28.MySQL主从复制–主从监控，主从故障，主从延时的分析处理√

1. 主从复制监控方式

1.1 针对主库监控

Bash | Copy

```
1.查看binlog日志文件，pos位置点，GTID。与从库对比计算数据差异
db01 [(none)]>show master status;
+------------------+----------+--------------+------------------+-------------------------------------------+
| File             | Position | Binlog_Do_DB | Binlog_Ignore_DB | Executed_Gtid_Set                         |
+------------------+----------+--------------+------------------+-------------------------------------------+
| mysql-bin.000004 |      341 |              |                  | 389ea959-b194-11eb-a24b-000c29edc386:1 |
+------------------+----------+--------------+------------------+-------------------------------------------+
主库执行到的日志文件   主库执行到的位置点

2.查看主库线程（长连接）
非交互方式：
[root@db01 ~]# mysql  -e "show processlist" |grep "Dump"
30  repl  10.0.0.52:8936  NULL  Binlog Dump GTID  6127  Master has sent all binlog to slave; waiting for more
31  repl  10.0.0.53:14267 NULL  Binlog Dump GTID  6088  Master has sent all binlog to slave; waiting for more

3.查看从库节点信息
db01 [(none)]>show slave hosts;
+-----------+-----------+------+-----------+--------------------------------------+
| Server_id | Host      | Port | Master_id | Slave_UUID                           |
+-----------+-----------+------+-----------+--------------------------------------+
|        53 | 10.0.0.53 | 3306 |        51 | 3eb0186d-b194-11eb-ae04-000c29b4ef39 |
|        52 | 10.0.0.52 | 3306 |        51 | 3bd8fc03-b194-11eb-9fc2-000c294178ee |
+-----------+-----------+------+-----------+--------------------------------------+
显示host列必须在从库的配置文件上添加参数
vim /etc/my.cnf
[mysqld]
report_host=10.0.0.52(从库ip地址)
report_port=3306
```

1.2 针对从库监控

Bash | Copy

```
db02 [(none)]>show slave status\G;
*************************** 1. row ***************************
               Slave_IO_State: Waiting for master to send event
                  Master_Host: 10.0.0.51
                  Master_User: repl
                  Master_Port: 3306
                Connect_Retry: 10
              Master_Log_File: mysql-bin.000004
          Read_Master_Log_Pos: 341
               Relay_Log_File: db02-relay-bin.000004
                Relay_Log_Pos: 458
        Relay_Master_Log_File: mysql-bin.000004
             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: 341
              Relay_Log_Space: 706
              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: 51
                  Master_UUID: 389ea959-b194-11eb-a24b-000c29edc386
             Master_Info_File: mysql.slave_master_info
                    SQL_Delay: 0
          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: 389ea959-b194-11eb-a24b-000c29edc386:1
             Executed_Gtid_Set: 389ea959-b194-11eb-a24b-000c29edc386:1
                 Auto_Position: 1
          Replicate_Rewrite_DB:
                  Channel_Name:
            Master_TLS_Version:
        Master_public_key_path:
         Get_master_public_key: 0
             Network_Namespace:

我们对查看的结果进行分类分析
a. 主库连接信息、binlog位置信息 。来自于mysql.slave_master_info表
Master_Host: 10.0.0.51
Master_User: repl
Master_Port: 3306
Connect_Retry: 10
Master_Log_File: mysql-bin.000004   从库获取到主库的日志名称（slave_io）
Read_Master_Log_Pos: 341            从库获取到主库的位置点(slave_io)

b.从库中relay-log的回放信息 。来自于mysql.slave_relay_log_info表
Relay_Log_File: db02-relay-bin.000004
Relay_Log_Pos: 458
Relay_Master_Log_File: mysql-bin.000004  从库执行到的日志名称（slave_sql）
Exec_Master_Log_Pos: 341                 从库执行到的位置点（slave_sql）


扩展：计算主从日志差异的日志量（字节）,可以准确的预估主从延时
首先我们要知道主库对接的是binlog日志，从库对接的是relaylog日志。分别都有属于自己的日志编号和位置点。
Relay_Master_Log_File，Exec_Master_Log_Pos两个参数记录从库relaylog日志对应的主库binlog日志对应关系
我们要计算主从日志差异的日志量需要用到的信息就是 主库执行到的位置点和从库执行到的位置点进行做差。
获取主库执行到的位置点
mysql> show master status下的
postion
获取从库执行到的位置点
mysql> show slave status下的
Exec_Master_Log_Pos

c.线程监控信息：主要用来排查主从故障
Slave_IO_Running: Yes
Slave_SQL_Running: Yes
Last_IO_Errno: 0
Last_IO_Error:
Last_SQL_Errno: 0
Last_SQL_Error:

d.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:

e.主从延时的时间（ 落后于主库的秒数）
Seconds_Behind_Master: 0
是一个监控指标，但是并不严谨，只能用来判断主从延时结果有没有

f.延时从库（人为设定的，作用于逻辑损坏的处理）
SQL_Delay: 0
SQL_Remaining_Delay: NULL

g.gtid复制信息
Retrieved_Gtid_Set: 389ea959-b194-11eb-a24b-000c29edc386:1
Executed_Gtid_Set: 389ea959-b194-11eb-a24b-000c29edc386:1
Auto_Position: 1
```

2.主从故障分析及处理

主从故障常说的是从库线程发生的故障

2.1 如何监控

```
mysql> show slave status \G;
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：1008，1007，1032，1062..
Last_SQL_Error:                      # IO线程报错详细信息
```

2.2 slave_io线程故障

slave_io负责的工作是：连接主库，请求主库，接受日志，存储日志。

slave_io的故障有两种状态 no和connecting

==connecting（连接故障）分析==

外部因素

```
1.网络不同
2.防火墙拦截
```

内部因素

```
1.数据库宕机
2.连接信息错误（用户，密码，ip,port,加密插件）
3.最大连接数上限
```

内部因素故障模拟

1.主库宕机

```
故障模拟
1.主库关闭数据库
db01 [(none)]> shutdown;
2.重启从库slave线程
db02 [(none)]>stop slave ;
db02 [(none)]>start slave;
3.查看slave报错
db02 [(none)]>show slave status ;
Slave_IO_Running: Connecting    ---io线程报连接错误
Last_IO_Errno: 2003             ---io线程报错代码
Last_IO_Error: error connecting to master 'repl@10.0.0.51:3306' - retry-time: 10 ---io线程报错信息

还原
1.开启数据库
/etc/init.d/mysqld start
2.手动启动slave查看
start slave;
db02 [(none)]>show slave status ;
Slave_IO_Running: yes
```

2.用户密码错误

```
1.模拟用户密码错误，更改repl主从用户密码
db01 [(none)]> alter user repl@'10.0.0.%' identified with mysql_native_password by '1';
2.重启从库slave线程
db02 [(none)]>stop slave ;
db02 [(none)]>start slave;
3.查看slave报错
db02 [(none)]>show slave status ;
Slave_IO_Running: Connecting
Last_IO_Errno: 1045
Last_IO_Error: error connecting to master 'repl@10.0.0.51:3306' - retry-time: 10 retries: 5 message:Access der

还原：
将repl密码修改正确
db01 [(none)]> alter user repl@'10.0.0.%' identified with mysql_native_password by '123';
```

3.连接数上限

```
首先连接它是那个参数
最大并发连接数
db01 [(none)]>select @@max_connections;
+-------------------+
| @@max_connections |
+-------------------+
|               151 | 最多152会预留给本地管理员一个
+-------------------+
1.模拟更改最大连接数
db01 [(none)]>set global max_connections=1;
2.重启从库slave线程
db02 [(none)]>stop slave ;
db02 [(none)]>start slave;
3.查看slave报错
db02 [(none)]>show slave status ;
Slave_IO_Running: Connecting
Last_IO_Errno: 1040
Last_IO_Error: error connecting to master 'repl@10.0.0.51:3306' - retry-time: 10 retries: 1 message: Too many

还原
set global max_connections=151;
```

==no（主从交互信息故障）分析==

1.server_id 或者server_uuid重复

```
1.模拟从库server_id与主库server_id相同
db02 [(none)]>set global server_id=51;
2.重启从库slave线程
db02 [(none)]>stop slave ;
db02 [(none)]>start slave;
3.查看slave报错
db02 [(none)]>show slave status ;
Slave_IO_Running: No
Last_IO_Errno: 13117
Last_IO_Error: Fatal error: The slave I/O thread stops because master and slave have equal MySQL server ids; t

还原
db02 [(none)]>set global server_id=52;
```

2.请求日志失败

①搭建环境情景：设置主库信息错误

Bash | Copy

```
故障模拟
1.从库解除主从状态
db02 [(none)]>stop slave;
db02 [(none)]>reset slave all;
db02 [(none)]>show slave status\G;
2.模拟设置主库信息时设置错误信息
查看一下主库执行的日志是5
db01 [(none)]>show master status;
+------------------+----------+--------------+------------------+-------------------------------------------+
| File             | Position | Binlog_Do_DB | Binlog_Ignore_DB | Executed_Gtid_Set                         |
+------------------+----------+--------------+------------------+-------------------------------------------+
| mysql-bin.000005 |      774 |              |                  | 389ea959-b194-11eb-a24b-000c29edc386:1-3 |
+------------------+----------+--------------+------------------+-------------------------------------------+
3.在从库设置错误主库信息
db02 [(none)]> CHANGE MASTER TO
  MASTER_HOST='10.0.0.51',
  MASTER_USER='repl',
  MASTER_PASSWORD='123',
  MASTER_PORT=3306,
  MASTER_LOG_FILE='mysql-bin.000012', ---错误信息
  MASTER_LOG_POS=1187,
  MASTER_CONNECT_RETRY=10;
4.开启从库线程
db02 [(none)]>start slave;
5.检查线程状态
db02 [(none)]>show slave status\G;
Slave_IO_Running: No
Last_IO_Errno: 13114
Last_IO_Error: Got fatal error 1236 from master when reading data from binary log: 'Could not find first log f

还原
重新设置主库信息
1.从库解除主从状态
db02 [(none)]>stop slave;
db02 [(none)]>reset slave all;
db02 [(none)]>show slave status\G;
2.为从库设置正确的主库信息
db02 [(none)]> 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=1187,
  MASTER_CONNECT_RETRY=10;
3.开启线程
db02 [(none)]>start slave;
```

②生产环境情景：主库日志损坏或缺失

恢复方案：

```
故障模拟
1.模拟主库日志被误删除
db01 [(none)]>reset master;
这条命令使主库所有记录从1号日志文件开始记录
2.查看从库线程状态
db02 [(none)]>show slave status\G;
Slave_IO_Running: No
Last_IO_Errno: 13114
Last_IO_Error: Got fatal error 1236 from master when reading data from binary log: 'could not find next log; t
the last event read from '/data/3306/binlog/mysql-bin.000005' at 956, the last byte read from '/data/3306/bin

故障恢复分为两种情况
如果清空主库日志后没有做操作，则重新设置主库信息，从头开始记录
1.db01 [(none)]>db01 [(none)]>show master status;
+------------------+----------+--------------+------------------+-------------------------------------------+
| File             | Position | Binlog_Do_DB | Binlog_Ignore_DB | Executed_Gtid_Set                         |
+------------------+----------+--------------+------------------+-------------------------------------------+
| mysql-bin.000001 |      341 |              |                  | 389ea959-b194-11eb-a24b-000c29edc386:1 |
+------------------+----------+--------------+------------------+-------------------------------------------+

2.db02 [(none)]> 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=341,
  MASTER_CONNECT_RETRY=10;

 3.db02 [(none)]>start slave;

如果清空主库日志后做了操作，则重新备份主库数据到从库，重新搭建主从架构。
```

通用的排错方法就是使用repl专用主从用户进行手动连接主库，查看报错信息。

```
mysql -urepl -p123 -h10.0.0.51 -p3306
```

2.3 slave_sql线程故障

slave_sql负责的工作是：回放relay log中的sql语句,可以理解为后台执行sql语句

slave_sql的故障只有一种状态 no

1007:对象已存在

1032:无法执行DML

1062:主键冲突,或约束冲突

1.relaylog日志损坏（不常见）

```
处理方法：  重构。
方法1：  备份主库+change master to + start slave;

方法2：  找到问题点+ change master + start slave;
  思路：  如何找到问题位置点。
    1. 找到SQL已经回放到什么位置了。
    SQL回放的realylog位置点，对应的主库binlog的位置点(relay-log.info)
    Relay_Log_File: db01-relay-bin.000006
    Relay_Log_Pos: 320
    ----》
    2. 找到主库相应位置点：
    Relay_Master_Log_File: mysql-bin.000001
    Exec_Master_Log_Pos: 600
    3.  change master to  mysql-bin.000001 600
```

2.回放relaylog时出现问题（关注的点）

```
出现故障的原因
1.主从节点配置不一样：  平台、版本、参数、SQL_MODE
2.从库被提前写入了
   2.1 修改的对象不存在（库、表、用户）
   2.2 创建的对象已存在（库、表、用户、约束冲突）

从库被提前写入的原因
1.人为手动误写入
2.异步主从，主库中间过程宕机，导致数据不一致
3.双主结构(双写入，相互同步数据) 书写一致的数据。 所以我们解决方案就是让双主架构单写入
4.高可用架构的脑裂
```

2.1 故障重现

```
手动模拟从库被误写入（创建的对象已存在）
1.连错节点，连接登陆到主库进行了误操作，创建了一个数据库error.
db02 [(none)]>create database error;
2.又登陆到主库创建了相同的数据库error
db01 [(none)]>create database error;
3.在主库进行了对数据库error的操作
db01 [(none)]>use error
db01 [error]>create table t1(id int);
4.查看从库时，发现数据库error下没有创建的表
db02 [(none)]>use error;
Database changed
db02 [error]>show tables;
Empty set (0.00 sec)
5.查看从库线程发现报错
db02 [error]>show slave status\G;
Slave_SQL_Running: No
Last_Errno: 1007
Last_Error: Error 'Can't create database 'error'; database exists' on query. Default database: 'error'. Query
```

2.2 解决方案

一.解决从库被写入方法

Bash | Copy

```
一.解决从库被写入方法
##针对上面故障重现的问题，官方给出了两种解决方案，或者使用工具pt-table-checksum  pt-table-sync校验主从数据，并同步。

###第一种：一切以主库为准
1.先对比主从库差异数据，将从库多余数据进行删除
1.1 修复操作不记录日志
mysql> set sql_log_bin=0;
1.2 删除从库多余数据
mysql> drop database error;
1.3 恢复日志记录
mysql> set sql_log_bin=1;
2.重启从库线程
mysql> stop slave;start slave;
3.查看线程
db02 [(none)]>show slave status\G;
Slave_SQL_Running: Yes


###第二种：一切以从库为准
从库已经有主库操作的数据库那么就不需要回放relaylog,让主从跳过错误
没开GTID模式下跳过操作
从库操作
mysql>stop slave;
mysql> set global sql_slave_skip_counnter=1;
mysql> start slave;
开GTID模式下跳过操作
从库操作
1.停止线程
mysql> stop slave;
2.找GITD的错误断点
查看线程状态
db02 [cry]>show slave status\G;
Retrieved_Gtid_Set: b02c2553-b3c0-11eb-b01f-000c29edc386:1-9 收到了gtid号码为9
Executed_Gtid_Set: b02c2553-b3c0-11eb-b01f-000c29edc386:1-7, 但是执行的gtid号到7 ，，所以断点是8，9
b1edf251-b3c0-11eb-b054-000c294178ee:1-2
3.将断点gtid号8和9注入空事务跳过
set gtid_next='b02c2553-b3c0-11eb-b01f-000c29edc386:8';
begin;commit;
set gtid_next='b02c2553-b3c0-11eb-b01f-000c29edc386:9';
begin;commit;
4.设置GTID继续按顺序记录
set gtid_next='automatic';
5.重启线程,再查看
mysql> stop slave;start slave
db02 [cry]>show slave status\G;
Slave_SQL_Running: Yes

###其他方法
1.设置从库只读
mysql> set global read_only=1;
```

```
mysql> set global super_read_only=1;
2.或者使用工具pt-table-checksum  pt-table-sync校验主从数据，并同步。
3.重新构建主从

```

二.解决主从数据不一致方法

Bash | Copy

```
二.解决主从数据不一致方法
因为数据不一致的产生根本原因是架构是异步的，所以最好的解决办法是换一种架构模式
1.换主从架构模式
2.mysql5.7版本以后，使用增强半同步（一致性强）
3.MGR(组复制）（一致性较强）
4.第三方PXC  （一致性很强）
```

2.3 sql线程避免故障标准方法

Bash | Copy

```
1. 从库只读 ，读写分离中间件。
2. 不使用双主结构。PXC、MGR替代。
3. 半同步、增强半同步复制等，或者PXC、MGR替代。
4. 使用pt相关工具校验主从数据，并同步。
```

3.主从延时分析及处理

3.1 简介

> 主库的操作，从库延迟很长时间才操作。

3.2 针对主从延时的监控方法

Bash | Copy

```
方法一：
mysql> show slave status\G;下的Seconds_Behind_Master: 0参数
这个参数不准确。因为它是基于日志的时间戳计算出的延时时间，无法判断日志接受后是否被执行。
我们用这个参数仅可以来判断主从之间是否延时。

方法二：
通过日志量判断主从是否延时，预值是10kb
(主库show master status下的position)减去（从库 show slave status下的exec_master_log_pos参数值）=日志量差异

方法三：
可以使用全局唯一的GTID号码，GTID号码是逻辑层面的比较方便，但是无法精准分析差异大小

```

3.3 主从延时原因

外部因素

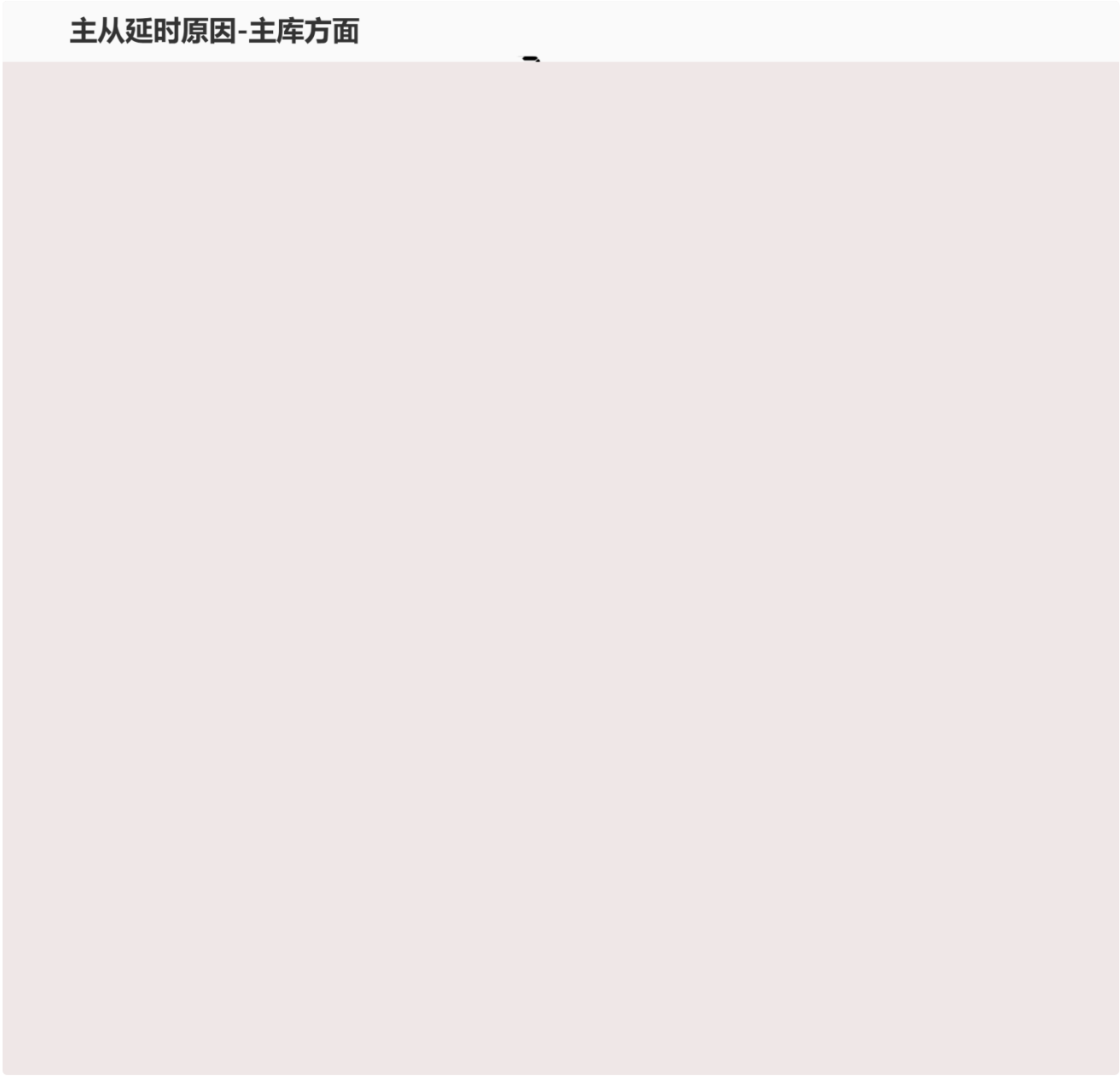
```
1.网络速度慢
2.主从节点硬件配置
3.主从节点设置参数差异性
4.主从节点版本
主从复制可以跨版本搭建，根据高版本向下兼容性原则 主库版本只能低于从库版本。
```

内部因素

主库

```
主库方面延时原因
dump 默认是串行工作的.如果主库并发事务量高,或者大事务时.传输时就会有较高延时.

解决方案: 5.6+版本,加入了Group Commit(GC)技术.两个指标时间延时+个数控制进行组提交.但是依然怕大事务.
binlog_group_commit_sync_delay=1
binlog_group_commit_sync_no_delay_count=1000
```



从库

Bash | Copy

```
从库方面延时原因:
SQL默认是串行工作的. 主库的并发事务量大或者大事务.都会导致 SQL线程回放慢.
解决方案:
5.6版本: 加入了SQL线程并发回放机制. 以database级别进行并发回放.
意思是,只要主库中的事务是来自于不同库的操作,可以并发回放.
slave_parallel_type = DATABASE
slave_parallel_workers = 0 (默认没有开启是0, 最大1024个)

5.7+版本: 加入了Logical_clock模式,使得在主库能够group commit的事务(last_committed=8),并且根据sequence_number=9在从库并
查看binlog日志中
mysqlbinlog /data/3306/binlog/mysql-bin.000001
last_committed号码相同的表示是在同一时间点内并发提交
sequence_number号码是从库的回放顺序

db01 [(none)]>show variables like '%slave%';
slave_parallel_type = DATABASE / logical_clock
slave_parallel_workers = )

8.0+  writesets 写集合方式. MGR.
slave_parallel_type = DATABASE / logical_clock / writeset
slave_parallel_workers = 0
```

3.4 总结

Bash | Copy

```
1. 历史遗留的延时问题,在版本升级过程中基本解决了.
2. 所以主从延时,我们面临的问题就是优化业务. 所以减少大事务,锁问题,性能较差SQL才是优化主从延时的重点.
```

%E4%B8%BB%E4%BB%8E%E7%9B%91%E6%8E%A7%EF%BC%8C%E4%B8%BB%E4%BB%8E%E6%95%85%E9%9A%9C%EF%BC%8C%E4%B8%BB%E4%BB%8E%E5%BB%B6%E6%97%B6%E7%9A%84%E