第1章 主从复制介绍

1.介绍

```
MySQL数据库的主从复制技术与使用scp/rsync等命令进行的异机文件级别复制类似，都是数据的远程传输.
只不过MySQL的主从复制技术是其软件自身携带的功能，无须借助第三方工具.
MySQL的主从复制并不是直接复制数据库磁盘上的文件，而是将逻辑的记录数据库更新的binlog日志发送到需要同步的数据库服务器本地，然后再由本地的数据库线程读取日志中的SQL语句并重新应用到MySQL数据库中，从而即可实现数据库的主从复制。
```

2.应用场景

```
1.从服务器作为主服务器的实时数据备份
2.主从服务器实现读写分离，从服务器实现负载均衡
3.根据业务重要性对多个服务器进行拆分
```

第2章 主从复制搭建部署

1.安装部署mysql实例

```
安装过程略，此处只给出3台实例的配置文件
# db-01配置
cat > /etc/my.cnf<<EOF
[mysqld]
user=mysql
datadir=/data/mysql_3306
basedir=/opt/mysql/
socket=/tmp/mysql.sock
port=3306
log_error=/var/log/mysql/mysql.err
server_id=51
log_bin=/binlog/mysql-bin

[mysql]
socket=/tmp/mysql.sock

[client]
socket=/tmp/mysql.sock
EOF

# db-52配置
cat > /etc/my.cnf<<EOF
[mysqld]
user=mysql
datadir=/data/mysql_3306
basedir=/opt/mysql/
socket=/tmp/mysql.sock
port=3306
log_error=/var/log/mysql/mysql.err
server_id=52
```

```

[mysql]
socket=/tmp/mysql.sock

[client]
socket=/tmp/mysql.sock
EOF

# db-53配置
cat > /etc/my.cnf<<EOF
[mysqld]
user=mysql
datadir=/data/mysql_3306
basedir=/opt/mysql/
socket=/tmp/mysql.sock
port=3306
log_error=/var/log/mysql/mysql.err
server_id=53

[mysql]
socket=/tmp/mysql.sock

[client]
socket=/tmp/mysql.sock
EOF

# 初始化
mysqld --initialize-insecure --user=mysql --basedir=/opt/mysql --
datadir=/data/mysql_3306/
```

2.主库操作

2.1 创建复制用户

```
mysql -uroot -p123456 -e "grant replication slave on *.* to repl@'10.0.0.%'
identified by '123';"
mysql -uroot -p123456 -e "select user,host,plugin from mysql.user;"
```

2.2 备份数据并发送到从库

```
mysqldump -uroot -p123456 -A --master-data=2 --single-transaction -R -E --
triggers --max_allowed_packet=64M > /data/full.sql
scp /data/full.sql 10.0.0.52:/tmp/
scp /data/full.sql 10.0.0.53:/tmp/
```

3.从库操作

3.1 查看从库位置点

```
-- CHANGE MASTER TO MASTER_LOG_FILE='mysql-bin.000001', MASTER_LOG_POS=444;
```

3.2 导入主库数据

```
mysql -uroot -p123456 < /tmp/full.sql
```

3.3 配置主从同步信息

```
CHANGE MASTER TO
MASTER_HOST='10.0.0.51',
MASTER_USER='repl',
MASTER_PASSWORD='123',
MASTER_PORT=3306,
MASTER_LOG_FILE='mysql-bin.000001',
MASTER_LOG_POS=444,
MASTER_CONNECT_RETRY=10;
```

3.4 启动线程

```
start slave;
```

4.查看复制状态

```
mysql -uroot -p123456 -e "show slave status\G"|grep "Running:"
```

第3章 主从复制原理

1.涉及到的线程

1.1 主库

线程说明：

```
binlog_dump_thread
负责接收slave请求和传送主库binlog给slave
```

查看命令：

```
mysql> show processlist;
+----+------+-----------------+------+-------------+------+-------------------------------------------------------+------------------+
| Id | User | Host            | db   | Command     | Time | State                                                 | Info             |
+----+------+-----------------+------+-------------+------+-------------------------------------------------------+------------------+
| 32 | repl | 10.0.0.53:47726 | NULL | Binlog Dump |  284 | Master has sent all binlog to slave; waiting for more updates | NULL             |
| 33 | repl | 10.0.0.52:55250 | NULL | Binlog Dump |  280 | Master has sent all binlog to slave; waiting for more updates | NULL             |
| 34 | root | localhost       | NULL | Query       |    0 | starting                                              | show processlist |
+----+------+-----------------+------+-------------+------+-------------------------------------------------------+------------------+
```

1.2 从库

线程说明:

```
IO线程 :
连接主库DUMP线程,请求Master日志、接收Master日志、存储日志（relay-log）。

SQL线程
回放relaylog
```

查看命令:

```
[root@db-52 ~]# mysql -uroot -p123456 -e "show slave status\G"|grep "Running:"
            Slave_IO_Running: Yes
            Slave_SQL_Running: Yes
```

2.涉及到的文件

2.1 主库

```
binlog日志文件
```

2.2 从库

relay-log 中继日志

```
命名方式:
datadir/HOSTNAME-relay-bin.00000N

作用:
存储获取到的binlog
```

主库信息文件

```
命名方式:
datadir/master.info
作用:
记录主库ip  port  user  password  binlog位置点等信息。
```

中继日志应用信息

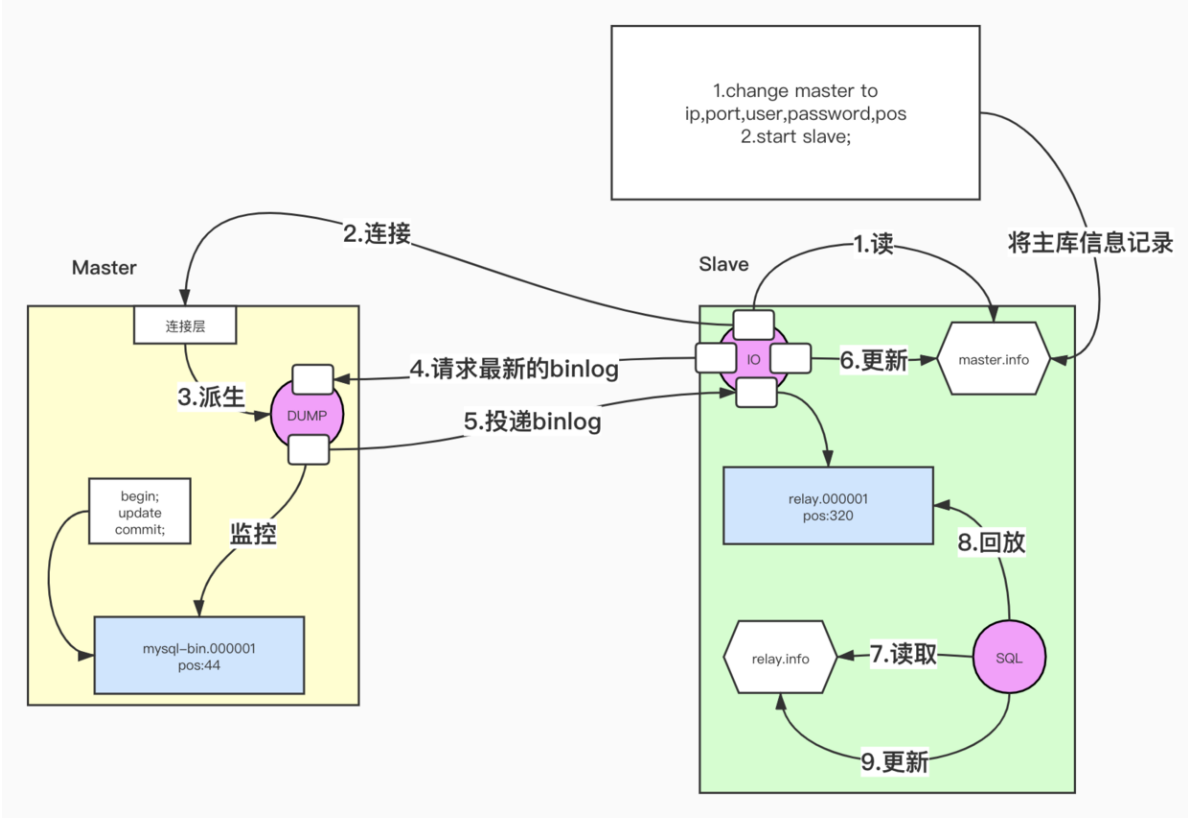
```
命名方式:
relay-log.info

作用:
记录SQL 线程回放到的位置点信息。
```

3.画图说明主从复制原理--必须掌握--面试必问!

文字:

```
从库： Change master to IP，Port，USER，PASSWORD，binlog位置信息写入到M.info中,执行Start slave(启动SQL，IO)。
从库： 连接主库。
主库： 分配Dump_T,专门和S_IO通信。show processlist;
从库： IO线程：IO线程请求新日志
主库： DUMP_T 接收请求,截取日志，返回给S_IO
从库： IO线程接收到binlog,此时网络层层面返回ACK给主库。主库工作完成。
从库： IO将binlog最终写入到relaylog中，并更新M.info。IO线程工作结束。
从库： SQL线程读R.info,获取上次执行到的位置点
从库： SQL线程向后执行新的relay-log，再次更新R.info。
```

画图：



第4章 主从复制监控

1.主库状态

查看复制线程：

```
[root@db-51 ~]# mysql -uroot -p123456 -e "show processlist" |grep "Dump"
mysql: [Warning] Using a password on the command line interface can be insecure.
32      repl    10.0.0.53:47726 NULL    Binlog Dump     568     Master has sent all binlog to slave; waiting for more updates   NULL
33      repl    10.0.0.52:55250 NULL    Binlog Dump     564     Master has sent all binlog to slave; waiting for more updates   NULL
```

查看复制节点信息：

```
[root@db-51 ~]# mysql -uroot -p123456 -e "show slave hosts;"

+-----------+------+------+-----------+--------------------------------------+
| Server_id | Host | Port | Master_id | Slave_UUID                           |
+-----------+------+------+-----------+--------------------------------------+
|        53 |      | 3306 |        51 | 0f61194b-f733-11ea-8ca6-000c29c20a5d |
|        52 |      | 3306 |        51 | 2fbe6b47-f731-11ea-9d25-000c29605eb5 |
+-----------+------+------+-----------+--------------------------------------+
```

2.从库状态

查看主从状态：

```
mysql -uroot -p123456  -e "show slave status \G"
```

主库连接信息、binlog位置信息

```
Master_Host: 10.0.0.51
Master_User: repl
Master_Port: 3306
Connect_Retry: 10
Read_Master_Log_Pos: 444
Relay_Master_Log_File: mysql-bin.000001
```

从库中relay-log的回放信息

```
Relay_Log_File: db-52-relay-bin.000002
Relay_Log_Pos: 320
Relay_Master_Log_File: mysql-bin.000001
Exec_Master_Log_Pos: 444
```

线程监控信息：主要用来排查主从故障-重点监控

```
Slave_IO_Running: Yes
Slave_SQL_Running: Yes
Last_IO_Errno: 0
Last_IO_Error:
Last_SQL_Errno: 0
Last_SQL_Error:
```

过滤复制相关信息

```
Replicate_Do_DB:
Replicate_Ignore_DB:
Replicate_Do_Table:
Replicate_Ignore_Table:
Replicate_Wild_Do_Table:
Replicate_Wild_Ignore_Table:
```

落后于主库的秒数-重点监控

```
Seconds_Behind_Master: 0
```

延时从库状态信息

```
SQL_Delay: 0
SQL_Remaining_Delay: NULL
```

GTID复制信息

```
Retrieved_Gtid_Set:    上次接收到的事务号
Executed_Gtid_Set:     已经执行过的事务号
Auto_Position: 0
```

3.位置点信息

IO 已经获取到的主库Binlog的位置点

```
Master_Log_File: mysql-bin.000001
Read_Master_Log_Pos: 444
作用： IO下次请求日志时，起点位置。
```

SQL 回放到的relaylog位置点

```
Relay_Log_File: db01-relay-bin.000006
Relay_Log_Pos: 320
```

SQL回放的realylog位置点，对应的主库binlog的位置点

```
Relay_Master_Log_File: mysql-bin.000001
Exec_Master_Log_Pos: 600
作用： 计算主从复制延时日志量。
```

第5章 主从复制故障

1.如何监控

```
Slave_IO_Running: Yes         # IO线程工作状态： YES、NO、Connecting
Slave_SQL_Running: Yes        # SQL线程工作状态：YES、NO
Last_IO_Errno: 0              # IO故障代码：2003,1045,1040,1593,1236
Last_IO_Error:                # IO线程报错详细信息
Last_SQL_Errno: 0             # SQL故障代码：  1008，1007
Last_SQL_Error:               # IO线程报错详细信息
```

2.IO线程故障

2.1 正常状态

```
Slave_IO_Running: Yes
```

2.2 不正常状态

```
NO
Connecting
```

2.3 故障原因

```
1.网络，端口，防火墙
2.用户 ，密码，授权
  replication slave
3.主库连接数上限
  mysql> select @@max_connections;
4.版本不统一  5.7 native  ，  8.0 sha2
```

2.4 模拟故障

主库操作

```
mysql> start slave; # 启动所有线程
mysql> stop slave;  # 关闭所有线程
mysql> start slave sql_thread; #单独启动SQL线程
mysql> start slave io_thread; #单独启动IO线程
mysql> stop  slave sql_thread;
mysql> stop  slave io_thread;

解除从库身份：
mysql> reset slave all;
mysql> show slave status \G
```

从库操作

```
stop slave;
reset slave all;

CHANGE MASTER TO
MASTER_HOST='10.0.0.51',
MASTER_USER='repl',
MASTER_PASSWORD='123',
MASTER_PORT=3307,
MASTER_LOG_FILE='mysql-bin.000003',
MASTER_LOG_POS=154,
MASTER_CONNECT_RETRY=10;
start slave;
```

2.5 解决思路

```
1.网络是否互通
2.确定复制账号授权是否正确
3.主从的server_id是否相同
4.主从的server_uuid是否相同
```

3.SQL线程故障

3.1 SQL线程主要工作

```
回放relay-log中的日志。可以理解为执行relay-log SQL
```

3.2 故障本质

```
为什么SQL线程执行不了SQL语句
```

3.3 故障原因

```
创建的对象已经存在
需要操作的对象不存在
约束冲突。
以上问题： 大几率出现在从库写入或者双主结构中容易出现。
```

3.4 故障模拟

```
(1)先在从库 create database oldguo charset=utf8;
(2)在主库  create database oldguo charset=utf8mb4;
(3)检查从库SQL线程状态
Slave_SQL_Running: No
Last_Error: Error 'Can't create database 'oldguo'; database exists' on query.
Default database: 'oldguo'. Query: 'create database oldguo'
```

3.5 故障处理

思路1: 一切以主库为准

```
在从库上进行反操作一下。重启线程
mysql> drop database oldguo;
mysql> start slave;
```

思路2: 以从库为准，跳过此次复制错误，不建议

```
binlog跳过:
stop slave;
set global sql_slave_skip_counter = 1;

GTID跳过:
stop slave;
SET @@SESSION.GTID_NEXT= '8f9e146f-0a18-11e7-810a-0050568833c8:4'

#将同步指针向下移动一个，如果多次不同步，可以重复操作。
start slave;
```

思路3: 暴力方法，遇到自动跳过，不建议。

```
/etc/my.cnf
slave-skip-errors = 1032,1062,1007

常见错误代码:
1007:对象已存在
1032:无法执行DML
1062:主键冲突,或约束冲突
```

思路4: 重新搭建主从

```
备份恢复 + 重新构建
```

第7章 过滤复制

1.过滤复制介绍

```
从节点仅仅复制指定的数据库,或指定数据库的指定数据表
```

2.主库实现

```
binlog_do_db       白名单
binlog_ignore_db   黑名单
通过是否记录binlog日志来控制过滤
```

3.从库实现

实现方法:

```
IO线程不做限制。
SQL线程回放时，选择性回放。
```

配置参数:

```
replicate_do_db=world
replicate_do_db=oldboy
replicate_ignore_db=

replicate_do_table=world.city
replicate_ignore_table=

replicate_wild_do_table=world.t*
replicate_wild_ignore_table=
```

配置方法1: 修改配置文件

```
replicate_do_db=world
replicate_do_db=oldboy
```

配置方法2: 在线热配置

```
STOP SLAVE SQL_THREAD;
CHANGE REPLICATION FILTER REPLICATE_DO_DB = (oldguo, oldboy);
START  SLAVE SQL_THREAD;
```

第8章 延时从库的应用

1.延时从库介绍

```
控制从库的SQL线程执行速度，二进制日志照常去主库取，但是存放到中继日志之后就延迟执行。
如果主库被误操作，这时候对中继日志进行处理，就不用根据全备二进制日志恢复，节省了大部分的时间
```

2.配置方法

```
stop slave;
CHANGE MASTER TO MASTER_DELAY = 300;
start slave;
```

3.查看状态

```
mysql> show slave status \G
SQL_Delay: 300
SQL_Remaining_Delay: NULL
```

3.故障处理流程

```
1. 及时监控故障： 主库 10:05发现故障，从库此时8:05数据状态
2. 立即将从库的SQL线程关闭。 需要对A业务挂维护页。
3. 停止所有线程。
4. 在延时从。恢复A库数据
   手工模拟SQL线程工作，找到drop之前位置点。
   SQL线程上次执行到的位置 ----> drop之前
   relay.info ----> 分析drop位置点 ----> 截取relaylog日志 ----> source
```

4.故障模拟及恢复

主库操作:

```
create database zhangya charset utf8mb4;
use zhangya;
create table t1(id int);
insert into t1 values(1),(2),(3);
commit;

drop database zhangya;
```

从库操作:

```
stop slave sql_thread;
show slave status \G;
```

截取日志：

起点：SQL上次执行到的位置点，

```
Relay_Log_File: db-52-relay-bin.000002
Relay_Log_Pos: 320
```

终点：drop 之前

```
mysql> show relaylog events in 'db-52-relay-bin.000002';
....略
| db-52-relay-bin.000002 | 985 | Query          |        51 |        1201 |
drop database json
```

截取日志：

```
mysqlbinlog --start-position=320 --stop-position=985 /data/mysql_3306/db-52-
relay-bin.000002 >/tmp/bin.sql
```

从库恢复操作：

```
stop slave;
reset slave all;
set sql_log_bin=0;
source /tmp/bin.sql;
set sql_log_bin=1;
```

第9章 GTID复制

1.GITD复制介绍

```
功能：主从之间自动校验GTID一致性：主库binlog，从库binlog ，relay-log

没有备份：
自动从主库的第一个gtid对应的pos号开始复制

有备份：
SET @@GLOBAL.GTID_PURGED='2386f449-98a0-11ea-993c-000c298e182d:1-10';
从库会自动从第11个gtid开始复制。
```

2.清理环境

```
pkill mysqld
rm -rf /data/mysql_3306/*
rm -rf /binlog/*
mkdir /binlog/
```

3.准备配置文件

db01配置

```
cat > /etc/my.cnf <<EOF
```

```
[mysqld]
user=mysql
datadir=/data/mysql_3306
basedir=/opt/mysql/
socket=/tmp/mysql.sock
port=3306
log_error=/var/log/mysql/mysql.err
server_id=51
log_bin=/binlog/mysql-bin
autocommit=0
binlog_format=row
gtid-mode=on
enforce-gtid-consistency=true
log-slave-updates=1

[mysql]
socket=/tmp/mysql.sock

[client]
socket=/tmp/mysql.sock
EOF
```

db02配置

```
cat > /etc/my.cnf <<EOF
[mysqld]
user=mysql
datadir=/data/mysql_3306
basedir=/opt/mysql/
socket=/tmp/mysql.sock
port=3306
log_error=/var/log/mysql/mysql.err
server_id=52
autocommit=0
gtid-mode=on
enforce-gtid-consistency=true
log-slave-updates=1

[mysql]
socket=/tmp/mysql.sock

[client]
socket=/tmp/mysql.sock
EOF
```

4.初始化数据

```
mysqld --initialize-insecure --user=mysql --basedir=/opt/mysql --
datadir=/data/mysql_3306/
```

5.启动数据库

```
/etc/init.d/mysqld start
```

6.创建用户

```
grant replication slave on *.* to repl@'10.0.0.%' identified by '123';
```

7.构建主从

52和53操作:

```
change master to
master_host='10.0.0.51',
master_user='repl',
master_password='123' ,
MASTER_AUTO_POSITION=1;
start slave;
```

第10章 主从延时问题的原因和处理

1.什么是主从延时

```
主库发生了操作，从库'很久'才跟上来，甚至一直追不上
```

2.如何监控主从延时

粗略估计:

```
show slave status \G
Seconds_Behind_Master: 0
```

准确计算:

```
日志量:
主库binlog位置点
从relay执行的位置点
```

3.如何计算延时的日志量

```
show master status;
cat /data/3308/data/relay-log.info
```

4.主从延时的原因

4.1 主库可能的原因

```
外部原因:
网络,硬件配置,主库业务繁忙,从库太多

主库业务繁忙 :
1. 拆分业务（分布式）： 组件分离 ，垂直 ， 水平
2. 大事务的拆分 。比如，1000w 业务 拆分为 20次执行。

```

```
内部 ：
1. 二进制日志更新问题：
解决方案：
sync_binlog=1

2. 问题： 5.7之前的版本，没有开GTID之前，主库可以并发事务，但是dump传输时是串行。
所以会导致，事务量，大事务时会出现比较严重延时。
解决方案：
5.6+ 版本，手工开启gtid，事务在主从的全局范围内就有了唯一性标志。
5.7+ 版本，无需手工开启，系统会自动生成匿名的GTID信息
有了GTID之后，就可以实现并发传输binlog。
但是,即使有这么多的优秀特性，我们依然需要尽可能的减少大事务，以及锁影响。
```

4.2 从库可能的原因

```
外部 ：
网络,从库配置低,参数设定。

内部 ：
IO线程：
写relay-log --> IO 性能。

SQL线程：
回放 SQL 默认在非GTID模式下是串行的

解决方案：
1. 开启GTID
```