MySQL高可用方案之MMM

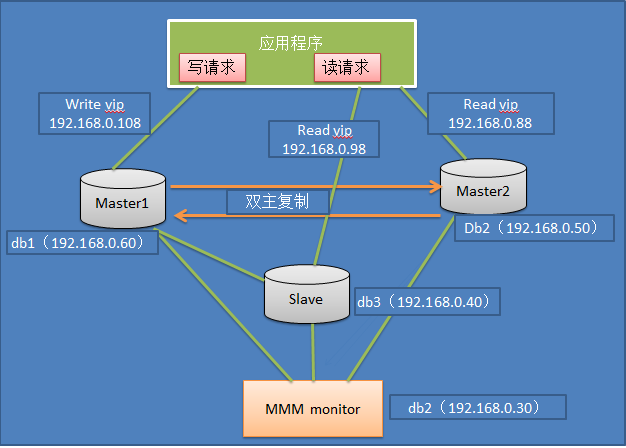
MMM（Master-Master replication manager for MySQL）是一套支持双主故障切换和双主日常管理的脚本程序。MMM使用Perl语言开发，主要用来监控和管理MySQL Master-Master（双主）复制，虽然叫做双主复制，但是业务上同一时刻只允许对一个主进行写入，另一台备选主上提供部分读服务，以加速在主主切换时刻备选主的预热，可以说MMM这套脚本程序一方面实现了故障切换的功能，另一方面其内部附加的工具脚本也可以实现多个slave的read负载均衡。

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

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：

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

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：

[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 ~]#

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。**