' IDENTIFIED
  BY 'replication';
mysql> flush privileges;
mysql> change master to
        -> master_host='192.168.0.203',
        -> master_user='replication',
        -> master_password='replication',
        -> master_log_file='mysql-bin.000002',
-> master_log_pos=106;    #对端状态显示的值
mysql> start slave;          #启动同步
```

db2 配置如下：

```
[root@db2 ~]# mysql -u root -p123.com
mysql> GRANT REPLICATION SLAVE ON *.* TO 'replication'@'192.168.0.%' IDENTIFIED
```

```
  BY  'replication';
mysql>  flush  privileges;
mysql>  change  master  to
        ->  master_host='192.168.0.202',
        ->  master_user='replication',
        ->  master_password='replication',
        ->  master_log_file='mysql-bin.000002',
        ->  master_log_pos=106;
mysql>  start  slave;    #启动同步
```

#主主同步配置完毕，查看同步状态 Slave_IO 和 Slave_SQL 是 YES 说明主主同步成功。

```
mysql> show slave status\G;
*************************** 1. row ***************************
               Slave_IO_State: Waiting for master to send event
                  Master_Host: 192.168.0.203
                  Master_User: replication
                  Master_Port: 3306
                Connect_Retry: 60
              Master_Log_File: mysql-bin.000002
          Read_Master_Log_Pos: 106
               Relay_Log_File: mysqld-relay-bin.000004
                Relay_Log_Pos: 251
        Relay_Master_Log_File: mysql-bin.000002
             Slave_IO_Running: Yes
            Slave_SQL_Running: Yes
              Replicate_Do_DB: test
          Replicate_Ignore_DB: information_schema
```

在 db2 插入数据测试下：

```
mysql> use test;
Database changed
mysql> show tables;
Empty set (0.00 sec)

mysql> create table user (number INT(10),name VARCHAR(255));
Query OK, 0 rows affected (0.05 sec)

mysql> insert into user values(01,'zhangsan');
Query OK, 1 row affected (0.00 sec)

mysql> show tables;
+----------------+
| Tables_in_test |
+----------------+
| user           |
+----------------+
1 row in set (0.00 sec)

mysql>
```

在 db2 查看是否同步成功：

```
mysql> use test;
Database changed
mysql> show tables;
+----------------+
| Tables_in_test |
+----------------+
| user           |
+----------------+
1 row in set (0.00 sec)

mysql> select number,name from user;
+--------+----------+
| number | name     |
+--------+----------+
|      1 | zhangsan |
+--------+----------+
1 row in set (0.00 sec)

mysql>
```

可以看到已经成功同步过去，同样在 db2 插入到 user 表数据，也能同步过去。我们的双主就成功了，开始做主从复制。

四、配置 slave1 和 slave2 做为 master1 的从库

#先看下 master1 状态值

```
mysql> show master status;
+------------------+----------+--------------+--------------------------+
| File             | Position | Binlog_Do_DB | Binlog_Ignore_DB         |
+------------------+----------+--------------+--------------------------+
| mysql-bin.000002 |      434 | test         | mysql, information_schema |
+------------------+----------+--------------+--------------------------+
1 row in set (0.00 sec)

mysql>
```

在 slave1 和 slave2 分别执行：

```
mysql>  change  master  to
    ->  master_host='192.168.0.202',
    ->  master_user='replication',
    ->  master_password='replication',
    ->  master_log_file='mysql-bin.000002',
    ->  master_log_pos=434;
```

在 slave1 和 slave2 查看如下说明主从复制成功。但是数据没过来，这是因为主从复制原理只同步配置完后的增删改记录，以后的数据是不能同步的，我们可以把主的数据库备份了，然后在送数据库还原。

```
mysql> show slave status\G;
*************************** 1. row ***************************
               Slave_IO_State: Waiting for master to send event
                  Master_Host: 192.168.0.202
                  Master_User: replication
                  Master_Port: 3306
                Connect_Retry: 60
              Master_Log_File: mysql-bin.000002
          Read_Master_Log_Pos: 434
               Relay_Log_File: mysqld-relay-bin.000002
                Relay_Log_Pos: 251
        Relay_Master_Log_File: mysql-bin.000002
             Slave_IO_Running: Yes
            Slave_SQL_Running: Yes
              Replicate_Do_DB: test
          Replicate_Ignore_DB: information_schema
```

```
[root@db1 ~]# mysqldump -uroot -p123.com test > test.sql
[root@db1 ~]# scp test.sql root@192.168.0.204:/root/
[root@db1 ~]# scp test.sql root@192.168.0.205:/root/
[root@db3 ~]# mysql -u root -p123.com test < test.sql
[root@db4 ~]# mysql -u root -p123.com test < test.sql
```

五、MySQL-MMM 安装配置

CentOS 默认没有 mysql-mmm 软件包，官方推荐使用 epel 的网络源，五台都安装 epel：

```
rpm -ivh http://mirrors.ustc.edu.cn/fedora/epel/6/x86_64/epel-release-6-8.noarch.rpm
```

1、monitor 节点安装

```
[root@monitor ~]#  yum -y install mysql-mmm-monitor
```

2、四台 db 节点安装

```
[root@db1 ~]# yum -y install mysql-mmm-agent
```

3、在四台 db 节点授权 monitor 访问

```
[root@db ~]# mysql -u root -p123.com
mysql> GRANT REPLICATIONCLIENT ON *.* TO 'mmm_monitor'@'192.168.0.%' IDENTIFIED
BY 'monitor';
mysql> GRANT SUPER,REPLICATION CLIENT, PROCESS ON *.* TO 'mmm_agent'@'192.168.0
.%' IDENTIFIED BY'agent';
```

4、修改 mmm_common.conf 文件（五台相同）

```
[root@monitor ~]# vi /etc/mysql-mmm/mmm_common.conf
active_master_role          writer
<host default>
        cluster_interface               eth0
        pid_path                                /var/run/mysql-mmm/mmm_agentd.pid
```

```
        bin_path                                /usr/libexec/mysql-mmm/
        replication_user                replication
        replication_password        replication
        agent_user                              mmm_agent
        agent_password                  agent
</host>
<host  db1>
        ip          192.168.0.202
        mode        master
        peer        db2
</host>
<host  db2>
        ip          192.168.0.203
        mode        master
        peer        db1
</host>
<host  db3>
        ip          192.168.0.204
        mode        slave
</host>
<host  db4>
        ip          192.168.0.205
        mode        slave
</host>
<role  writer>
        hosts     db1,  db2
        ips         192.168.0.211
        mode        exclusive         #只有一个host可以writer，一般写操作是这个模式
</role>
<role  reader>
        hosts     db3,  db4
        ips         192.168.0.212,192.168.0.213
        mode        balanced            #多个host可以reader，一般读操作是这个模式
</role>
```

#通过 scp 命令传送到其他四台：

```
scp /etc/mysql-mmm/mmm_common.conf root@192.168.0.202/203/204/205:/etc/mysql-mmm/
```

5、修改四台 db 代理端 mmm_agent.conf 文件

```
[root@db  ~]#  vi  /etc/mysql-mmm/mmm_agent.conf
include  mmm_common.conf
this  db1       #分别修改为本机的主机名，即db1、db2、db3和db4
```

6、修改管理端 mmm_mon.conf 文件

```
[root@monitor  ~]#  vi  /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                        192.168.0.202,192.168.0.203,192.168.0.204,192.1
68.0.205
#真实数据库IP，来检测网络是否正常
        auto_set_online          10     #恢复后自动设置在线的时间
</monitor>
<host  default>
        monitor_user                mmm_monitor
        monitor_password        monitor
</host>
debug  0
```

六、启动 MySQL-MMM

1、db 代理端启动

```
[root@db1 ~]# /etc/init.d/mysql-mmm-agent start

[root@db1 ~]# chkconfigmysql-mmm-agent on
```

2、monitor 管理端启动

```
[root@monitor ~]# /etc/init.d/mysql-mmm-monitor start

[root@monitor ~]# chkconfigmysql-mmm-monitor on
```

七、测试集群

1、查看集群状态

```
[root@monitor ~]# mmm_control show
  db1(192.168.0.202) master/HARD_OFFLINE. Roles:
  db2(192.168.0.203) master/ONLINE. Roles: writer(192.168.0.211)
  db3(192.168.0.204) slave/REPLICATION_FAIL. Roles:
  db4(192.168.0.205) slave/REPLICATION_FAIL. Roles:

[root@monitor ~]#
```

由此看来，主 db1 是对外一个写入的角色，但不真正提供只写，要想实现读写分离还需要结合 amoeba。后面的虚拟 IP 是真正来访问 Mysql 数据库的。

2、故障转移切换

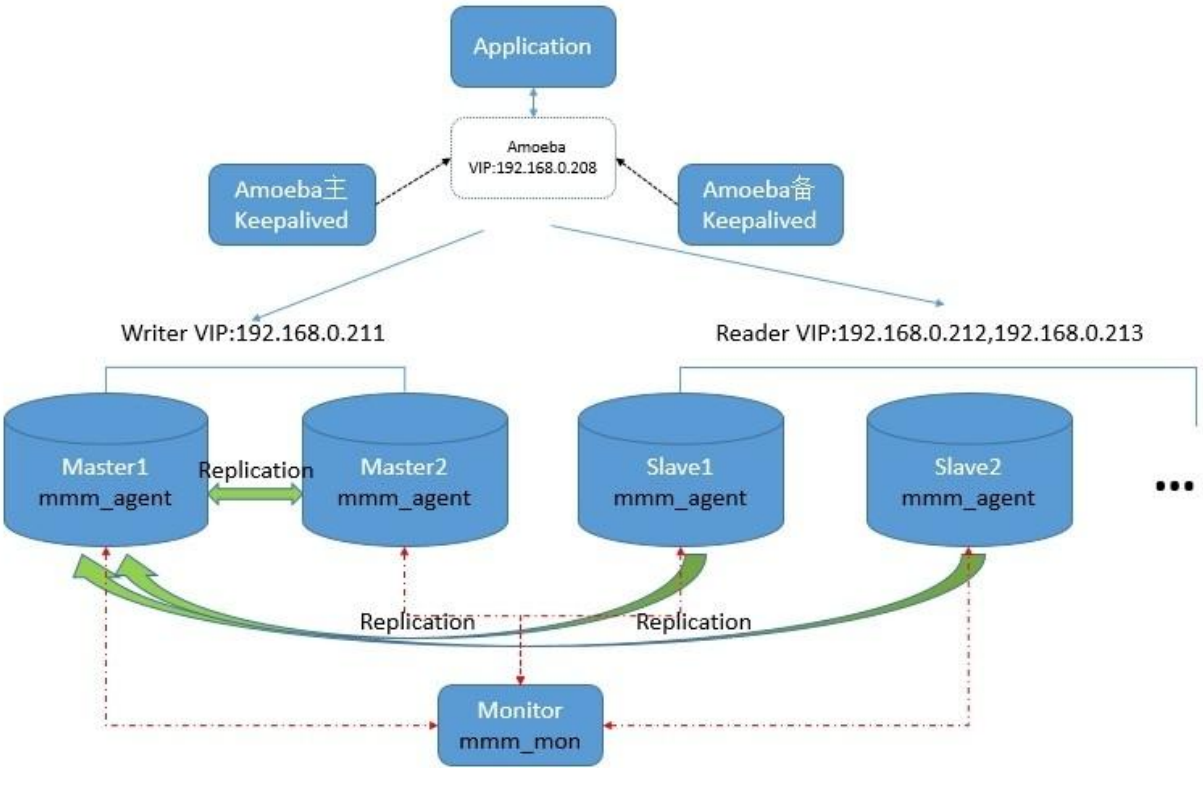
停掉主 db1 数据库，等待几秒后，可以看到数据库 db1 处于 HARD_OFFLINE（离线状态），检测不到数据库的存在。

```
[root@monitor ~]# mmm_control show
  db1(192.168.0.202) master/AWAITING_RECOVERY. Roles:
  db2(192.168.0.203) master/ONLINE. Roles: writer(192.168.0.211)
  db3(192.168.0.204) slave/ONLINE. Roles: reader(192.168.0.213)
  db4(192.168.0.205) slave/ONLINE. Roles: reader(192.168.0.212)

[root@monitor ~]#
```

启动主 db1 数据库后，可以看到数据库 db1 处于 AWAITING_RECOVER（恢复状态），几秒后将恢复在线状态。模拟 Slave 故障也是如此，DOWN 掉一个，虚拟 IP 会全部在另一台正常数据库上。

至此，MySQL-MMM 架构配置完毕。后续会写在此基础上实现读写分离、负载均衡机制。如图：



相关 PDF 文档可以到 Linux 公社资源站下载：

----------分割线----------

免费下载地址在 http://linux.linuxidc.com/

用户名与密码都是 www.linuxidc.com

具体下载目录在 /2017 年资料/4 月/15 日/MySQL 高可用集群之 MySQL-MMM/

下载方法见 http://www.linuxidc.com/Linux/2013-07/87684.htm

----------分割线----------

本文永久更新链接地址：http://www.linuxidc.com/Linux/2017-04/142819.htm