# [Mysql实现企业级数据库主从复制架构实战](https://www.cnblogs.com/along21/p/8011596.html)

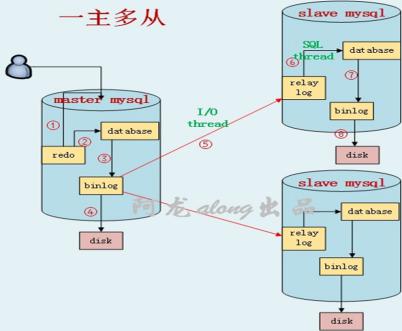
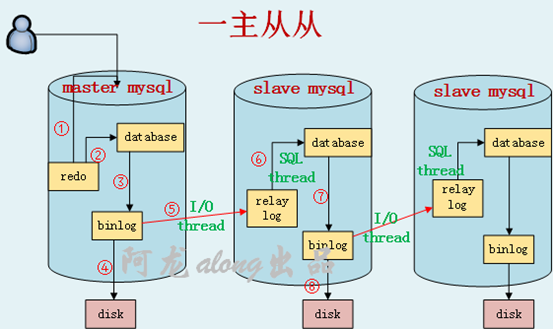
　　环境背景：公司规模已经形成，用户数据已成为公司的核心命脉，一次老王一不小心把数据库文件删除，通过mysqldump备份策略恢复用了两个小时，在这两小时中，公司业务中断，损失100万，老王做出深刻反省，公司因此对于数据库的性能和可靠性提出更高要求。

要求对数据库进行改造，使其承载力进行提升，故障修复时间减少，有没有能实现的方案呢？

### **1、实验架构及其原理**

数据库高可用架构分为：

* 主从：一主一从，一主多从，一主从从
* 双主：

### **2、过程分析**

****（1）********主数据库（innodb引擎）的操作：****

① 一个写的请求，先写到**redo事务日志**中，

② mysql的进程读事务日志，**事务日志的内容**做到**数据库内存**中；此时可以回复客户端，数据为脏数据

③ 请求的操作记录到**二进制日志**中

④ 二进制日志再写**磁盘**中写；优化策略，变随机写为顺序写

****（2）********从数据库的操作：****

⑤ **I/O thread**线程：从主的数据库上，把**二进制文件**的内容**拉**过来，写到**relay log中继日志**中

⑥ **SQL thread**线程：把**relay log**内容**拉**出来，写到**数据库内存**中

⑦ 从数据库也可以把执行的操作记录到自己的**二进制文件**中，**非必须**

⑧ 从数据库的二进制写到自己的**磁盘**中

### **3、主从的优势**

一个master 写入，多个slave同时读出；大大提高了读的效率

现实中，很多都是读的请求大，写的请求相对小的多，如电商网站，大多都是人们去访问，下单的较少；所以主从的关系已经能很好的提高性能了

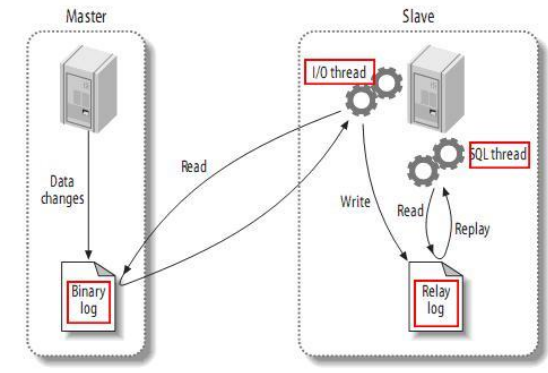
### **4、实验前准备**

① iptables -F && setenforce 清空防火墙策略，关闭selinux

② 拿两台服务器都使用yum 方式安装Mysql 服务，要求版本一致

③ 分别启动两台服务器mysql

## 实战一：Mysql实现数据库简单一主多从复制实战



### **1、环境准备**

**centos 系统服务器3 台、一台用户做Mysql 主服务器，2台用于做Mysql 从服务器**，配置好yum 源、 防火墙关闭、各节点时钟服务同步、各节点之间可以通过主机名互相通信

|  |  |  |  |
| --- | --- | --- | --- |
| 机器名称 | IP配置 | 服务角色 | 备注 |
| master-mysql | 192.168.30.107 | 主数据库 | 二进制日志 |
| slave-mysql1 | 192.168.30.7 | 从数据库 | 中继日志 |
| slave-mysql2 | 192.168.30.2 | 从数据库 | 中继日志 |

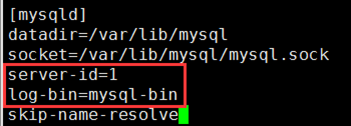
### **2、在主master 主服务器上**

① vim /etc/my.cnf 修改mysql主配置文件，对master进行配置，包括打开二进制日志，指定唯一的servr ID

server-id=1 #配置server-id，让主服务器有唯一ID号，一般为ip的最后一位

log-bin=mysql-bin #打开mysql日志，日志格式为二进制

skip-name-resolve #关闭名称解析，（非必须）



systemctl start mariadb 开启服务（service mysqld start）

② 创建并授权slave mysql 用的复制帐号

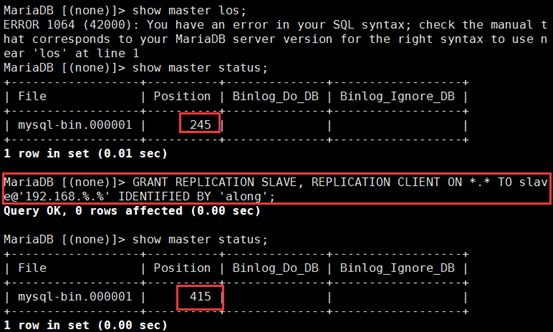
GRANT REPLICATION SLAVE, REPLICATION CLIENT ON \*.\* TO slave@'192.168.%.%' IDENTIFIED BY 'along';

GRANT REPLICATION SLAVE, REPLICATION CLIENT ON \*.\* TO slave@'192.168.%.%' IDENTIFIED BY '**123456**';

分析：在Master的数据库中建立一个备份帐户：每个slave使用标准的MySQL用户名和密码连接master。进行复制操作的用户会授予REPLICATION SLAVE权限。

③ 查看主服务器状态

在Master的数据库执行show master status，查看主服务器二进制日志状态，位置号



### **3、在从slave mysql1上**

① 修改主配置文件

vim /etc/my.cnf 打开中继日志，指定唯一的servr ID，设置只读权限

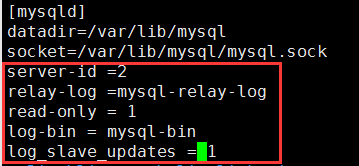
server-id=2 #配置server-id，让从服务器有唯一ID号

relay-log = mysql-relay-bin #打开Mysql日志，日志格式为二进制

read-only = 1 #设置只读权限

log-bin = mysql-bin #开启从服务器二进制日志，（非必须）

log\_slave\_updates = 1 #使得更新的数据写进二进制日志中



systemctl start mariadb 开启服务（service mysqld start）

② 启动从服务器复制线程，让slave连接master，并开始重做master二进制日志中的事件。

**reset master; # 清空所有 binlog 文件**

MariaDB [(none)]> change master to master\_host='192.168.30.107',

-> master\_user='slave',

-> master\_password='along',

-> master\_log\_file='mysql-bin.000001',

-> master\_log\_pos=245;

MariaDB [(none)]> start slave; # 启动复制线程，就是打开I/O线程和SQL线程；实现拉主的bin-log到从的relay-log上；再从relay-log写到数据库内存里

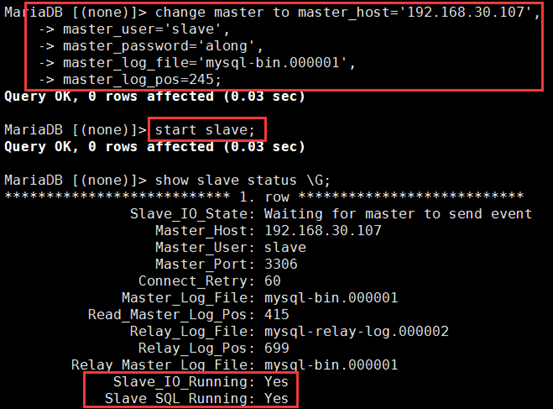
change master to master\_host='192.168.199.211',master\_user='slave',master\_password='123456', master\_log\_file='mysql-bin.000001',master\_log\_pos=154;

③ 查看从服务器状态

可使用SHOW SLAVE STATUS\G查看从服务器状态，如下所示，也可用show processlist \G查看当前复制状态：

Slave\_IO\_Running: Yes #IO线程正常运行

Slave\_SQL\_Running: Yes #SQL线程正常运行

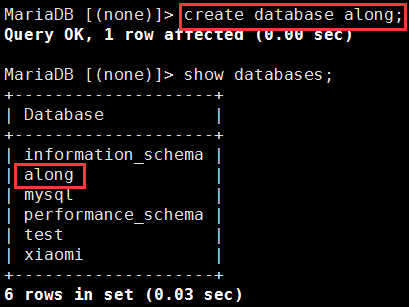


### **4、测试**

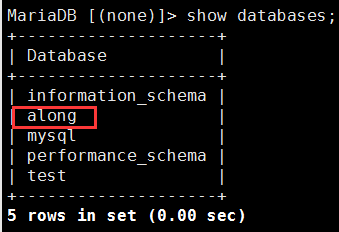
① 在主上创建一个along库

create database test01;

show databases;



② 从上自动生成along数据库（show databases;）



### **5、若要继续添加新salve，实现一主多从**

假如master 已经运行很久了，想对新安装的slave 进行数据同步，甚至它没有master 的数据。

（1）在主master-mysql 上

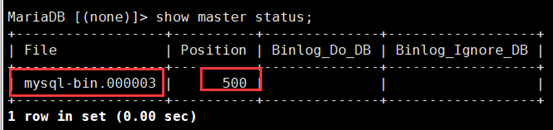
① 进行完全备份

mysqldump --all-databases > /backup/mysql-all-backup-`date +%F-%T`.sql

把备份生成的文件发给salve-mysql2机器上scp /backup/mysql-all-backup-2017-11-20-22\:04\:06.sql @192.168.30.2:

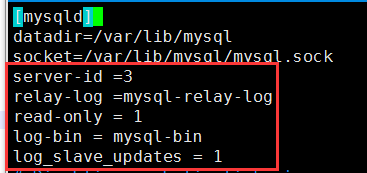
② 查看现在的二进制文件状态

MariaDB [(none)]> show master status;



（2）在从slave-mysql2上

① vim /etc/my.cnf 修改主配置文件，设为从

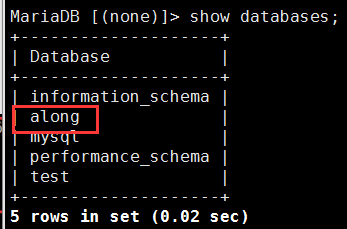


② 进行master的完全备份恢复

mysql -uroot -p < mysql-all-backup-2017-11-20-22\:04\:06.sql

systemctl start mariadb 开启服务

恢复完后，数据直接与主完全一致



③ 启动从服务器复制线程

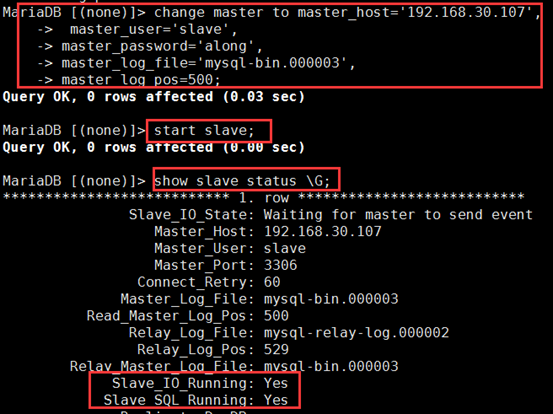
MariaDB [(none)]> change master to master\_host='192.168.30.107',

-> master\_user='slave',

-> master\_password='along',

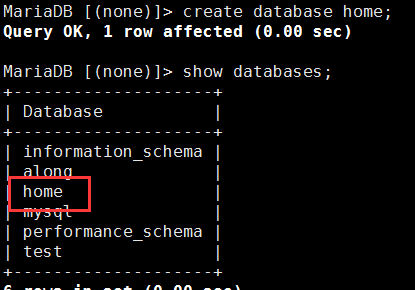
-> master\_log\_file='mysql-bin.000003',

-> master\_log\_pos=500;

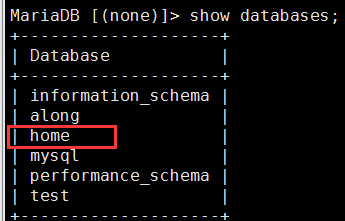


### **6、测试三台机器的一主多从关系**

① 在master 上创建home数据库

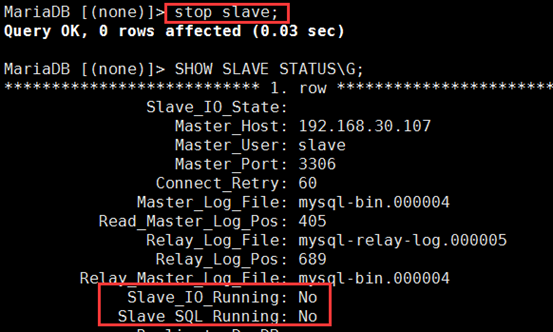


② 在slave-mysql1 和2 上自动生成home库



### **7、解除主从关系，恢复独立的服务器**

① MariaDB [(none)]> stop slave; 关闭两个线程



② vim /etc/my.cnf 删除3行

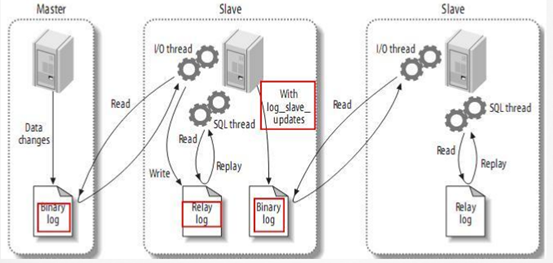
relay-log =mysql-relay-log

read-only = 1

log\_slave\_updates = 1

③ systemctl restart mariadb 重启服务

## **实战二：实现主从从架构及复制过滤器**



****架构原理：****一个主master，一个从slave1；从slave1再做主，另一个slave2以他为主做从；大体做法与上实验相似

****复制过滤原理：****复制过滤器：（黑、白名单）仅复制有限一个或几个数据库相关的数据，而非所有；由复制过滤器进行；

有两种实现思路：

(1) 主服务器

主服务器仅向二进制日志中记录有关特定数据库相关的写操作；****binlog\_do\_db****= #仅允许从复制这个库的二进制日志

binlog\_****ignore****\_db= #除了这个库，其他都允许复制

(2) 从服务器

从服务器的SQL THREAD仅重放关注的数据库或表相关的事件，并将其应用于本地；****Replicate\_Do\_DB=**** #只复制主的这个数据库数据

Replicate\_****Ignore****\_DB= #除了这个都复制

### **1、环境准备**

|  |  |  |  |
| --- | --- | --- | --- |
| 机器名称 | IP配置 | 服务角色 | 备注 |
| master-mysql | 192.168.30.107 | 主数据库 | 二进制日志 |
| slave-mysql1 | 192.168.30.7 | 从数据库 | 中继日志 |
| slave-mysql2 | 192.168.30.2 | 从数据库 | 中继日志 |

### **2、在主master 主服务器上**

① vim /etc/my.cnf 修改mysql主配置文件，对master进行配置，打开二进制日志，指定唯一的servr ID，设置复制过滤

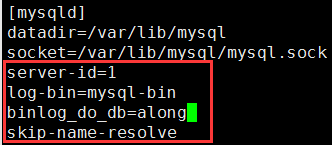
server-id=1 #配置server-id，让主服务器有唯一ID号

log-bin=mysql-bin #打开Mysql日志，日志格式为二进制

skip-name-resolve #关闭名称解析，（非必须）

binlog\_ignore\_db=home #除了home数据库，其他都允许从复制主的二进制文件

#binlog\_do\_db=along #仅允许从复制along数据库的二进制文件



systemctl start mariadb 开启服务

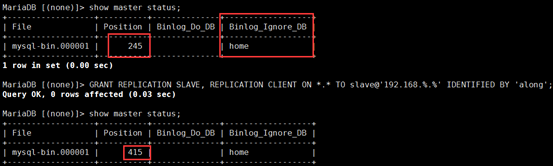
② ****创建并授权****slave mysql 用的复制帐号

GRANT REPLICATION SLAVE, REPLICATION CLIENT ON \*.\* TO slave@'192.168.%.%' IDENTIFIED BY 'along';

分析：在Master的数据库中建立一个备份帐户：每个slave使用标准的MySQL用户名和密码连接master。进行复制操作的用户会授予REPLICATION SLAVE权限。

③ 查看主服务器状态

在Master的数据库执行show master status，查看主服务器二进制日志状态，位置号



### **3、在从slave mysql1上**

① 修改主配置文件

vim /etc/my.cnf 打开中继日志，指定唯一的servr ID，设置只读权限

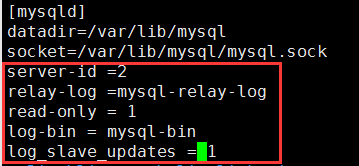
server-id=2 #配置server-id，让从服务器有唯一ID号

relay\_log = mysql-relay-bin #打开Mysql日志，日志格式为二进制

read\_only = 1 #设置只读权限

log\_bin = mysql-bin #开启从服务器二进制日志，（必须）

log\_slave\_updates = 1 #使得更新的数据写进二进制日志中



systemctl start mariadb 开启服务

② ****启动从服务器复制线程****，让slave连接master，并开始重做master二进制日志中的事件。

MariaDB [(none)]> change master to master\_host='192.168.30.107',

-> master\_user='slave',

-> master\_password='along',

-> master\_log\_file='mysql-bin.000001',

-> master\_log\_pos=245;

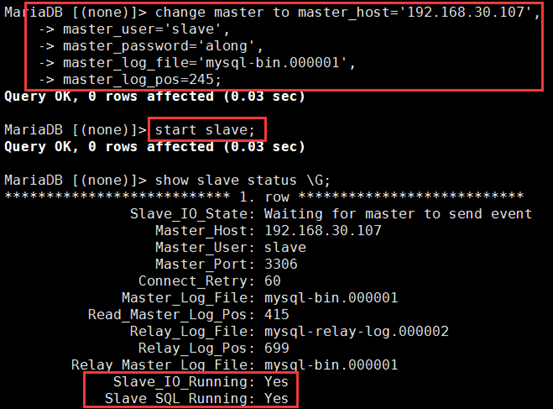
MariaDB [(none)]> start slave; # 启动复制线程，就是打开I/O线程和SQL线程；实现拉主的bin-log到从的relay-log上；再从relay-log写到数据库内存里

③ 查看从服务器状态

可使用SHOW SLAVE STATUS\G查看从服务器状态，如下所示，也可用show processlist \G查看当前复制状态：

Slave\_IO\_Running: Yes #IO线程正常运行

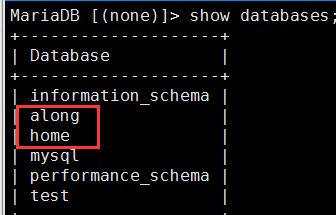
Slave\_SQL\_Running: Yes #SQL线程正常运行



### **4、测试主从和复制过滤**

（1）测试主从关系

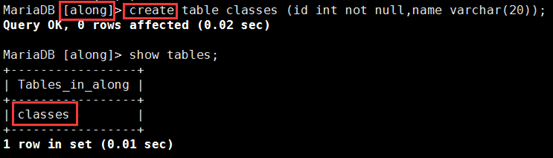
在主上创建一个along、home库；从上自动生成along、home数据库



（2）测试****复制过滤****

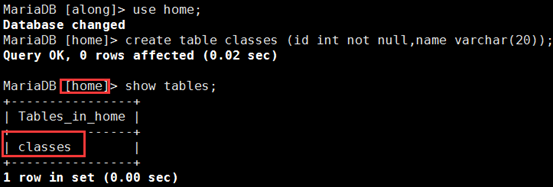
① 在主上：在along库中创建一个classes的表；从上自动生成

MariaDB [home]> create table classes (id int not null,name varchar(20));

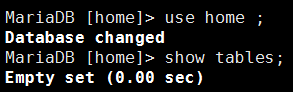


② 在主上：在home库中创建一个classes的表；从上没有生成

MariaDB [home]> create table classes (id int not null,name varchar(20));



slave-mysql1 上，过滤成功

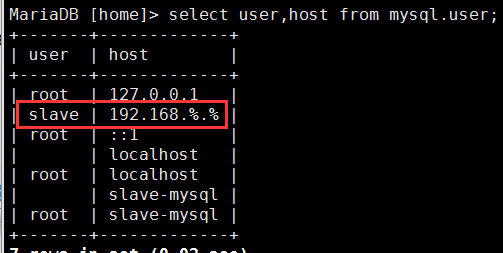


### **5、设置slave-mysql2 为slave-mysql1 的从，且在mysql2 设置复制过滤**

（1）在slave-mysql1上，不用怎么设置

因为上边主配置文件已经开启了自己的二进制文件；且slave-mysql1 是从开始就同步master的，所以授权命令也同步过了

MariaDB [home]> select user,host from mysql.user; 可以查看自己授权过的用户



（2）slave-mysql2 上，可以像上实验一样，先给主的完全备份在本机恢复一下

① 在主上完备

mysqldump --all-databases > /backup/mysql-all-backup-`date +%F-%T`.sqlscp /backup/mysql-all-backup-2017-11-21-11:14:59.sql @192.168.30.2:

② 进行master的完全备份恢复

mysql -uroot -p < mysql-all-backup-2017-11-20-22\:04\:06.sql

③ 在slave-mysql2 上

vim /etc/my.cnf 修改主配置文件，设为从；且设置过滤

server-id =3

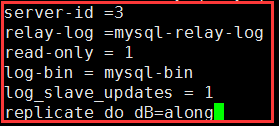
relay-log =mysql-relay-log

read-only = 1

log-bin = mysql-bin

log\_slave\_updates = 1

replicate\_do\_dB=along #只复制它的主的along数据库



systemctl start mariadb 开启服务

④ mysql 打开数据库，查看数据恢复成功；

****启动从服务器复制线程****，让slave连接master，并开始重做master二进制日志中的事件。

MariaDB [(none)]> change master to master\_host='192.168.30.7',

-> master\_user='slave',

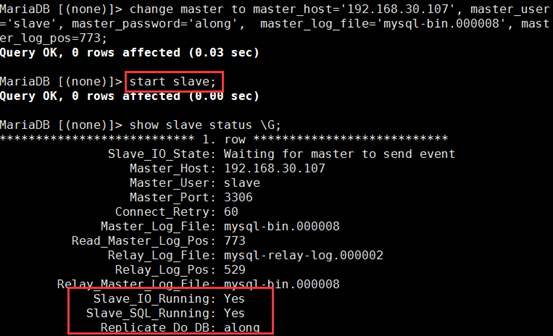
-> master\_password='along',

-> master\_log\_file='mysql-bin.000008',

-> master\_log\_pos=773;

MariaDB [(none)]> start slave;

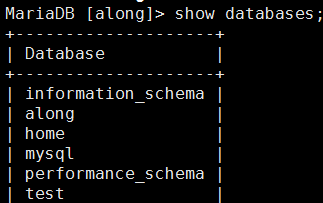
⑤ MariaDB [(none)]> show slave status \G; 查看，两个进程打开，且只复制主的along数据库



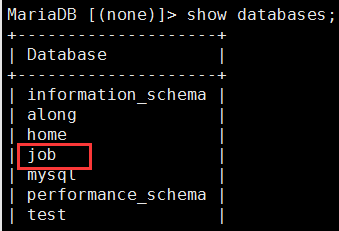
### **6、测试主从从和slave-mysql2的复制过滤**

（1）在主上删除job数据库，master 和slave-mysql1 都删除成功

MariaDB [home]> drop database job; 删除job库

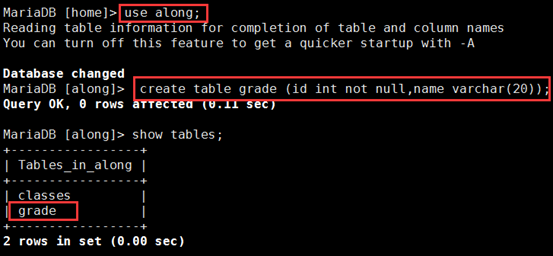


因为slave-mysql2 只同步slave-mysql1 的along库，所以没有删除

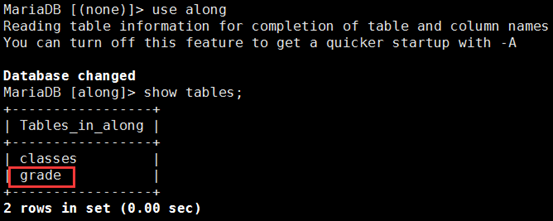


（2）在主上的along数据库，创建一个grade 表，master 和slave-mysql1 都删除成功

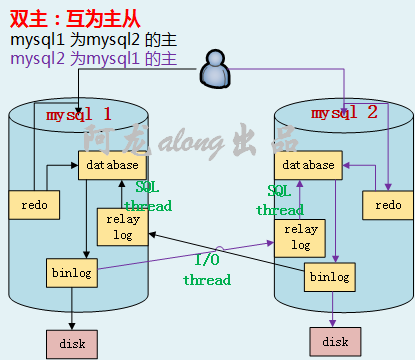
MariaDB [along]> create table grade (id int not null,name varchar(20));



在slave-mysql2 上也自动生成成功



## **实战三：mysql数据库双主的实现**



****原理：****双主就是双方互为主从

为了解决双主同时对一个数据库进行写入，采用自增长ID来解决，两个mysql写入用奇偶ID岔开

① 创建表，设置ID为自增长

create table userInfo (id int PRIMARY KEY AUTO\_INCREMENT,name varchar(50) NOT NULL);

② 定义一个节点使用奇数id：从1开始，步长为2，

auto\_increment\_increment=2 #表示自增长字段每次递增的量，步长

auto\_increment\_offset=1 #表示自增长字段从那个数开始

③ 另一个节点使用偶数id：从2开始，步长为2，

auto\_increment\_increment=2

auto\_increment\_offset=2

****应用：****只适合小型公司，小并发访问量，毕竟同时写入易出错

### **1、环境准备**

|  |  |  |  |
| --- | --- | --- | --- |
| 机器名称 | IP配置 | 服务角色 | 备注 |
| mysql1 | 192.168.30.107 | 数据库 | 中继日志、二进制日志 |
| mysql2 | 192.168.30.7 | 数据库 | 中继日志、二进制日志 |

### **2、配置总配置文件，除了ID号和起始数，两边都是一样的**

vim /etc/my/cnf

server-id =1 #mysql1的配置ID为1，mysql2的ID为2

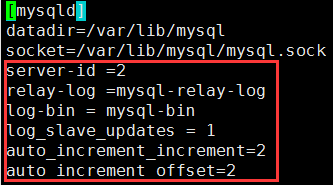
relay-log =mysql-relay-log

log-bin = mysql-bin

log\_slave\_updates = 1

auto\_increment\_increment=2 #表示自增长字段每次递增的量，步长

auto\_increment\_offset=1 #表示自增长字段从那个数开始，mysql1从1开始；mysql2从2开始



systemctl start mariadb

### **3、互相设为对方的从**

（1）授权远程登录的用户

mysql1、2 上

GRANT REPLICATION SLAVE, REPLICATION CLIENT ON \*.\* TO slave@'192.168.%.%' IDENTIFIED BY 'along';

（2）开启复制线程

① mysql1 上

MariaDB [(none)]> change master to master\_host='192.168.30.7',

-> master\_user='slave',

-> master\_password='along',

-> master\_log\_file='mysql-bin.000002',

-> master\_log\_pos=245;

MariaDB [(none)]> start slave; # 启动复制线程

② mysql2 上

MariaDB [(none)]> change master to master\_host='192.168.30.107',

-> master\_user='slave',

-> master\_password='along',

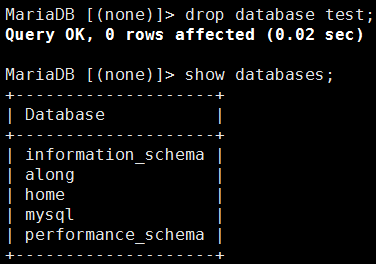
-> master\_log\_file='mysql-bin.000002',

-> master\_log\_pos=245;

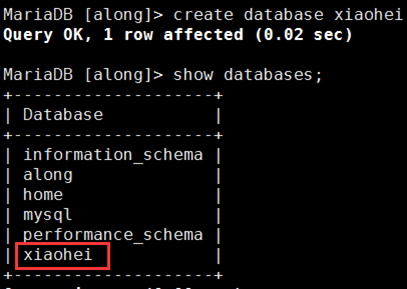
MariaDB [(none)]> start slave; # 启动复制线程

### **4、测试双方互为主从的关系**

在mysql1上，删除test数据库；mysql2 上也自动删除

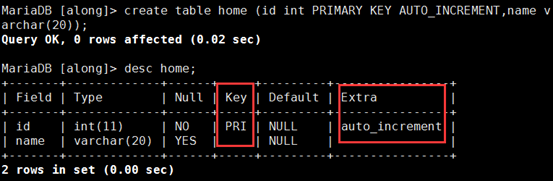


在mysql2上，创建xiaohei数据库；mysql2 上也自动生成



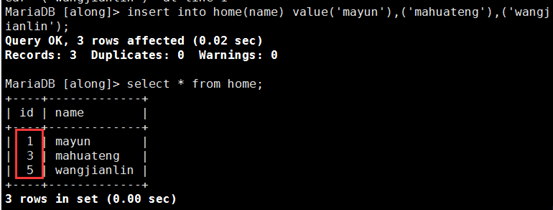
### **5、设置自增长ID的表**

MariaDB [along]> create table home (id int PRIMARY KEY AUTO\_INCREMENT,name varchar(20));



① 在mysql1上向表中插入数据

MariaDB [along]> insert into home(name) value('mayun'),('mahuateng'),('wangjianlin');



② 在mysql2上向表中插入数据

MariaDB [along]> insert into home(name) value('dinglei'),('liyanhong'),('leijun');



## **实验四：实现半同步复制的一主多从**

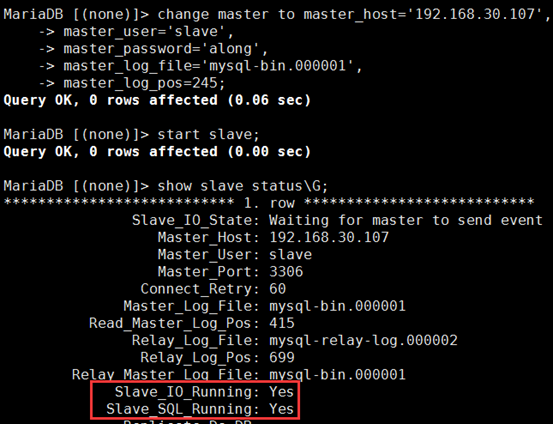
****原理：****介于异步复制和全同步复制之间，主库在执行完客户端提交的事务后不是立刻返回给客户端，而是****等待至少一个从库接收到并写到relay log中才返回给客户端****。相对于异步复制，半同步复制提高了数据的安全性，同时它也造成了一定程度的延迟，这个延迟最少是一个TCP/IP往返的时间。所以，半同步复制最好在低延时的网络中使用。

注意：本来是应该最少2个从mysql，才能有真正的效果，但是原理都是一样的，我就只用了一主一从

### **1、环境准备**

|  |  |  |  |
| --- | --- | --- | --- |
| 机器名称 | IP配置 | 服务角色 | 备注 |
| master-mysql | 192.168.30.107 | 主数据库 | 二进制日志 |
| slave-mysql | 192.168.30.7 | 从数据库 | 中继日志 |

### **2、按照实战一，实现主从**



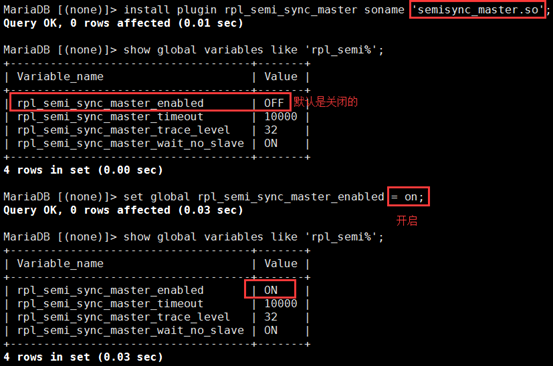
### **3、加载模块，实现半同步**

（1）在主master-mysql 上：

MariaDB [(none)]> install plugin rpl\_semi\_sync\_master soname 'semisync\_master.so'; 加载模块

MariaDB [(none)]> show global variables like 'rpl\_semi%'; 查看是否开启

MariaDB [(none)]> set global rpl\_semi\_sync\_master\_enabled = on; 开启



（2）在从slave-mysql 上：

MariaDB [(none)]> INSTALL PLUGIN rpl\_semi\_sync\_slave SONAME 'semisync\_slave.so';

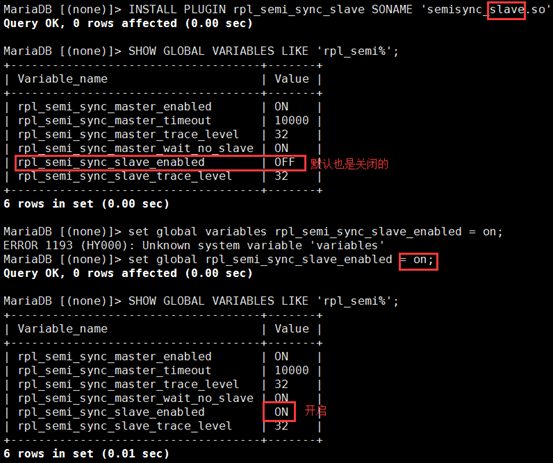
SHOW GLOBAL VARIABLES LIKE 'rpl\_semi%';

set global variables rpl\_semi\_sync\_slave\_enabled = on;

为了主从同步，再重启启动下slave 两个进程

MariaDB [(none)]> stop slave;

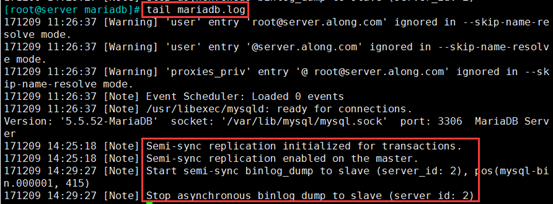
MariaDB [(none)]> start slave;



### **4、查询日志，验证**

cd /var/log/mariadb/

tail mariadb.log



注释：

① 用于事务的半同步复制。

② 在主服务器上启用了半同步复制。

③ 启动半同步的binlog 转储到slave (id:2)上

④ 停止异步的二进制文件转储

## **实验五：实现MHA 高可用mysql数据库架构**

****原理：****总之就是当主master mysql宕机时，从slave mysql顶上去的一系列操作

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****;

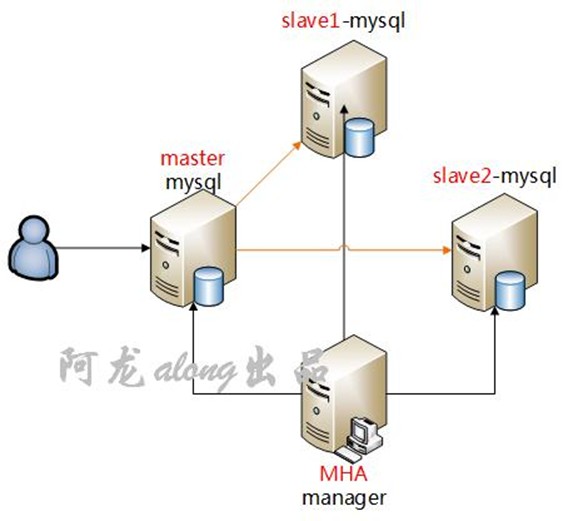
③ 应用差异的****中继日志(relay log) 到其他slave;****

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master****;

⑥ 使用****其他的slave 连接新的master****

****架构图****



### **1、环境准备**

|  |  |  |  |
| --- | --- | --- | --- |
| 机器名称 | IP配置 | 服务角色 | 备注 |
| master-mysql | 192.168.30.107 | 主数据库 | 二进制日志、中继日志 |
| slave-mysql1 | 192.168.30.7 | 从数据库 | 二进制日志、中继日志 |
| slave-mysql2 | 192.168.30.2 | 从数据库 | 二进制日志、中继日志 |
| MHA manager | 192.168.30.3 | MHA的管理节点 |  |

### **2、实现三台服务器的一主多从**

****注意点：****

① 每个节点****都需开启二进制和中继日志****，因为主会宕机，当主的机器修复完毕，可以作为从继续使用，所以中继日志是必须的；从也会在主宕机的时候，顶为主，所以二进制日志也是必须的

② 各从节点必须显示启用其****read-only 属性****，并****关闭relay\_log\_purge****清理中继日志的功能

③ 注意每个mysql 的****server-id都不能相同****

（1）vim /etc/my.cnf 修改配置文件

① 主的配置文件

server-id=1

log-bin=mysql-bin

relay-log=mysql-relay-log

skip-name-resolve

② 从的配置文件，各个从的配置文件除了ID，其他都相同

server-id =2[/3] #各自对应自己的id

relay-log =mysql-relay-log

log-bin = mysql-bin

read\_only = on

relay\_log\_purge = 0

skip\_name\_resolve

systemctl start mariadb 启动服务

（2）在主上：授权

MariaDB [(none)]> grant replication slave,replication client on \*.\* to slave@'192.168.30.%' identified by 'along';

（3）在从上：开启I/O，SQL线程，实现主从

MariaDB [(none)]> change master to master\_host='192.168.30.107',

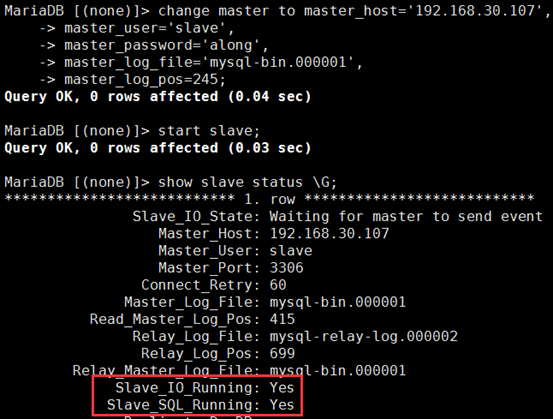
master\_user='slave',

master\_password='along',

master\_log\_file='mysql-bin.000001',

master\_log\_pos=245;

MariaDB [(none)]> start slave ;



### **3、配置MHA的准备**

****（1）MHA的安装****

需安装2个包 rz，我已经放到我网盘里，需要的私聊<http://pan.baidu.com/s/1kV8BCJt>

mha4mysql-manager-0.56-0.el6.noarch.rpm

mha4mysql-node-0.56-0.el6.noarch.rpm

所有节点，包括Manager都需安装：

yum -y****localinstall****mha4mysql-\*

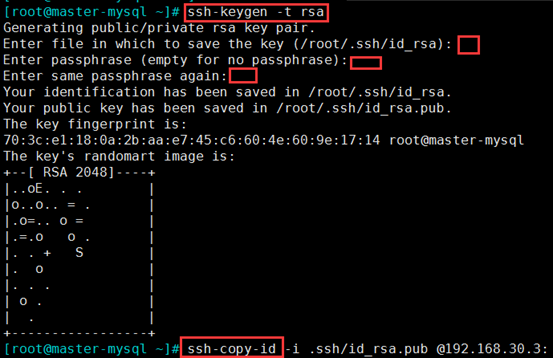
****（2）实现各个节点都基于秘钥认证****

分析：MHA 集群中的各节点彼此之间均需要基于ssh 互信通信，以实现远程控制及数据管理功能。

例：主master 机器：

ssh-keygen -t rsa 生成公私秘钥对，可以直接敲3个回车，不须加密ssh-copy-id -i .ssh/id\_rsa.pub root@192.168.30.7: ssh-copy-id -i .ssh/id\_rsa.pub root@192.168.30.2:ssh-copy-id -i .ssh/id\_rsa.pub root@192.168.30.3: 把公钥发给其他3个机器

注意：每个mysql服务器都需要发送自己的公钥



（3）给MHA manager授权

MariaDB [(none)]>****grant all on \*.\***** to 'mhaadm'@'192.168.30.%' identified by 'along';

注意：MHA manager 需要做很多事，所以给很大的权限；且已经实现主从，所以只需在master上执行授权命令

（4）定义MHA 管理配置文件

mkdir /etc/mha\_master 创建配置文件存放目录

vim /etc/mha\_master/app.cnf 设置配置文件，****注意注释不要加在配置文件中，否则检测不过****

[server default] // 适用于server1,2,3 个server 的配置

user=mhaadm //mha 管理用户

password=along //mha 管理密码

manager\_workdir=/etc/mha\_master/app //mha\_master 自己的工作路径

manager\_log=/etc/mha\_master/manager.log // mha\_master 自己的日志文件

remote\_workdir=/mydata/mha\_master/app // 每个远程主机的工作目录在何处

ssh\_user=root // 基于ssh 的密钥认证

repl\_user=slave // 数据库用户名

repl\_password=along // 数据库密码

ping\_interval=1 // ping 间隔时长

[server1] // 节点1hostname=192.168.30.107 // 节点1 主机地址

ssh\_port=22 // 节点1 的ssh 端口

candidate\_master=1 // 将来可不可以成为master 候选节点/ 主节点[server2]hostname=192.168.30.7

ssh\_port=22

candidate\_master=1

[server3]hostname=192.168.30.2

ssh\_port=22

candidate\_master=1

### **4、开启服务之前的检测**

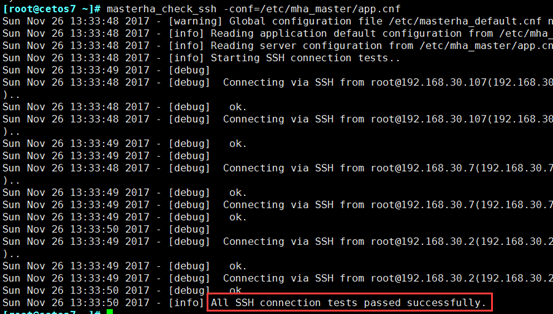
注意：开启服务之前的检查非常有必要，因为mha服务是管理mysql的，所有要求很严格，如果检查通不过，开启服务时会有一堆错误。****每次启动服务之前都需检测环境。****

****① 检测各节点间ssh 互信通信配置是否Ok:****

****masterha\_check\_ssh**** -conf=/etc/mha\_master/app.cnf

　　输出信息最后一行类似如下信息，表示其通过检测。

　　[info]All SSH connection tests passed successfully.



****② 检查管理的MySQL 复制集群的连接配置参数是否OK ：****

****masterha\_check\_repl**** -conf=/etc/mha\_master/app.cnf

　　输出信息最后一行类似如下信息，表示其通过检测。

　　MySQL Replication Health is OK.

③****注意：****如果测试时会报错 ，可能是从节点上没有账号，因为这个架构，任何一个从节点，将有可能成为主节点，所以也需要创建账号。

因此，这里只要在****mater 节点上再次执行****以下操作即可：

　　MariaDB [(none)]> ****grant all on \*.\***** to 'mhaadm'@'192.168.30.%' identified by 'along';

　　MariaDB [(none)]> FLUSH PRIVILEGES;

Manager 节点上再次运行，就显示Ok 了。

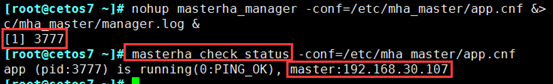
### **5、启动MHA**

（1）开启mha服务

nohup masterha\_manager -conf=/etc/mha\_master/app.cnf &> /etc/mha\_master/manager.log &

启动成功后，可用过如下命令来查看master 节点的状态:

masterha\_check\_status -conf=/etc/mha\_master/app.cnf



app (pid:3777) is running(0:PING\_OK), master:192.168.30.107

上面的信息中"app (pid:3777)is running(0:PING\_OK)" 表示MHA 服务运行OK ，否则，则会显示为类似"app is stopped(1:NOT\_RUNNINg)."

（2）如果要停止MHA ，需要使用master\_stop 命令。

****masterha\_stop****-conf=/etc/mha\_master/app.cnf

### **6、测试MHA 测试故障转移**

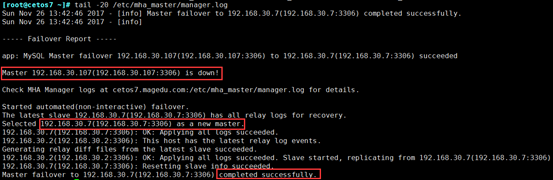
(1) ****在master 节点关闭mariadb 服务****， 模拟主节点数据崩溃

****killall -9**** mysqld mysqld\_safe

****rm -rf**** /var/lib/mysql/\*

(2) 在manager 节点查看日志：

****tail -20 /etc/mha\_master/manager.log****日志文件中出现如下信息，表示manager 检测到192.168.30.107节点故障，而后自动执行故障转移，将192.168.30.7 提升为主节点。



****注意****：故障转移完成后，****manager 将会自动停止****，此时使用

masterha\_check\_status 命令检测将会遇到错误提示，如下所示：

IMG_306

### **7、提供新的从节点以修复复制集群**

原有 master 节点故障后，需要重新准备好一个****新的 MySQL 节点****。基于来自于master 节点的备份恢复数据后，将其配置为新的 master 的从节点即可。注意，新加入的节点如果为****新增节点，其 IP 地址要配置为原来 master 节点的 IP****，否则，还需要修改 app.cnf 中相应的 ip 地址。随后再次启动 manager ，并再次检测其状态。

除了增加新的mysql 节点，也可以将坏掉主mysql 修复，再将其作为从加入集群中。由于机器有限，我就用修复好的主作为从，修复复制集群。

（1）修复主mysql

yum -y remove mariadb-server

yum -y install mariadb-server

（2）在另两个机器上备份，在修复好的机器上恢复

① 在另外正常的机器上备份

****mysqldump --all-databases**** > /backup/mysql-all-backup-`date +%F-%T`.sql

scp /backup/mysql-all-backup-2017-11-26-14\:03\:19.sql @****192.168.30.107:****

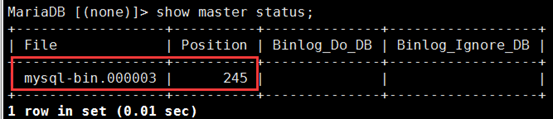
② 在修复的机器上修复

mysql -uroot -p < mysql-all-backup-2017-11-26-14\:03\:19.sql

（3）把修复的机器作为新主的从

① 在新主上查询二进制日志和位置号

MariaDB [(none)]> show master status;



② 在新修复的机器上，设为从，启动线程

MariaDB [(none)]> change master to master\_host='192.168.30.7',

master\_user='slave',

master\_password='along',

master\_log\_file='mysql-bin.000003',

master\_log\_pos=245;

MariaDB [(none)]> start slave;

③ 在新主上重新授权

revoke delete on \*.\* from 'mhaadm'@'192.168.30.%';

revoke delete on \*.\* from 'slave'@'192.168.30.%';

grant replication slave,replication client on \*.\* to slave@'192.168.30.%' identified by 'along';

grant all on \*.\* to 'mhaadm'@'192.168.30.%' identified by 'along';

(4) 新节点提供后再次执行检查操作

masterha\_check\_status -conf=/etc/mha\_master/app.cnf

masterha\_check\_repl -conf=/etc/mha\_master/app.cnf

检查无误，再次运行，这次要记录日志

masterha\_manager -conf=/etc/mha\_master/app.cnf >/etc/mha\_master/manager.log 2>&1 &

### ****8、**新节点上线，故障转换恢复注意事项**

(1) 在生产环境中，当你的主节点挂了后，一定要在从节点上做一个备份，拿着备份文件把主节点手动提升为从节点，并指明从哪一个日志文件的位置开始复制

(2) 每一次自动完成转换后，****每一次的(replication health ) 检测不ok 始终都是启动不了****必须手动修复主节点，除非你改配置文件

(3) 手动修复主节点提升为从节点后，再次运行检测命令

masterha\_check\_repl --conf=/etc/mha\_master/app.cnf

(4) 再次运行起来就恢复成功了

masterha\_manager --conf=/etc/mha\_master/app.cnf