第三十八章：Mysql主从复制及读写分离

**一、Mysql5.7的版本优势；**

**二、数据库集群的概述；**

**三、Mysql数据库的主从复制；**

**四、Mysql数据库的读写分离；**

**五、案例：搭建Mysql数据库集群（主从复制--二进制日志文件方式）；**

**六、案例：搭建Mysql数据库集群（主从复制--GTID方式）；**

**七、案例：实现Mysql主从复制+读写分离；**

**一、Mysql5.7的版本优势；**

**1.安全性：**

MySQL数据库初始化完成以后，会产生一个 root@localhost 用户，从MySQL 5.7开始，root用户的密码不再是空，而是随机产生一个密码；

MySQL官方已经删除了test数据库，默认安装完后是没有test数据库的；

提供了更为简单SSL安全访问配置，并且默认连接就采用SSL的加密方式；

可以为用户设置密码过期策略，一定时间以后，强制用户修改密码；

ALTER USER 'jeffrey'@'localhost' PASSWORD EXPIRE INTERVAL 90 DAY；

可以”锁”住用户，用以暂时禁用某个用户；

ALTER USER 'jeffrey'@'localhost' ACCOUNT LOCK；

ALTER USER l 'jeffrey'@'localhost' ACCOUNT UNLOCK；

**2.灵活性：**

JSON：随着非结构化数据存储需求的持续增长，各种非结构化数据存储的数据库应运而生（如MongoDB）。从最新的数据库使用 排行榜 来看，MongoDB已经超过了PostgreSQL，其火热程度可见一般。各大关系型数据库也不甘示弱，纷纷提供对JSON的支持，以应对非结构化数据库的挑战。MySQL数据库从5.7.8版本开始，也提供了对JSON的支持。其使用方式如下：

CREATE TABLE t1 (jdoc JSON);

INSERT INTO t1 VALUES('{"key1": "value1", "key2": "value2"}');

MySQL支持JSON以后，总是避免不了拿来与MongoDB进行一些比较。但是，MySQL对JSON的支持，至少有两点能够完胜MongoDB：

1.可以混合存储结构化数据和非结构化数据，同时有关系型数据库和非关系型数据库的优点；

2.能够提供完整的事务支持：generated column是MySQL 5.7引入的新特性，所谓generated column，就是数据库中这一列由其他列计算而得；

**3.易用性：**

sys schema是MySQL 5.7.7中引入的一个系统库，包含了一系列视图、函数和存储过程， 该项目专注于MySQL的易用性；

例如，我们可以通过sys schema快速的知道，哪些语句使用了临时表，哪个用户请求了最多的io，哪个线程占用了最多的内存，哪些索引是无用索引等；

**4.性能：**

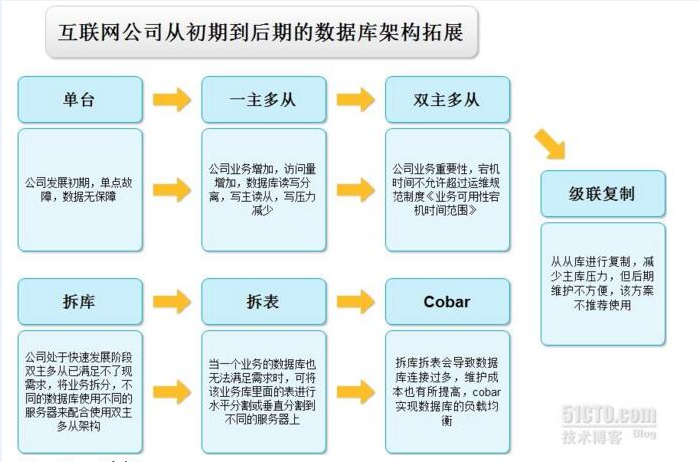
临时表的性能改进：

加速连接处理：在MySQL 5.7之前，变量的初始化操作（THD、VIO）都是在连接接收线程 里面完成的，现在将这些工作下发给工作线程，以减少连接接收线程的工作量，提高连 接的处理速度。这个优化对那些频繁建立短连接的应用，将会非常有用。

复制性能的改进：复制延迟是Mysql很大的问题之一，MySQL 5.7版本已经支持”真正”的 并行复制功能。MySQL 5.7并行复制的思想简单易懂，简而言之，就是”一个组提交的 事务都是可以并行回放的”，因为这些事务都已进入到事务的prepare阶段，则说明事 务之间没有任何冲突（否则就不可能提交）。这里需要注意的是，为了兼容MySQL 5.6 基于库的并行复制，Mysql提供了两个特定了变量；

**二、数据库集群的概述；**

**数据库发展阶段：**



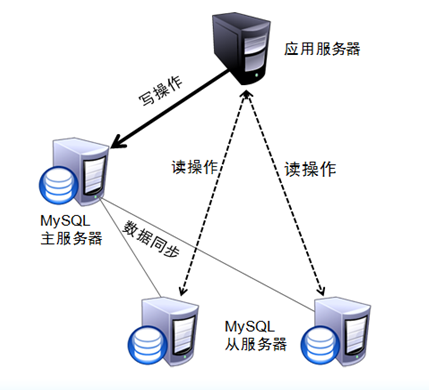
**Mysql数据库存在的瓶颈：**

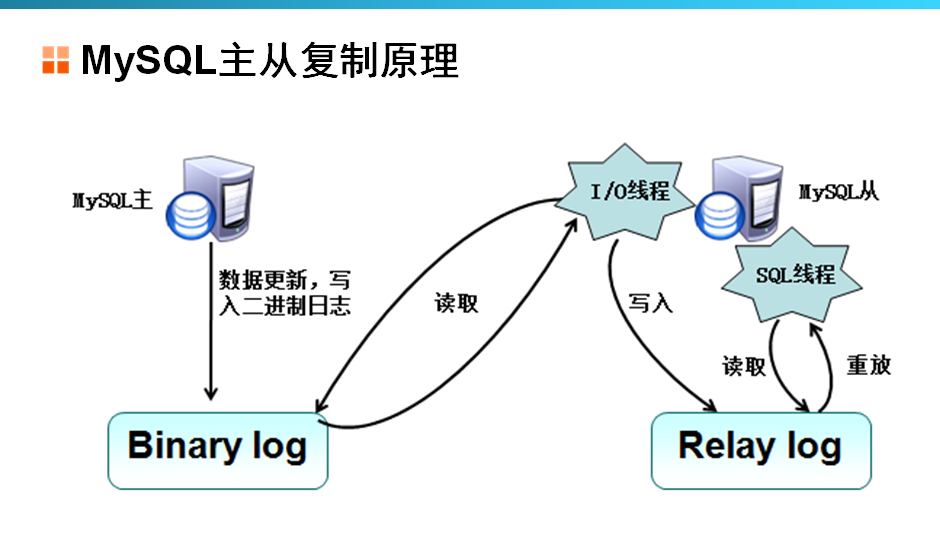
CPU瓶颈：Mysql在单个事务查询中只能使用一个CPU，而且他多CPU支持的不好，多加了也没用，所以我们要提高CPU速度，这里我们简单理解为提高CPU主频，当然64位架构会更佳（即64位OS配合64位的CPU）。

I/O瓶颈：I/O瓶颈发生在装入数据远大于内存容量的时候，磁盘的I/O读写性能有限，当读写速度远不及数据处理时，可以改变磁盘的存储性能（通过物理或逻辑方式RAID），使用类似DRBD等块复制的磁盘方式；

**三、Mysql数据库的主从复制；**

**概述：**单台mysql数据库服务器，需要承载客户端的读取和写入操作的请求，那么若客户端请求过多，服务器会不堪重负，那么可以选择Mysql数据库集群的方式来增加Mysql服务处理数据的效率，像web集群一样，但是不得不考虑的是多台Mysql数据库之间数据的一致性；





**主从复制过程原理：**

1、在每一个事务更新之前，Master在二进制日志记录这些改变，写入二进制日志完成后，Master通知存储引擎提交事务；

2、Slave将Master的Binary log复制到其中继日志，首先Slave开始一个I/O线程，I/O线程在Master的Binary log中读取事件，如果已经跟上Master，他会睡眠并等待Master产生新的事件，I/O线程将这些事件写入到中继日志Relay log，中继日志通常存放在OS的缓存中，所以开销较小；

3、Slave中的Sql线程在中继日志Relay log中读取事件，并重放其中的事件而更新Slave的数据，使其与Master中的数据一致；

**注：**复制过程中，Slave中的数据复制是串行化的，并行操作无法在Slave上实现（复制过程有一个很重要的限制——复制在slave上是串行化的，也就是说master上的并行更新操作不能在slave上并行操作）；

**复制类型：**

1、基于语句的复制：在主服务器上执行的sql语句，在从服务器上同样也会执行，mysql默认选用基于语句的复制，效率比较高，MySQL 5.0及之前的版本仅支持基于语句的复制（也叫做逻辑复制，logical replication），这在数据库并不常见。master记录下改变数据的查询，然后，slave从中继日志中读取事件，并执行它，这些SQL语句与master执行的语句一样。；

2、基于行的复制：把改变的内容复制过去，而不是把命令重新执行一遍；

3、混合类型的复制：默认采用基于语句的复制，一旦发现基于语句无法精确复制时，就会采用基于行；

**主从复制方式：**

基于二进制文件：二进制日志文件；

基于GTID方式：全局事务标示符，自5.6版本开启的新型复制方式，GTID的组成：前面是server\_uuid：后面是一个序列号

例如：server\_uuid：sequence number

7800a22c-95ae-11e4-983d-080027de205a:10

UUID：每个mysql实例的唯一ID，由于会传递到slave，所以也可以理解为源ID。

Sequence number：在每台MySQL服务器上都是从1开始自增长的序列，一个数值对应一个事务。

工作过程：

1、master更新数据时，会在事务前产生GTID，一同记录到binlog日志中。

2、slave端的I/O 线程将变更的binlog，写入到本地的relay log中。

3、sql线程从relay log中获取GTID，然后对比slave端的binlog是否有记录。

4、如果有记录，说明该GTID的事务已经执行，slave会忽略。

5、如果没有记录，slave就会从relay log中执行该GTID的事务，并记录到binlog。

6、在解析过程中会判断是否有主键，如果有就用二级索引，如果没有就用全部扫描。

优势：

1、更简单的实现failover，不用以前那样在需要找二进制日志文件log\_file和位置值log\_pos。

2、更简单的搭建主从复制。

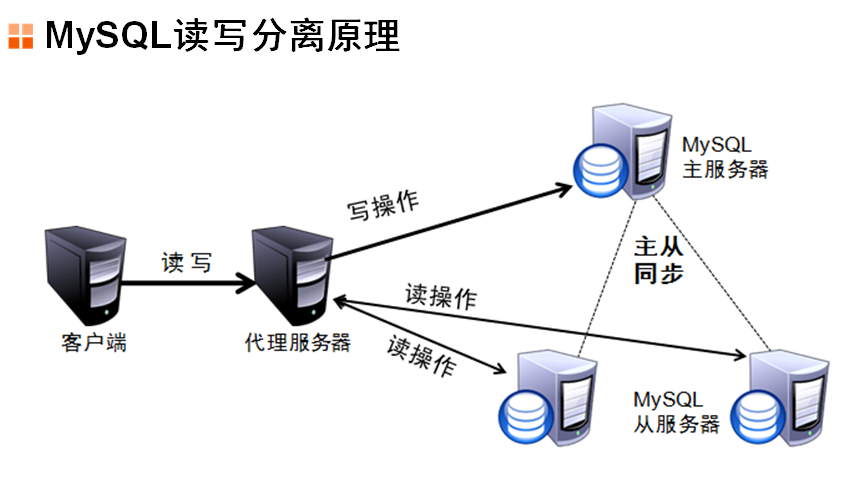
3、比传统的复制更加安全。

4、GTID是连续的没有空洞的，保证数据的一致性，零丢失。

5、slave无需知道master的pos值和日志文件值，只需要知道master的ip、用户名、密码即可；

**四、Mysql数据库的读写分离；**

**读写分离架构：**



**实现方式：**

1.基于程序代码内部实现：

在代码中根据select和insert进行路由分类，性能比较好，无需任何硬件代理条件，缺点是需要开发人员的编写，运维人员无从下手；

2.基于中间代理层实现：

在客户端与数据库间添加代理层，代理服务器接收到客户端的请求之后，判断请求后，转发给后端的数据库；

mysql-proxy：为mysql的开源项目，但是mysql官方不建议将其引用在生产环境中；

amoeba：变形虫，由陈思儒开发，Java语言编写而成，再生产环境中amoeba代理层软件是一个不错的选择；

**五、案例：搭建Mysql数据库集群（主从复制--二进制日志文件方式）；**

**案例环境：**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 系统 | IP地址 | 主机名 | 所需软件 | 硬件 |
| Centos 7.4 1708 64bit | 192.168.100.101 | master.linuxfan.cn | boost\_1\_59\_0.tar.gz  mysql-5.7.12.tar.gz | 内存：4G |
| Centos 7.4 1708 64bit | 192.168.100.102 | slave1.linuxfan.cn | boost\_1\_59\_0.tar.gz  mysql-5.7.12.tar.gz | 内存：4G |
| Centos 7.4 1708 64bit | 192.168.100.103 | slave2.linuxfan.cn | boost\_1\_59\_0.tar.gz  mysql-5.7.12.tar.gz | 内存：4G |

**案例步骤：**

* 在master主机上安装ntp时间服务；
* 在两台slave主机中配置同步master的ntp时间（在此配置相同，在此只列举slave1的配置）；
* 安装三台Mysql数据库服务器（在此步骤相同，在此只列举master主机一台的安装）；
* 优化配置Master主机的mysql服务；
* 优化配置slave1主机的mysql服务；
* 优化配置slave2主机的mysql服务；
* 配置master主机上的master角色，配置主节点；
* 配置两台slave主机的slave角色，配置从节点（在此配置相同，在此只列举slave1的配置）；
* 配置Master角色指定主从复制中特定的数据库及优化；
* 验证Slave角色的数据库复制情况；
* **在master主机上安装ntp时间服务；**

[root@master ~]# yum -y install ntp

[root@master ~]# sed -i '/^server/s/^/#/g' /etc/ntp.conf

[root@master ~]# cat <<END >>/etc/ntp.conf

server 127.127.1.0

fudge 127.127.1.0 stratum 8

END

[root@master ~]# systemctl start ntpd

[root@master ~]# systemctl enable ntpd

Created symlink from /etc/systemd/system/multi-user.target.wants/ntpd.service to /usr/lib/systemd/system/ntpd.service.

* **在两台slave主机中配置同步master的ntp时间（在此配置相同，在此只列举slave1的配置）；**

[root@slave1 ~]# yum -y install ntpdate

[root@slave1 ~]# ntpdate 192.168.100.101

4 Aug 17:27:31 ntpdate[1740]: adjust time server 192.168.100.101 offset 0.000048 sec

[root@slave1 ~]# date

2018年 08月 04日 星期六 17:27:41 CST

[root@slave1 ~]# echo "/usr/sbin/ntpdate 192.168.100.101" >>/etc/rc.local

[root@slave1 ~]# chmod +x /etc/rc.local

* **安装三台Mysql数据库服务器（在此步骤相同，在此只列举master主机一台的安装）；**

[root@master ~]# wget http://dev.mysql.com/Downloads/MySQL-5.7/mysql-5.7.12.tar.gz

[root@master ~]# wget https://sourceforge.net/projects/boost/files/boost/1.59.0/boost\_1\_59\_0.tar.gz/download ##需要使用windows下载，不能使用linux下载

[root@master ~]# yum -y remove mysql-\* boost-\*

[root@master ~]# rpm -qa mysql

[root@master ~]# rpm -qa boost

[root@master ~]# yum -y install gcc gcc-c++ ncurses bison libgcrypt perl cmake

[root@master ~]# mount /dev/cdrom /mnt

[root@master ~]# rpm -ivh /mnt/Packages/ncurses-devel-5.9-13.20130511.el7.x86\_64.rpm --nodeps

[root@master ~]# tar -zxvf boost\_1\_59\_0.tar.gz

[root@master ~]# mv boost\_1\_59\_0 /usr/local/boost/

[root@master ~]# groupadd mysql

[root@master ~]# useradd -r -g mysql mysql

[root@master ~]# tar zxvf mysql-5.7.12.tar.gz -C /usr/src/

[root@master ~]# cd /usr/src/mysql-5.7.12/ ##编译多次时，需要删除CMAKE的缓存，rm -rf /usr/src/mysql-5.7.12/CMakeCache.txt

[root@master mysql-5.7.12]# cmake . -DCMAKE\_INSTALL\_PREFIX=/usr/local/mysql -DMYSQL\_DATADIR=/usr/local/mysql/data \

-DDEFAULT\_CHARSET=utf8 -DDEFAULT\_COLLATION=utf8\_general\_ci \

-DMYSQL\_TCP\_PORT=3306 -DMYSQL\_USER=mysql -DWITH\_MYISAM\_STORAGE\_ENGINE=1 -DWITH\_INNOBASE\_STORAGE\_ENGINE=1 \

-DWITH\_ARCHIVE\_STORAGE\_ENGINE=1 -DWITH\_BLACKHOLE\_STORAGE\_ENGINE=1 \

-DWITH\_MEMORY\_STORAGE\_ENGINE=1 -DENABLE\_DOWNLOADS=1 -DDOWNLOAD\_BOOST=1 -DWITH\_BOOST=/usr/local/boost -DSYSCONFDIR=/etc

注释：

-DCMAKE\_INSTALL\_PREFIX=/usr/local/mysql [MySQL安装的根目录]

-DMYSQL\_DATADIR=/mydata/mysql/data [MySQL数据库文件存放目录]

-DDEFAULT\_CHARSET=utf8 [设置默认字符集为utf8]

-DDEFAULT\_COLLATION=utf8\_general\_ci [设置默认字符校对]

-DMYSQL\_TCP\_PORT=3306 [MySQL的监听端口]

-DMYSQL\_USER=mysql [MySQL用户名]

-DWITH\_MYISAM\_STORAGE\_ENGINE=1 [安装MySQL的myisam数据库引擎]

-DWITH\_INNOBASE\_STORAGE\_ENGINE=1 [安装MySQL的innodb数据库引擎]

-DWITH\_ARCHIVE\_STORAGE\_ENGINE=1 [安装MySQL的archive数据库引擎]

-DWITH\_BLACKHOLE\_STORAGE\_ENGINE=1 [安装MySQL的blackhole数据库引擎]

-DWITH\_MEMORY\_STORAGE\_ENGINE=1 [安装MySQL的memory数据库引擎]

-DENABLE\_DOWNLOADS=1 [编译时允许自主下载相关文件]

-DDOWNLOAD\_BOOST=1 [允许下载BOOST]

-DWITH\_BOOST=/usr/local/boost [指定系统中存在的BOOST]

-DSYSCONFDIR=/etc [MySQL配置文件所在目录]

-DWITH\_READLINE=1 [MySQL的readline library]

-DMYSQL\_UNIX\_ADDR=/var/run/mysql/mysql.sock [MySQL的通讯目录]

-DENABLED\_LOCAL\_INFILE=1 [启用加载本地数据]

-DWITH\_PARTITION\_STORAGE\_ENGINE=1 [启动mysql的分区存储结构]

-DEXTRA\_CHARSETS=all [使MySQL支持所有的扩展字符]

-DWITH\_DEBUG=0 [禁用调试模式]

-DMYSQL\_MAINTAINER\_MODE=0

-DWITH\_SSL:STRING=bundled [通讯时支持ssl协议]

-DWITH\_ZLIB:STRING=bundled [允许使用zlib library]

[root@master ~]# make -j `cat /proc/cpuinfo | grep processor| wc -l`

[root@master ~]# make install

[root@master ~]# ls /usr/local/mysql

bin COPYING data docs include lib logs man mysql-test README share support-files

[root@master ~]# chown -R mysql:mysql /usr/local/mysql

[root@master ~]# cp /usr/src/mysql-5.7.12/support-files/my-default.cnf /etc/my.cnf

[root@master ~]# cp /usr/src/mysql-5.7.12/support-files/mysql.server /etc/init.d/

[root@master ~]# chmod +x /etc/init.d/mysql.server

[root@master ~]# cat <<END >>/usr/lib/systemd/system/mysqld.service

[Unit]

Description=mysqldapi

After=network.target

[Service]

Type=forking

PIDFile=/usr/local/mysql/logs/mysqld.pid

ExecStart=/etc/init.d/mysql.server start

ExecReload=/etc/init.d/mysql.server restart

ExecStop=/etc/init.d/mysql.server stop

PrivateTmp=Flase

[Install]

WantedBy=multi-user.target

END

注解：

PrivateTmp=Flase ##此配置必须关闭，不然mysql连接文件mysql.sock文件会默认生成在以下位置/tmp/systemd-private-83bba738e8ff4837b5ae657eff983821-mysqld.service-BPxWpJ/tmp/mysql.sock，导致数据库无法连接，将此配置项关闭后，则文件正常生成在/tmp/mysql.sock

[root@master ~]# echo "export PATH=$PATH:/usr/local/mysql/bin/" >>/etc/profile

[root@master ~]# source /etc/profile

[root@master ~]# mysqld --initialize-insecure --user=mysql --basedir=/usr/local/mysql --datadir=/usr/local/mysql/data

注解：

--initialize-insecure ##禁用mysql的密码策略（密码复杂性等），--initializeaize是开启密码策略，自动生成密码在mysqld.log文件中

--user=mysql ##运行的账户

--basedir=/usr/local/mysql ##mysql的安装位置

--datadir=/usr/local/mysql/data ##mysql数据库服务数据的物理存放路径

[root@master ~]# cat <<END >/etc/my.cnf

[mysqld]

basedir = /usr/local/mysql

datadir = /usr/local/mysql/data

port = 3306

sql\_mode=NO\_ENGINE\_SUBSTITUTION,STRICT\_TRANS\_TABLES

character\_set\_server=utf8

init\_connect='SET NAMES utf8'

log-error=/usr/local/mysql/logs/mysqld.log

pid-file=/usr/local/mysql/logs/mysqld.pid

skip-name-resolve

END

注解：

[mysqld] ##声明区域

basedir = /usr/local/mysql ##mysql的安装位置

datadir = /usr/local/mysql/data ##mysql的物理文件存放位置

port = 3306 ##mysql服务监听的端口

sql\_mode=NO\_ENGINE\_SUBSTITUTION,STRICT\_TRANS\_TABLES ##mysql的模式

character\_set\_server=utf8 ##字符集

init\_connect='SET NAMES utf8'

log-error=/usr/local/mysql/logs/mysqld.log ##指定日志文件位置

pid-file=/usr/local/mysql/logs/mysqld.pid ##指定运行服务所产生的pid文件位置

skip-name-resolve ##禁用数据库的反向DNS解析

[root@master ~]# mkdir /usr/local/mysql/logs

[root@master ~]# chown mysql:mysql /usr/local/mysql/logs/

[root@master ~]# systemctl start mysqld

[root@master ~]# systemctl enable mysqld

Created symlink from /etc/systemd/system/multi-user.target.wants/mysqld.service to /usr/lib/systemd/system/mysqld.service.

[root@master ~]# netstat -utpln |grep mysqld

tcp 0 0 0.0.0.0:3306 0.0.0.0:\* LISTEN 2885/mysqld

[root@master ~]# mysql ##登录mysql设置密码

mysql> alter user 'root'@'localhost' identified by '123123';

mysql> exit

[root@master ~]# mysql -uroot -p123123

mysql> exit

* **优化配置Master主机的mysql服务；**

[root@master ~]# vi /etc/my.cnf ##末尾添加

log-bin=master-bin

log-slave-updates

server-id=1

innodb\_flush\_log\_at\_trx\_commit=2

master\_info\_repository=table

relay\_log\_info\_repository=TABLE

:wq

注解：

log-bin=master-bin ##指定生成二进制文件，可加目录

log-slave-updates ##指定开启slave角色的更新

server-id=1 ##指定id号码

innodb\_flush\_log\_at\_trx\_commit=2 ##见下方

master\_info\_repository=table

relay\_log\_info\_repository=TABLE

[root@master ~]# systemctl restart mysqld

[root@master ~]# ls /usr/local/mysql/data/

master-bin.000001 master-bin.index ...

* **优化配置slave1主机的mysql服务；**

[root@slave1 ~]# vi /etc/my.cnf ##末尾添加

relay-log=relay1-log-bin

relay-log-index=slave-relay1-bin.index

server-id=2

innodb\_flush\_log\_at\_trx\_commit=2

slave-parallel-type=LOGICAL\_CLOCK

slave\_parallel\_workers=16

master\_info\_repository=table

relay\_log\_info\_repository=TABLE

:wq

注解：

relay-log=relay-log-bin ##中继日志文件名

relay-log-index=slave-relay-bin.index ##中继日志索引文件

server-id=2 ##id号码

innodb\_flush\_log\_at\_trx\_commit=2

##指定数据库在存储数据时，事务的刷新方式，如下：

0：该模式速度最快，但不太安全，mysqld进程的崩溃会导致上一秒钟所有事务数据的丢失。

1：最安全的，默认值，但是最慢是每一次事务提交或事务外的指令都需要把日志写入（flush）硬盘。在mysqld 服务崩溃或者服务器主机crash的情况下，binary log 只有可能丢失最多一个语句或者一个事务。

2：比0安全，速度比较快，是不写入硬盘而是写入系统缓存。日志仍然会每秒flush到硬盘，所以你一般不会丢失超过1-2秒的更新。只有在操作系统崩溃或者系统断电的情况下，上一秒钟所有事务数据才可能丢失。

slave-parallel-type=LOGICAL\_CLOCK

##选择并行复制的类型

slave\_parallel\_workers=16

##worker线程数量

master\_info\_repository=table

##默认是file，master的更新或者slave的更新直接存放在文件内，若出现故障，会出现数据丢失；若改为table，则为表的方式进行存储更新，支持事务，安全性更高；

relay\_log\_info\_repository=TABLE

##指定中继日志更新，提升性能

[root@slave1 ~]# systemctl restart mysqld

[root@slave1 ~]# ls /usr/local/mysql/data/

relay1-log-bin.000001 slave-relay1-bin.index ...

* **优化配置slave2主机的mysql服务；**

[root@slave2 ~]# vi /etc/my.cnf ##末尾添加

relay-log=relay2-log-bin

relay-log-index=slave-relay2-bin.index

server-id=3

innodb\_flush\_log\_at\_trx\_commit=2

slave-parallel-type=LOGICAL\_CLOCK

slave\_parallel\_workers=16

master\_info\_repository=table

relay\_log\_info\_repository=TABLE

:wq

[root@slave2 ~]# systemctl restart mysqld

[root@slave2 ~]# ls /usr/local/mysql/data/

relay2-log-bin.000001 slave-relay2-bin.index ...

* **配置master主机上的master角色，配置主节点；**

[root@master ~]# mysql -uroot -p123123

mysql> show variables like 'server\_id';

+---------------+-------+

| Variable\_name | Value |

+---------------+-------+

| server\_id | 1 |

+---------------+-------+

mysql> grant replication slave on \*.\* to 'linuxfan'@'192.168.100.%' identified by '123123';

Query OK, 0 rows affected, 1 warning (0.00 sec)

mysql> flush privileges;

Query OK, 0 rows affected (0.01 sec)

mysql> show master status;

+-------------------+----------+--------------+------------------+-------------------+

| File | Position | Binlog\_Do\_DB | Binlog\_Ignore\_DB | Executed\_Gtid\_Set |

+-------------------+----------+--------------+------------------+-------------------+

| master-bin.000001 | 154 | | | |

+-------------------+----------+--------------+------------------+-------------------+

mysql> create database linuxfan;

Query OK, 1 row affected (0.01 sec)

mysql> show databases;

+--------------------+

| Database |

+--------------------+

| information\_schema |

| linuxfan |

| mysql |

| performance\_schema |

| sys |

+--------------------+

mysql> exit

* **配置两台slave主机的slave角色，配置从节点（在此配置相同，在此只列举slave1的配置）；**

[root@master ~]# mysql -uroot -p123123

mysql> show variables like 'server\_id';

+---------------+-------+

| Variable\_name | Value |

+---------------+-------+

| server\_id | 2 |

+---------------+-------+

mysql> change master to master\_host='192.168.100.101',master\_user='linuxfan',master\_password='123123',master\_log\_file='master-bin.000001',master\_log\_pos=1; ##指定时slave的状态必须处于stop状态

Query OK, 0 rows affected, 2 warnings (0.01 sec)

mysql> start slave;

Query OK, 0 rows affected (0.01 sec)

mysql> show slave status\G;

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 1. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Slave\_IO\_State: Waiting for master to send event

Master\_Host: 192.168.100.101

Master\_User: linuxfan

Master\_Port: 3306

Connect\_Retry: 60

Master\_Log\_File: master-bin.000001

Read\_Master\_Log\_Pos: 154

Relay\_Log\_File: relay-log-bin.000002

Relay\_Log\_Pos: 321

Relay\_Master\_Log\_File: master-bin.000001

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: 154

Relay\_Log\_Space: 526

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:

Replicate\_Ignore\_Server\_Ids:

Master\_Server\_Id: 1

Master\_UUID: c5e6dabb-9353-11e8-a760-000c294fc900

Master\_Info\_File: /usr/local/mysql/data/master.info

SQL\_Delay: 0

SQL\_Remaining\_Delay: NULL

Slave\_SQL\_Running\_State: Slave has read all relay log; waiting for mo

Master\_Retry\_Count: 86400

Master\_Bind:

Last\_IO\_Error\_Timestamp:

Last\_SQL\_Error\_Timestamp:

Master\_SSL\_Crl:

Master\_SSL\_Crlpath:

Retrieved\_Gtid\_Set:

Executed\_Gtid\_Set:

Auto\_Position: 0

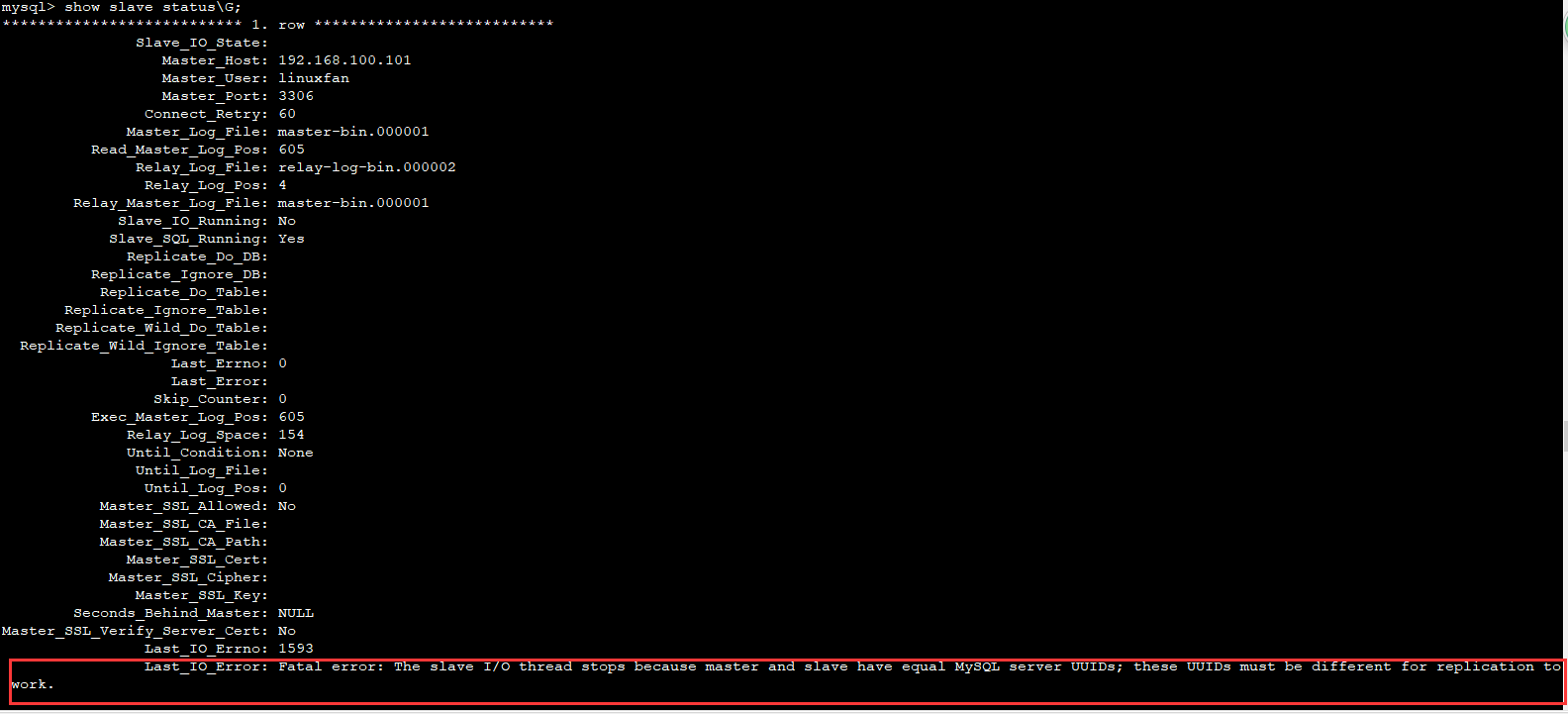
Replicate\_Rewrite\_DB:

Channel\_Name:

Master\_TLS\_Version:

1 row in set (0.00 sec)

**错误1：**slave中出现无法识别Master的UUID；

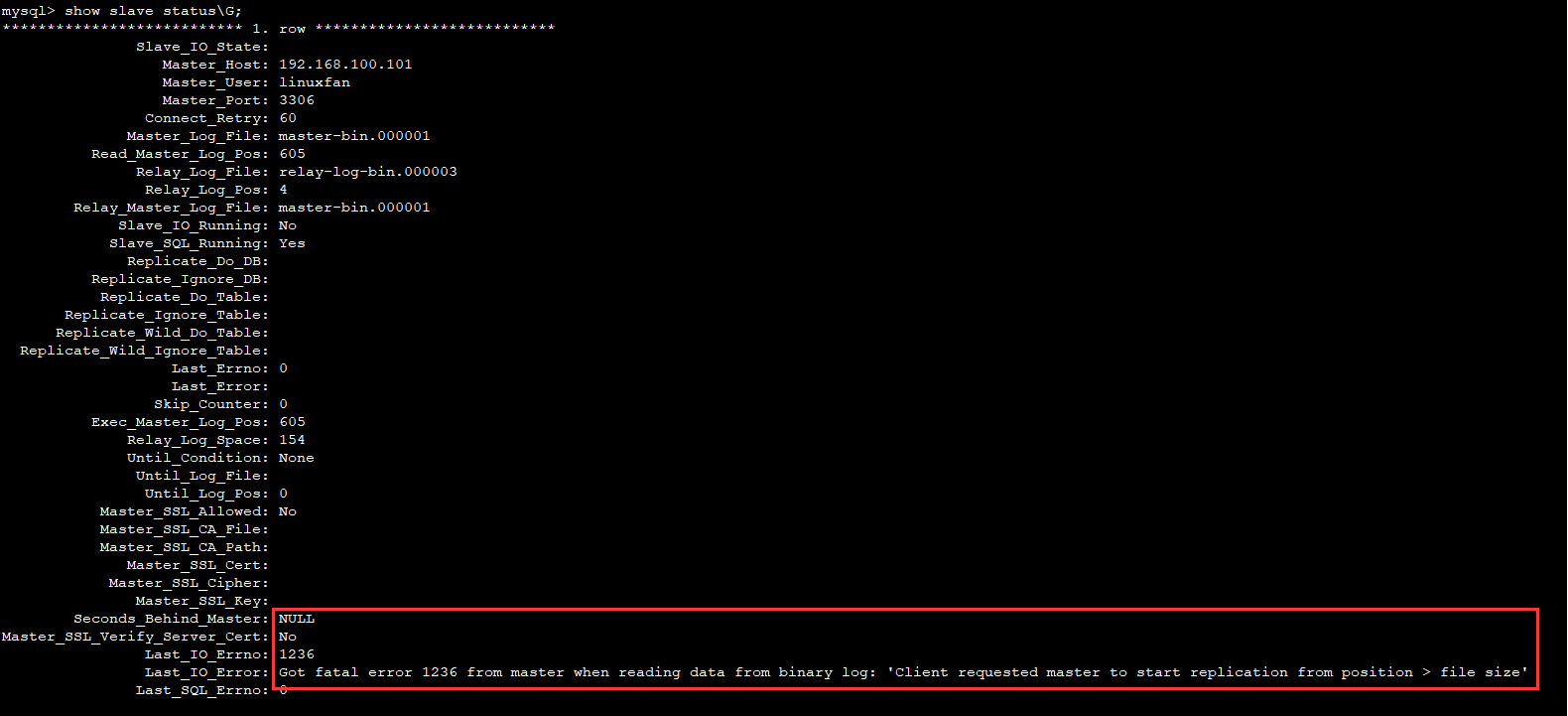


解决方法：

[root@slave1 ~]# rm -rf /usr/local/mysql/data/auto.cnf ##重新生成UUID的配置文件

[root@slave1 ~]# systemctl restart mysqld

**错误2：**slave中指定的master的position值大于当前值；



解决方法：

[root@slave1 ~]# mysql -uroot -p123123 ##重新指定master的position值

mysql> stop slave;

mysql> change master to ...

mysql> start slave;

mysql> show databases;

+--------------------+

| Database |

+--------------------+

| information\_schema |

| linuxfan |

| mysql |

| performance\_schema |

| sys |

+--------------------+

* **配置Master角色指定主从复制中特定的数据库及优化；**

[root@master ~]# vi /etc/my.cnf ##末尾添加

binlog-do-db=cloud1

binlog-do-db=linuxfan

binlog-ignore-db=mysql

binlog-ignore-db=information\_schema

binlog-ignore-db=performance\_schema

binlog-ignore-db=sys

binlog-ignore-db=cloud2

expire\_logs\_days=7

:wq

注解：

binlog-do-db=cloud1 ##允许同步的数据库名称

binlog-do-db=linuxfan

binlog-ignore-db=mysql ##不允许同步的数据库名称

binlog-ignore-db=information\_schema

binlog-ignore-db=performance\_schema

binlog-ignore-db=sys

binlog-ignore-db=cloud2

expire\_logs\_days=7 ##指定日志存放的时间为7天

[root@master ~]# systemctl restart mysqld

[root@master ~]# mysql -uroot -p123123

mysql> show databases;

+--------------------+

| Database |

+--------------------+

| information\_schema |

| linuxfan |

| mysql |

| performance\_schema |

| sys |

+--------------------+

mysql> use linuxfan;

Database changed

mysql> create table it(岗位 char(16),姓名 char(16),身份证号 char(48),学历 char(16),工资 int);

Query OK, 0 rows affected (0.02 sec)

mysql> insert into it values('网络工程师','张三','150404199201091132','高 中','5500');

Query OK, 1 row affected (0.02 sec)

mysql> select \* from it;

+-----------------+--------+--------------------+--------+--------+

| 岗位 | 姓名 | 身份证号 | 学历 | 工资 |

+-----------------+--------+--------------------+--------+--------+

| 网络工程师 | 张三 | 150404199201091132 | 高中 | 5500 |

+-----------------+--------+--------------------+--------+--------+

mysql> create database cloud1;

Query OK, 1 row affected (0.00 sec)

mysql> create database cloud2;

Query OK, 1 row affected (0.00 sec)

mysql> show databases;

+--------------------+

| Database |

+--------------------+

| information\_schema |

| cloud1 |

| cloud2 |

| linuxfan |

| mysql |

| performance\_schema |

| sys |

+--------------------+

mysql> exit

* **验证Slave角色的数据库复制情况；**

[root@slave1 ~]# mysql -uroot -p123123

mysql> show databases;

+--------------------+

| Database |

+--------------------+

| information\_schema |

| cloud1 |

| linuxfan |

| mysql |

| performance\_schema |

| sys |

+--------------------+

mysql> use linuxfan;

Database changed

mysql> select \* from it;

+-----------------+--------+--------------------+--------+--------+

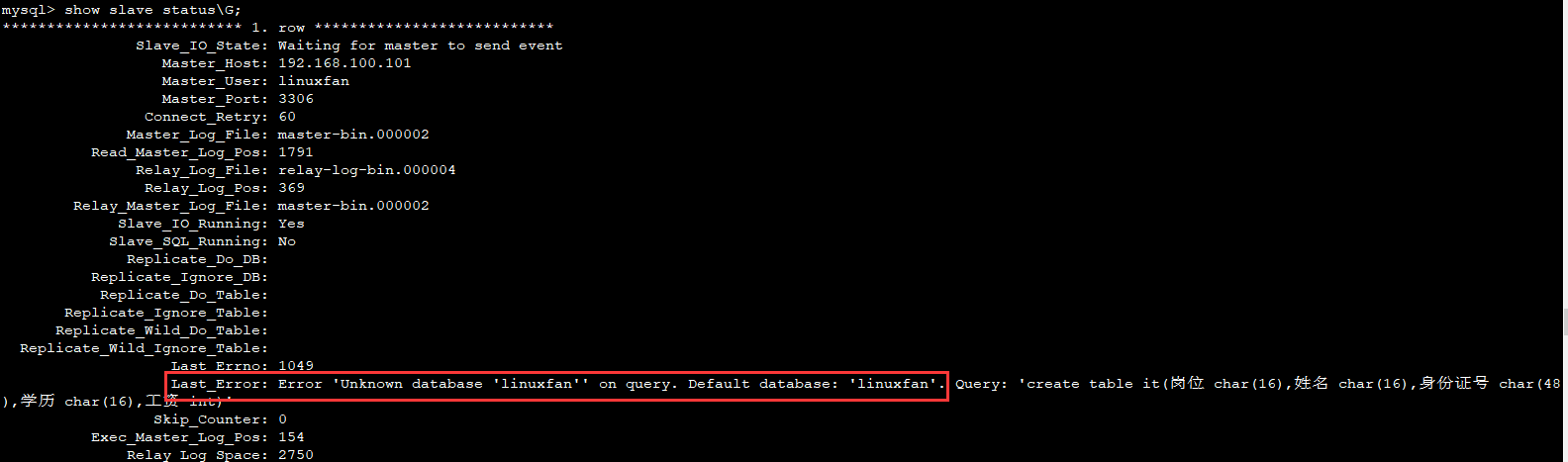
| 岗位 | 姓名 | 身份证号 | 学历 | 工资 |

+-----------------+--------+--------------------+--------+--------+

| 网络工程师 | 张三 | 150404199201091132 | 高中 | 5500 |

+-----------------+--------+--------------------+--------+--------+

**错误：**slave中无法找到当前insert语句所在的数据库，错误原因是在之前的同步中出现了失败，导致后续同步无法继续运行；



解决方案：在slave中创建缺少的数据库或者其他数据；

**六、案例：搭建Mysql数据库集群（主从复制--GTID方式）；**

**案例环境：**同上

**案例步骤：**

* 在master主机上安装ntp时间服务；
* 在两台slave主机中配置同步master的ntp时间（在此配置相同，在此只列举slave1的配置）；
* 安装三台Mysql数据库服务器（在此步骤相同，在此只列举master主机一台的安装）；
* 优化配置Master主机的mysql服务；
* 优化配置slave1主机的mysql服务；
* 优化配置slave2主机的mysql服务；
* 配置master主机上的master角色，配置主节点；
* 配置两台slave主机的slave角色，配置从节点（在此配置相同，在此只列举slave1的配置）；
* 验证主从复制；
* **在master主机上安装ntp时间服务；**

同上

* **在两台slave主机中配置同步master的ntp时间（在此配置相同，在此只列举slave1的配置）；**

同上

* **安装三台Mysql数据库服务器（在此步骤相同，在此只列举master主机一台的安装）；**

同上

* **优化配置Master主机的mysql服务；**

[root@master ~]# vi /etc/my.cnf ##末尾添加

log-bin=master-bin

log-slave-updates

server-id=1

innodb\_flush\_log\_at\_trx\_commit=2

master\_info\_repository=table

relay\_log\_info\_repository=TABLE

binlog-format=ROW

gtid-mode=ON

enforce-gtid-consistency=true

binlog\_cache\_size = 4M

max\_binlog\_size = 1G

max\_binlog\_cache\_size = 2G

skip-name-resolve

:wq

注解：

binlog-format=ROW ##二进制文件的格式

gtid-mode=ON ##开启GTID模式

enforce-gtid-consistency=true ##开启GTID的一些安全限制

binlog\_cache\_size = 4M ##单个事务的二进制日志占用内存的最大值

max\_binlog\_size = 1G ##二进制文件最大的大小

max\_binlog\_cache\_size = 2G ##二进制文件占用内存的最大值

[root@master ~]# systemctl restart mysqld

[root@master ~]# ls /usr/local/mysql/data/

master-bin.000001 master-bin.index ...

* **优化配置slave1主机的mysql服务；**

[root@slave1 ~]# vi /etc/my.cnf ##末尾添加

relay-log=relay1-log-bin

relay-log-index=slave-relay1-bin.index

server-id=2

innodb\_flush\_log\_at\_trx\_commit=2

slave-parallel-type=LOGICAL\_CLOCK

slave\_parallel\_workers=16

master\_info\_repository=table

relay\_log\_info\_repository=TABLE

binlog-format=ROW

binlog-row-image = minimal

log-bin=slave1-bin

log-bin-index=slave1-log-bin.index

gtid-mode=ON

enforce-gtid-consistency=true

binlog\_cache\_size = 4M

max\_binlog\_size = 1G

max\_binlog\_cache\_size = 2G

slave-sql-verify-checksum=1

binlog-rows-query-log\_events=1

log-slave-updates

relay\_log\_purge = 1

relay\_log\_recovery = 1

skip-name-resolve

:wq

注解：

binlog-format=ROW ##二进制文件的格式

binlog-row-image = minimal ##使mysql复制中的行体积更小

log-bin=slave1-bin ##二进制日志文件的名称

log-bin-index=slave1-log-bin.index ##二进制日志文件的索引文件

gtid-mode=ON ##开启GTID

enforce-gtid-consistency=true ##开启GTID的一些安全限制

binlog\_cache\_size = 4M ##单个事务的二进制日志占用内存的最大值

max\_binlog\_size = 1G ##二进制日志文件的最大大小

max\_binlog\_cache\_size = 2G ##二进制日志文件占用内存的最大值

slave-sql-verify-checksum=1 ##判断从库SQL Threa线程从relay log中读出的事件是否正常,默认设置为1

binlog-rows-query-log\_events=1 ##优化主从性能

log-slave-updates ##开启slave角色的日志更新

relay\_log\_purge = 1 ##不需要中继日志时自动清空

relay\_log\_recovery = 1 ##当slave从库宕机后，假如relay-log损坏了，导致一部分中继日志没有处理，则自动放弃所有未执行的relay-log，并且重新从master上获取日志，这样就保证了relay-log的完整性。

[root@slave1 ~]# systemctl restart mysqld

[root@slave1 ~]# ls /usr/local/mysql/data/

slave1-log-bin.index slave1-bin.000001 ...

* **优化配置slave2主机的mysql服务；**

[root@slave2 ~]# vi /etc/my.cnf ##末尾添加

relay-log=relay2-log-bin

relay-log-index=slave-relay2-bin.index

server-id=3

innodb\_flush\_log\_at\_trx\_commit=2

slave-parallel-type=LOGICAL\_CLOCK

slave\_parallel\_workers=16

master\_info\_repository=table

relay\_log\_info\_repository=TABLE

binlog-format=ROW

binlog-row-image = minimal

log-bin=slave2-bin

log-bin-index=slave2-log-bin.index

gtid-mode=ON

enforce-gtid-consistency=true

binlog\_cache\_size = 4M

max\_binlog\_size = 1G

max\_binlog\_cache\_size = 2G

slave-sql-verify-checksum=1

binlog-rows-query-log\_events=1

log-slave-updates

relay\_log\_purge = 1

relay\_log\_recovery = 1

skip-name-resolve

:wq

[root@slave2 ~]# systemctl restart mysqld

[root@slave2 ~]# ls /usr/local/mysql/data/

slave2-log-bin.index slave2-bin.000001 ...

* **配置master主机上的master角色，配置主节点；**

[root@master ~]# mysql -uroot -p123123

mysql> show variables like "%gtid%"; ##查看GTID方式是否开启

+----------------------------------+-----------+

| Variable\_name | Value |

+----------------------------------+-----------+

| binlog\_gtid\_simple\_recovery | ON |

| enforce\_gtid\_consistency | ON |

| gtid\_executed\_compression\_period | 1000 |

| gtid\_mode | ON |

| gtid\_next | AUTOMATIC |

| gtid\_owned | |

| gtid\_purged | |

| session\_track\_gtids | OFF |

+----------------------------------+-----------+

mysql> grant replication slave on \*.\* to 'linuxfan'@'192.168.100.%' identified by '123123';

Query OK, 0 rows affected, 1 warning (0.00 sec)

mysql> flush privileges;

Query OK, 0 rows affected (0.01 sec)

mysql> exit

[root@master ~]# systemctl restart mysqld

* **配置两台slave主机的slave角色，配置从节点（在此配置相同，在此只列举slave1的配置）；**

[root@slave1 ~]# mysql -uroot -p123123

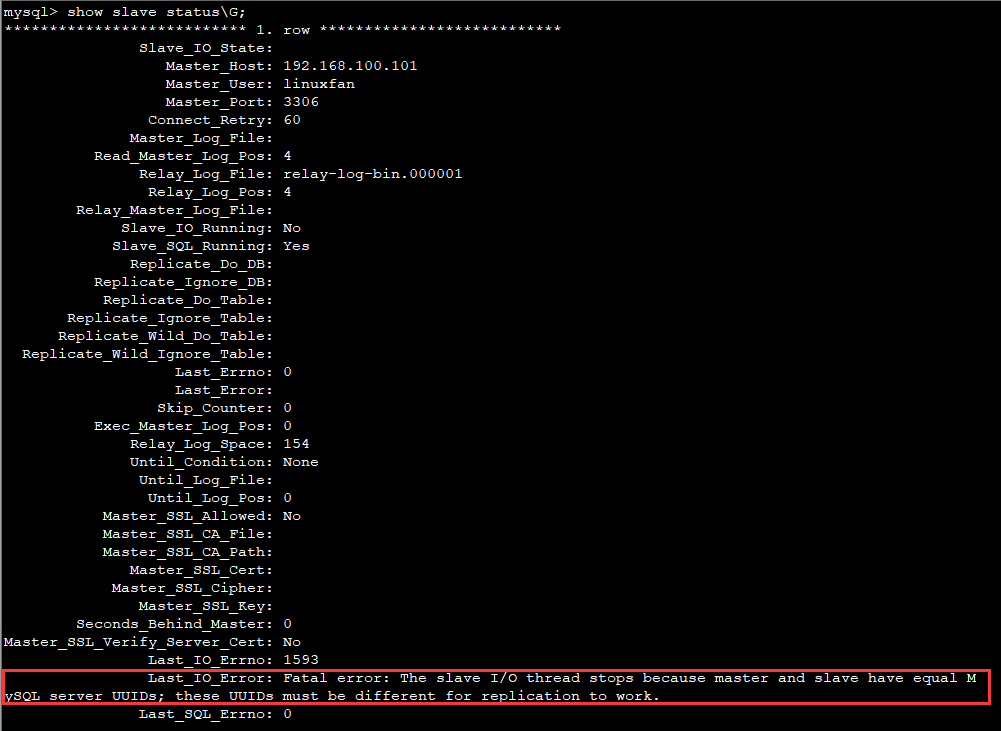
mysql> change master to master\_host='192.168.100.101',master\_user='linuxfan',master\_password='123123',master\_port=3306,master\_auto\_position=1,master\_delay=5; ##自动扫描pos值，同步延迟5秒

Query OK, 0 rows affected, 2 warnings (0.40 sec)

mysql> start slave;

Query OK, 0 rows affected (0.10 sec)

mysql> show slave status\G;

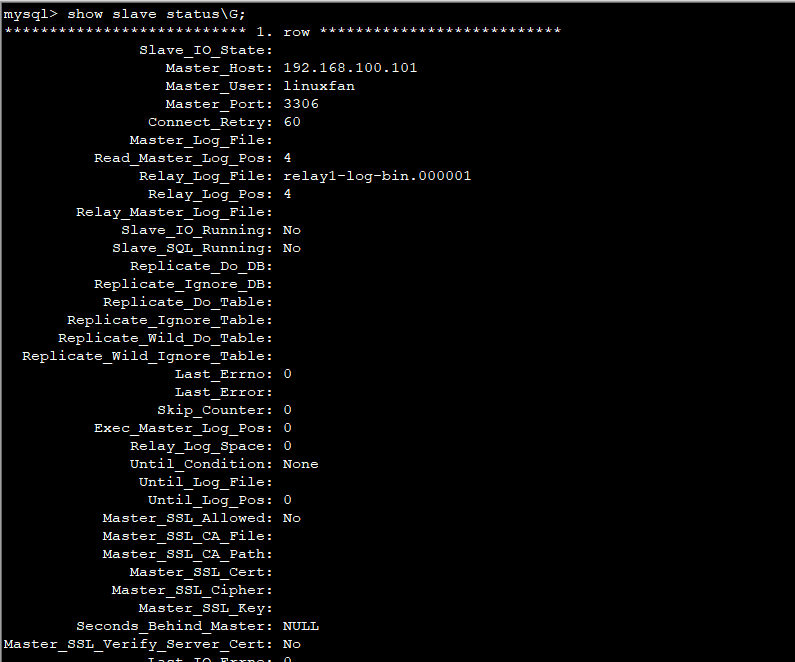


mysql> exit

[root@slave1 ~]# rm -rf /usr/local/mysql/data/auto.cnf

[root@slave1 ~]# systemctl restart mysqld

[root@slave1 ~]# mysql -uroot -p123123



mysql> reset slave;

Query OK, 0 rows affected (0.00 sec)

mysql> start slave;

Query OK, 0 rows affected (0.07 sec)

mysql> show slave status\G;

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 1. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Slave\_IO\_State: Waiting for master to send event

Master\_Host: 192.168.100.101

Master\_User: linuxfan

Master\_Port: 3306

Connect\_Retry: 60

Master\_Log\_File: master-bin.000003

Read\_Master\_Log\_Pos: 353

Relay\_Log\_File: relay-log-bin.000004

Relay\_Log\_Pos: 568

Relay\_Master\_Log\_File: master-bin.000003

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: 353

Relay\_Log\_Space: 1028

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:

Replicate\_Ignore\_Server\_Ids:

Master\_Server\_Id: 1

Master\_UUID: c5e6dabb-9353-11e8-a760-000c294fc900

Master\_Info\_File: mysql.slave\_master\_info

SQL\_Delay: 5

SQL\_Remaining\_Delay: NULL

Slave\_SQL\_Running\_State: Slave has read all relay log; waiting for more updates

Master\_Retry\_Count: 86400

Master\_Bind:

Last\_IO\_Error\_Timestamp:

Last\_SQL\_Error\_Timestamp:

Master\_SSL\_Crl:

Master\_SSL\_Crlpath:

Retrieved\_Gtid\_Set: c5e6dabb-9353-11e8-a760-000c294fc900:1-3

Executed\_Gtid\_Set: c5e6dabb-9353-11e8-a760-000c294fc900:1-3

Auto\_Position: 1

Replicate\_Rewrite\_DB:

Channel\_Name:

Master\_TLS\_Version:

1 row in set (0.00 sec)

mysql> exit

* **验证主从复制；**

[root@master ~]# mysql -uroot -p123123

mysql> create database cloud;

Query OK, 1 row affected (0.00 sec)

mysql> show databases;

+--------------------+

| Database |

+--------------------+

| information\_schema |

| cloud |

| mysql |

| performance\_schema |

| sys |

+--------------------+

mysql> exit

[root@slave1 ~]# mysql -uroot -p123123

mysql> show databases;

+--------------------+

| Database |

+--------------------+

| information\_schema |

| cloud |

| mysql |

| performance\_schema |

| sys |

+--------------------+

mysql> exit

**七、案例：实现Mysql主从复制+读写分离；**

**案例环境：**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 系统类型 | IP地址 | 主机名 | 所需软件 | 硬件 |
| Centos 7.4 1708 64bit | 192.168.100.101 | master.linuxfan.cn | boost\_1\_59\_0.tar.gz  mysql-5.7.12.tar.gz | 内存：4G  CPU核心：2 |
| Centos 7.4 1708 64bit | 192.168.100.102 | slave1.linuxfan.cn | boost\_1\_59\_0.tar.gz  mysql-5.7.12.tar.gz | 内存：4G  CPU核心：2 |
| Centos 7.4 1708 64bit | 192.168.100.103 | slave2.linuxfan.cn | boost\_1\_59\_0.tar.gz  mysql-5.7.12.tar.gz | 内存：4G  CPU核心：2 |
| Centos 7.4 1708 64bit | 192.168.100.104 | am.linuxfan.cn | jdk-6u14-linux-x64.bin  amoeba-mysql-binary-2.2.0.tar.gz | 内存：512M  CPU核心：1 |
| Centos 7.4 1708 64bit | 192.168.100.105 | lamp.linuxfan.cn |  | 内存：512M  CPU核心：1 |

**案例步骤：**

* 搭建并测试一主两从的主从复制环境；
* 安装am节点的amoeba代理程序；
* 在master节点进行授权允许amoeba程序能够连接Mysql集群，slave节点会同步权限；
* 配置am节点的代理服务并启动服务；
* 客户端测试；
* **搭建并测试一主两从的主从复制环境；**
* **安装am节点的amoeba代理程序；**

[root@am ~]# yum -y remove java

[root@am ~]# ls

jdk-6u14-linux-x64.bin amoeba-mysql-binary-2.2.0.tar.gz ...

[root@am ~]# chmod +x jdk-6u14-linux-x64.bin

[root@am ~]# ./jdk-6u14-linux-x64.bin

[root@am ~]# mv jdk1.6.0\_14/ /usr/local/jdk1.6

[root@am ~]# vi /etc/profile

export JAVA\_HOME=/usr/local/jdk1.6

export CLASSPATH=$CLASSPATH:$JAVA\_HOME/lib:$JAVA\_HOME/jre/lib

export PATH=$JAVA\_HOME/lib:$JAVA\_HOME/jre/bin/:$PATH:$HOME/bin

export AMOEBA\_HOME=/usr/local/amoeba

export PATH=$PATH:$AMOEBA\_HOME

:wq

[root@am ~]# source /etc/profile

[root@am ~]# java -version

java version "1.6.0\_14"

Java(TM) SE Runtime Environment (build 1.6.0\_14-b08)

Java HotSpot(TM) 64-Bit Server VM (build 14.0-b16, mixed mode)

[root@am ~]# mkdir /usr/local/amoeba

[root@am ~]# tar zxvf amoeba-mysql-binary-2.2.0.tar.gz -C /usr/local/amoeba/

[root@am ~]# ls /usr/local/amoeba/

benchmark bin changelogs.txt conf lib LICENSE.txt README.html

[root@am ~]# chmod -R 755 /usr/local/amoeba/

* **在master节点进行授权允许amoeba程序能够连接Mysql集群，slave节点会同步权限；**

**[root@master ~]# mysql -uroot -p123123**

mysql> grant all on \*.\* to 'amoebauser'@'192.168.100.104' identified by '123123';

Query OK, 0 rows affected, 1 warning (0.00 sec)

mysql> flush privileges;

Query OK, 0 rows affected (0.00 sec)

mysql> exit

**[root@slave1 ~]# mysql -uroot -p123123**

**mysql> show grants for 'amoebauser'@'192.168.100.104';**

**+---------------------------------------------------------------+**

**| Grants for amoebauser@192.168.100.104 |**

**+---------------------------------------------------------------+**

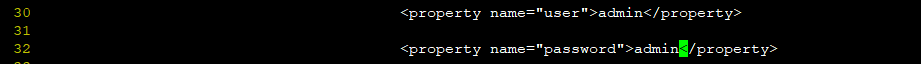
**| GRANT ALL PRIVILEGES ON \*.\* TO 'amoebauser'@'192.168.100.104' |**

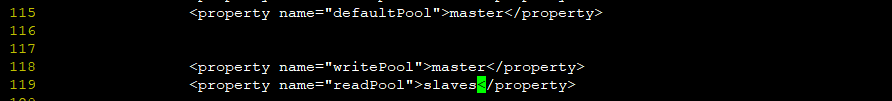
**+---------------------------------------------------------------+**

**mysql> exit**

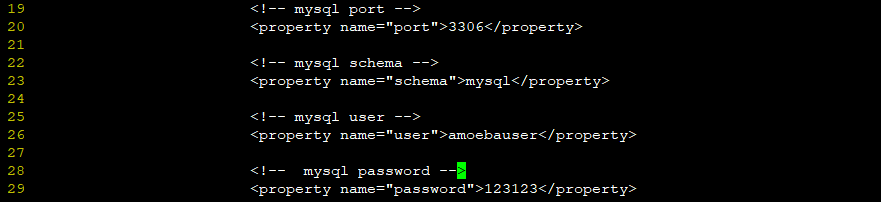
* **配置am节点的代理服务并启动服务；**

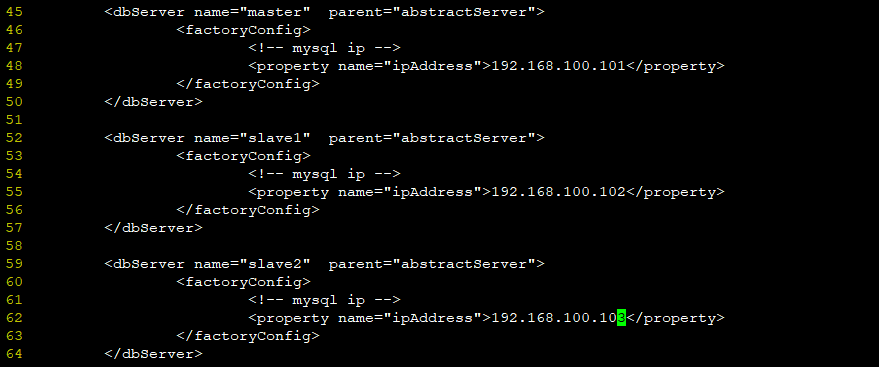
[root@am ~]# vim /usr/local/amoeba/conf/amoeba.xml

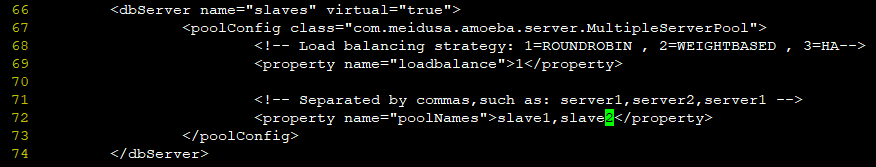




[root@am ~]# vim /usr/local/amoeba/conf/dbServers.xml







[root@am ~]# /usr/local/amoeba/bin/amoeba start & ##后台启动代理服务

[1] 1273

[root@am ~]# log4j:WARN log4j config load completed from file:/usr/local/amoeba/conf/log4j.xml

2018-08-06 00:32:32,191 INFO context.MysqlRuntimeContext - Amoeba for Mysql current versoin=5.1.45-mysql-amoeba-proxy-2.2.0

log4j:WARN ip access config load completed from file:/usr/local/amoeba/conf/access\_list.conf

2018-08-06 00:32:32,381 INFO net.ServerableConnectionManager - Amoeba for Mysql listening on 0.0.0.0/0.0.0.0:8066.

2018-08-06 00:32:32,383 INFO net.ServerableConnectionManager - Amoeba Monitor Server listening on /127.0.0.1:35233

[root@am ~]# netstat -utpln |grep 8066

tcp 0 0 0.0.0.0:8066 0.0.0.0:\* LISTEN 1273/java

* **客户端测试；**

[root@lamp ~]# yum -y install mysql

[root@lamp ~]# mysql -uadmin -padmin -h192.168.100.104 -P 8066

MySQL [(none)]> show databases;

+--------------------+

| Database |

+--------------------+

| information\_schema |

| cloud |

| mysql |

| performance\_schema |

| sys |

+--------------------+

MySQL [(none)]> create database linuxfan;

Query OK, 1 row affected (0.01 sec)

MySQL [(none)]> create table linuxfan.table(id int(10),name varchar(10),address varchar(20));

MySQL [(none)]> insert into linuxfan.table values(1,'hehe','this is master');

MySQL [(none)]> insert into linuxfan.table values(2,'hehe','this is slave1');

MySQL [(none)]> insert into linuxfan.table values(3,'hehe','this is slave2');

MySQL [(none)]> exit

分别在master、slave1、slave2中验证数据库linuxfan；