华北ORACLE用户组

CNNOUG

MySQL主从同步技术基础

杜诚文

CONTENTS

01

主从复制

主从复制

MySQL 复制

MySQL 中的复制功能用于将更改从主服务器复制到一个或多个从属服务器。主服务器将更改写入二进制日志，从属服务器请求主服务器的二进制文件并应用其内容。日志文件的格式影响从属服务器应用更改的方式。MySQL 支持基于语句的、基于行的以及混合格式的日志记录。

从属服务器数量

一个主服务器可以具有的从属服务器数量没有限制。但是，每个额外从属服务器使用主服务器上的较少资源，给定环境中主服务器的最佳从属服务器数量取决于许多因素：模式大小、写入次数、主服务器和从属服务器的相对性能以及 CPU 和内存可用性等因素。

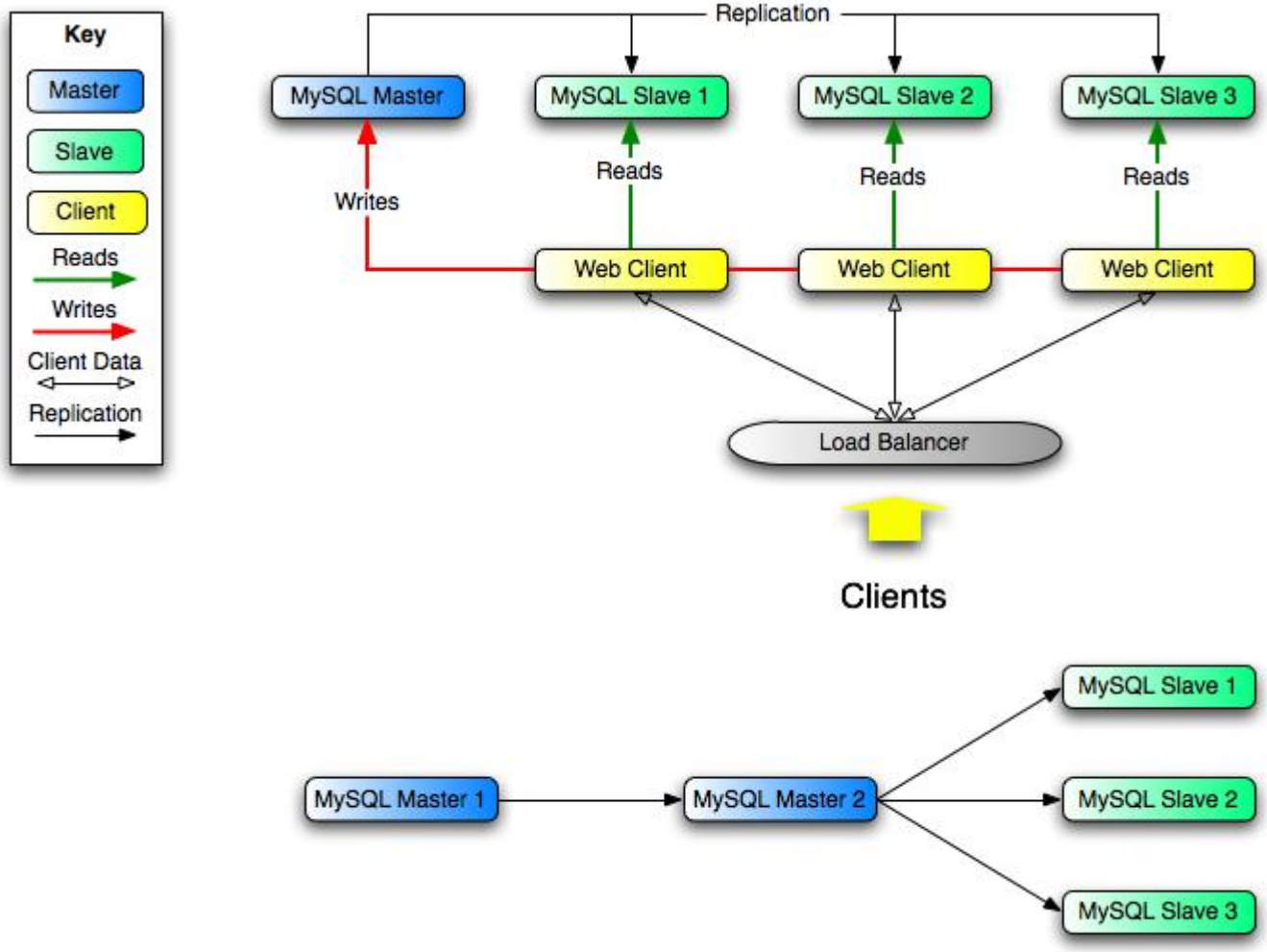
主从复制的优点

横向扩展解决方案：在多个从库之间分配负载以提高性能。所有写入和更新都必须在主服务器上进行。但是，读取可以在一个或多个从设备上进行。该模型可以提高写入性能，同时显着提高了从设备的读取速度。

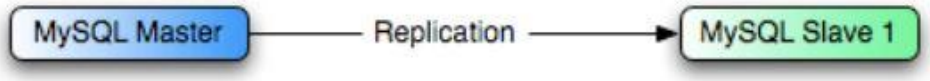
数据安全性：因为数据可以被延迟复制到从库，因此在主库发生的误操作，可以通过从库进行恢复。

数据分析：可以在主服务器上创建实时数据，而信息分析可以在从服务器上进行，而不会影响主服务器的性能。

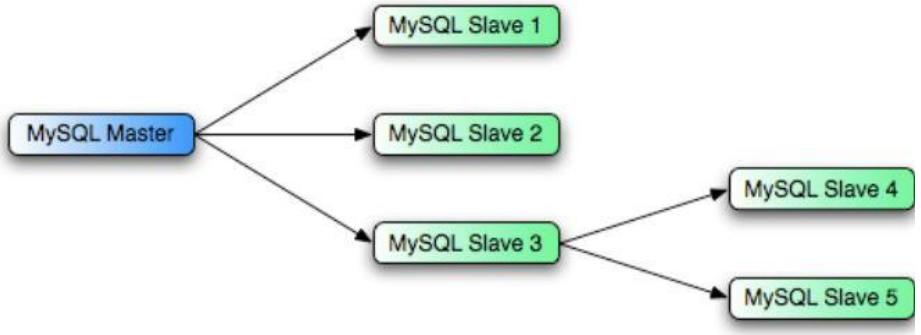
远程数据分发：可以使用复制为远程站点创建数据的本地副本，而无需永久访问主服务器。



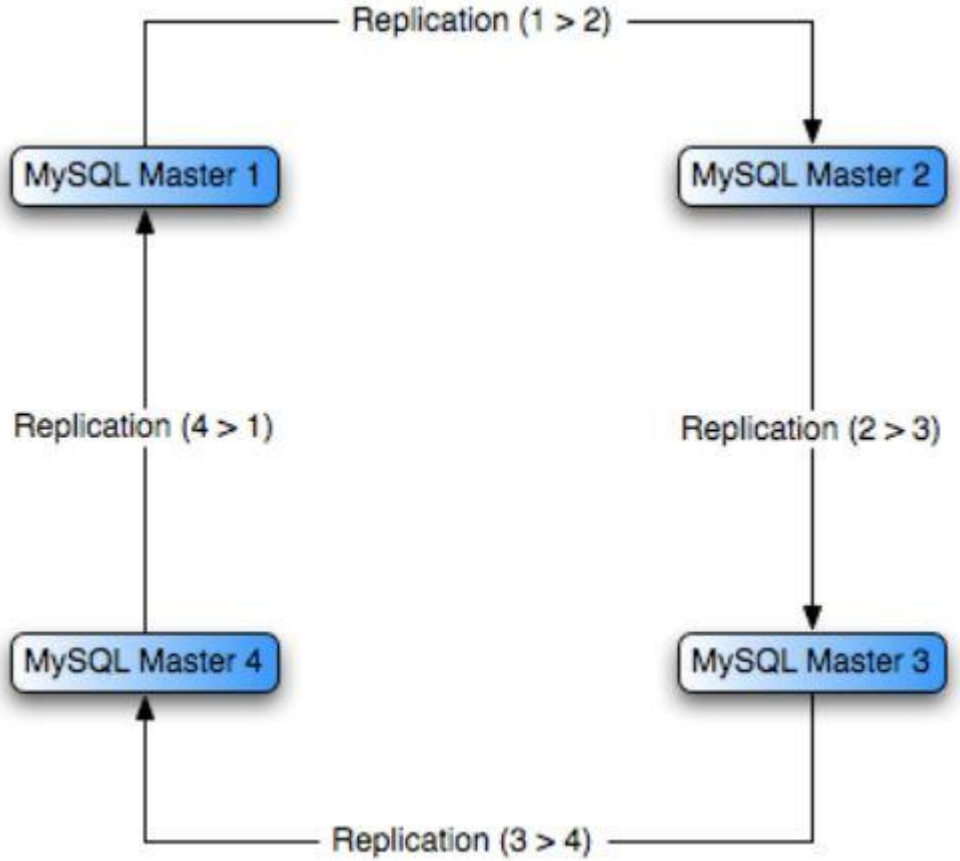
主从复制拓扑



Replication with a Single Slave



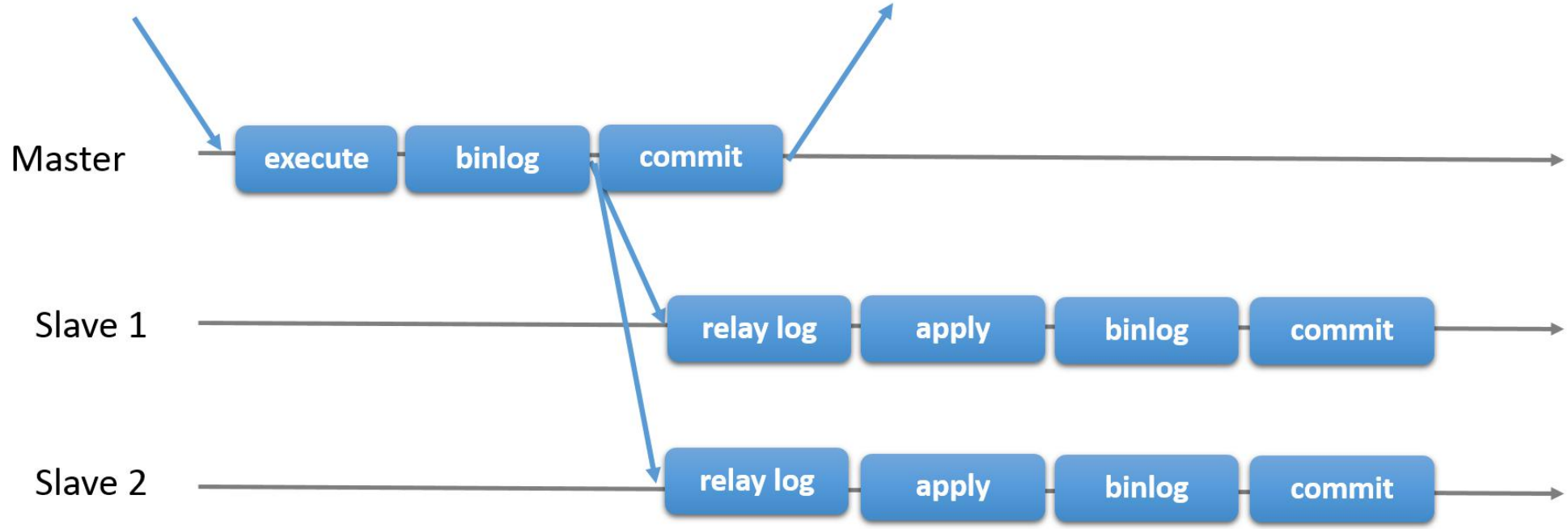
Replication Chains



Replication with Circular Masters

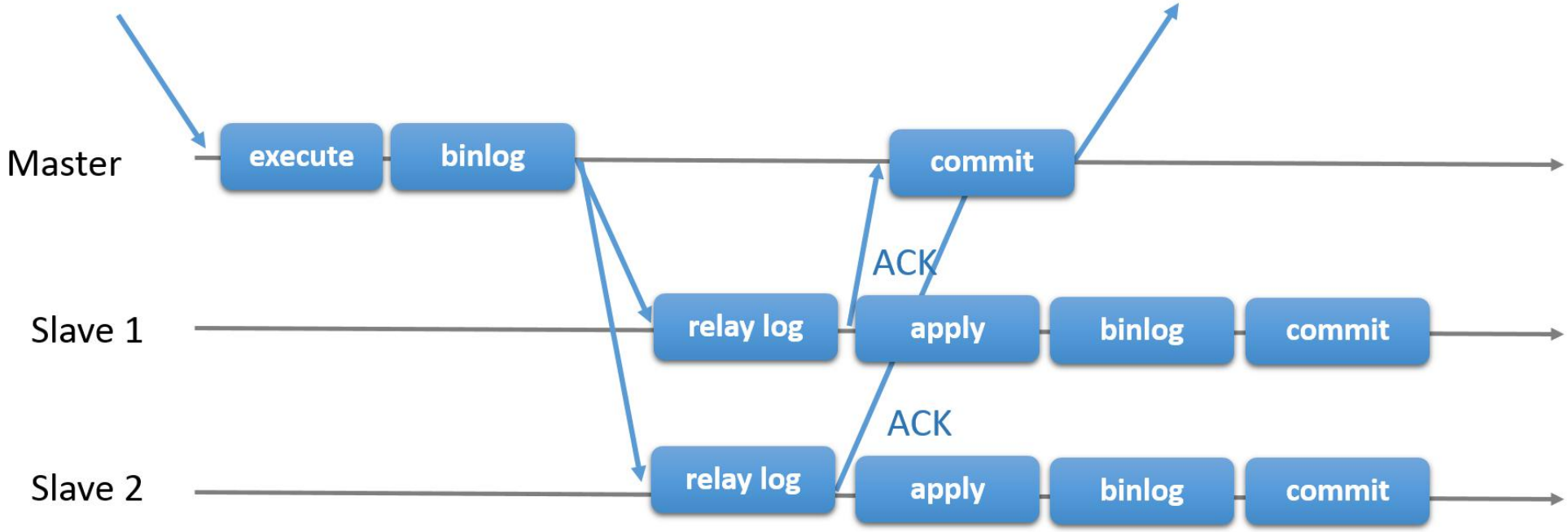
异步复制

MySQL 使用其默认配置进行复制过程中，主服务器从客户机接受更改事件，提交那些事件并将其写入二进制日志。在单独的线程中，主服务器将二进制日志流处理到连接的从属服务器。因为主服务器提交更改而不等待任何从属服务器的响应，所以这称为异步复制。



半同步复制

半同步复制过程中，主服务器提交事务后进行等待，直到至少一个半同步从属服务器确认已经收到了该事务。如果主服务器在提交事务后出现故障，则该事务还存在于至少一个从属服务器上。



半同步复制

半同步复制需要在性能和数据完整性之间进行权衡。使用半同步复制时事务速度比使用异步复制时慢，因为主服务器在提交之前等待从属服务器响应。

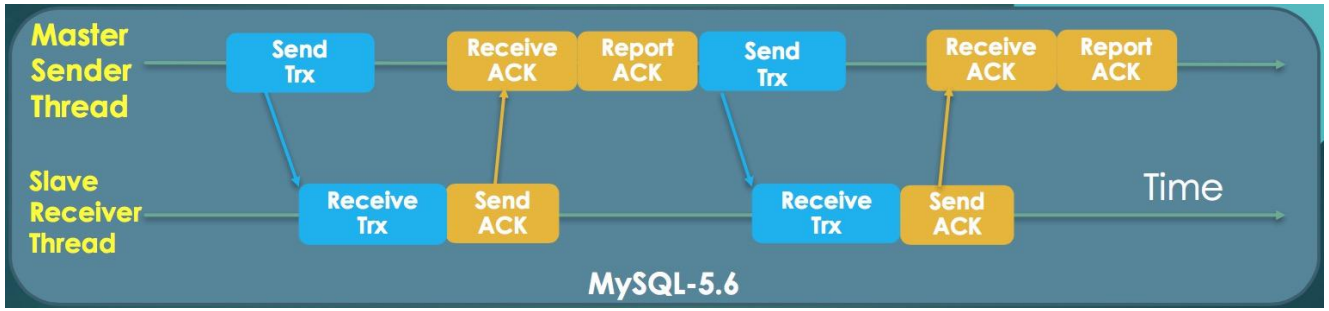
每个事务需要花费以下的额外时间开销：

- TCP/IP 往返以将提交发送到从属服务器
- 从属服务器在其中继日志中记录提交
- 主服务器等待从属服务器确认该提交

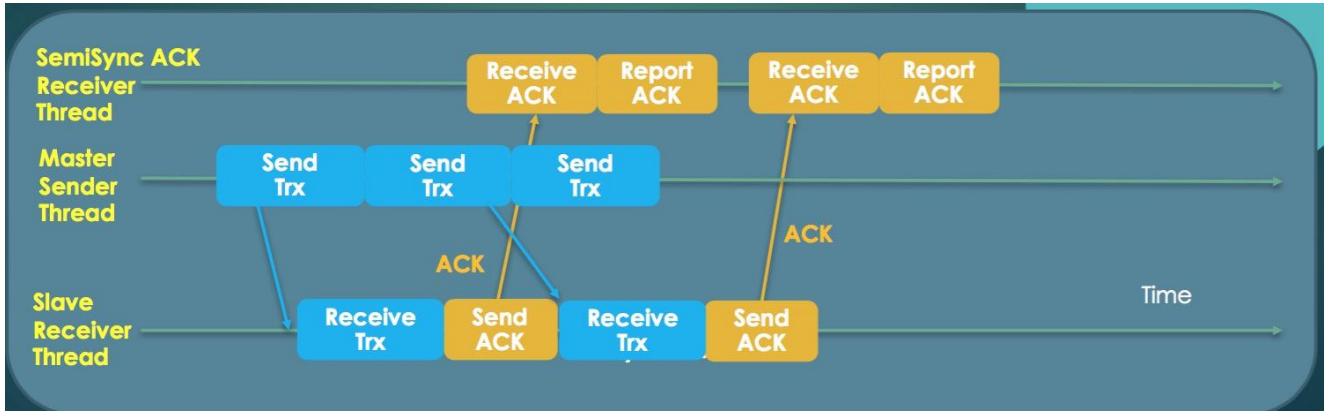
如果主服务器在超时期间内没有收到半同步从属服务器的响应，该主服务器仍提交该事务，但同步模式下降为异步模式。

半同步复制

- MySQL5.5/5.6的半同步复制是一个单工通讯方式，master把事务发送完毕后，要接收和处理slave的应答，处理完应答之后才能继续发送下一个事务。



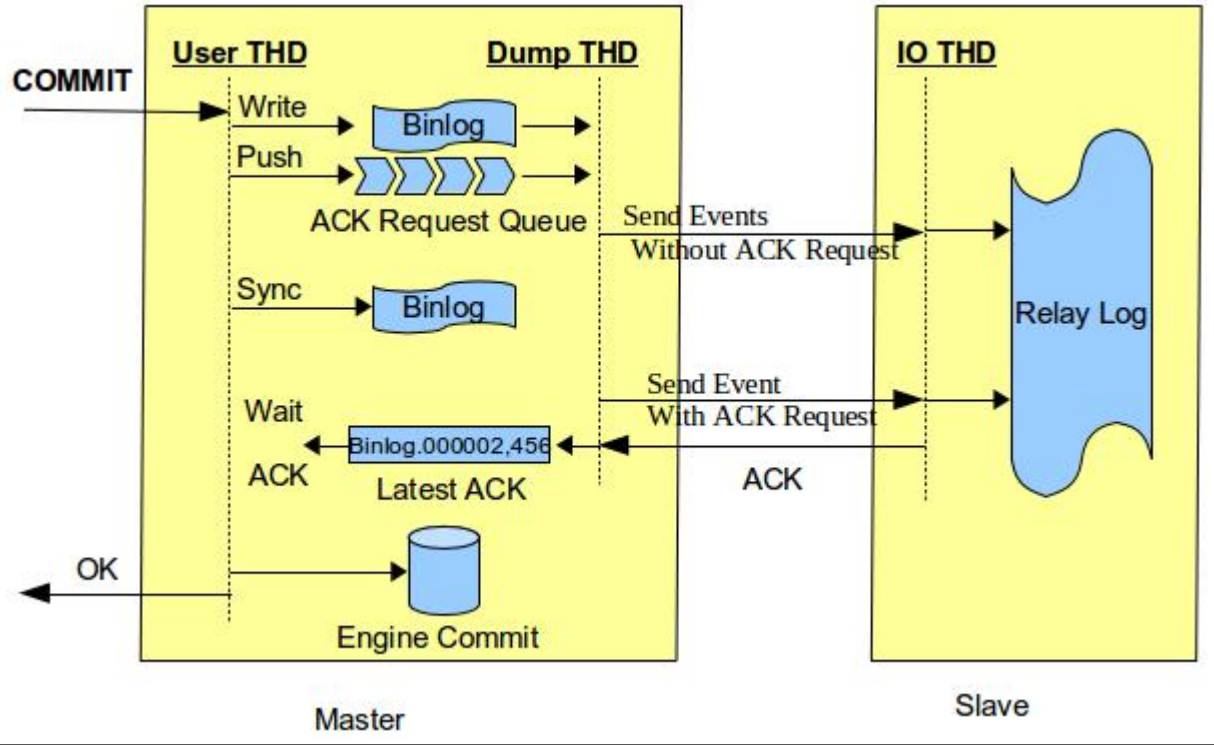
- MySQL5.7的半同步复制创建了单独的应答接收线程，发送和接收互不影响，发送效率得到大幅提升，相比MySQL5.5/5.6延迟会小很多，性能得到大幅提升。



增强半同步复制(5.7)

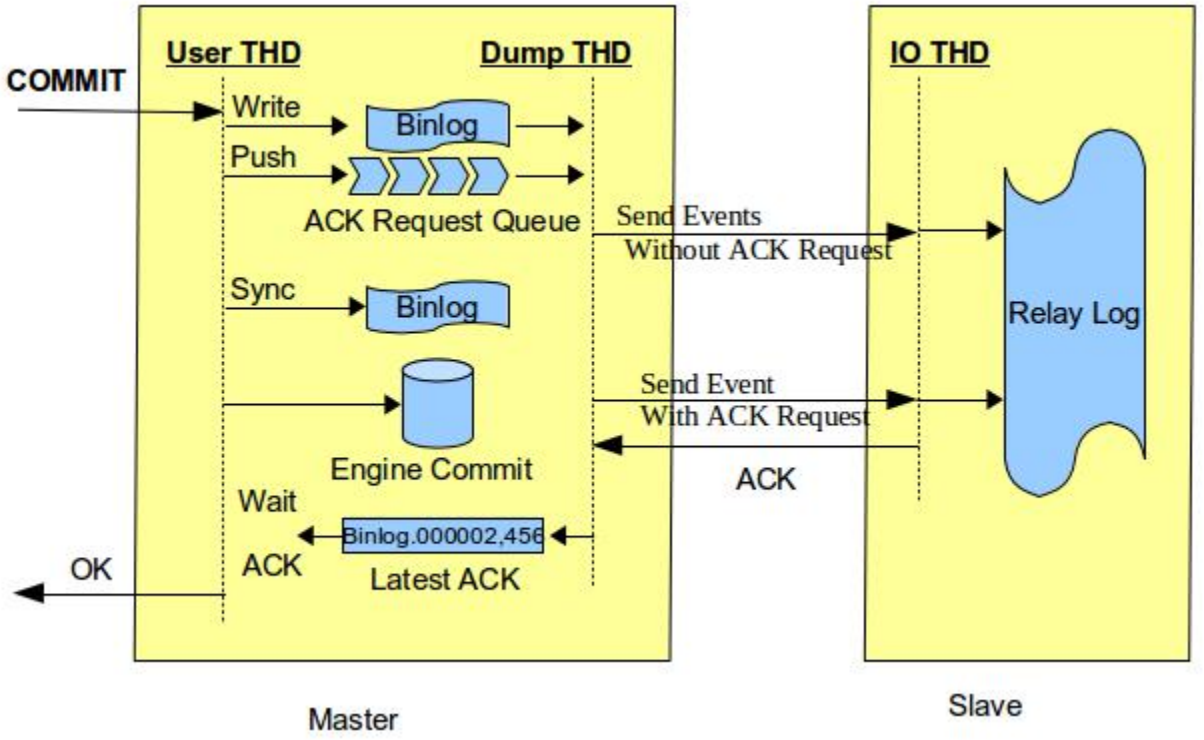
增强半同步复制特性由：rpl_semi_sync_master_wait_point 参数控制。

AFTER_SYNC（默认值）：主服务器将每个事务写入其二进制日志和从服务器，并将二进制日志同步到磁盘。主机在同步后等待从机确认事务接收。**收到确认后，主服务器将事务提交给存储引擎并将结果返回给客户端**，然后客户端可以继续。



增强半同步复制(5.7)

AFTER_COMMIT：主服务器将每个事务写入其二进制日志和从服务器，同步二进制日志，并**将事务提交到存储引擎**。提交后，主服务器等待事务接收的从属确认。收到确认后，主服务器将结果返回给客户端，然后客户端可以继续。



02

二进制日志

二进制日志(Binary Log)用途

MySQL的二进制日志，主要用于记录修改数据或有可能引起数据变更的MySQL语句。二进制日志中记录了对MySQL数据库执行更改的所有操作，并且记录了语句发生时间、执行时长、操作数据等其它额外信息，但是它不记录SELECT、SHOW等那些不修改数据的SQL语句。

二进制日志有两个重要目的：

- 用于复制，配置了主从复制的时候，主服务器会将其产生的二进制日志发送到从服务器端，从服务器端会利用这个二进制日志的信息在本地重做，实现主从同步。。

- 用户恢复，MySQL可以在全备和差异备份的基础上，利用二进制日志进行基于时间点或者事物ID的恢复操作。

启用二进制日志记录会使服务器性能下降。但是，二进制日志进行复制和恢复操作的好处会超过性能下降的影响。

二进制日志格式

MySQL二进制日志格式通过参数binlog_format参数控制。支持三种格式：

- **STATEMENT**：基于语句的日志记录
- **ROW**(默认值)：基于行的日志记录
- **MIXED**：使用混合格式的日志记录

```
root@localhost [(none)]>show variables like 'log_bin';

root@localhost [(none)]>show variables like 'binlog_format';

mysql> SET GLOBAL binlog_format = 'ROW';
```

基于SQL语句的复制(SBR)

优点

生成的日志量少，节约网络传输IO。

日志文件包含进行任何更改的所有语句，因此可用于审核数据库。

缺点

对于非确定性事件(UUID()，USER())，无法保证主从复制数据的一致性。

相比于基于行的复制方式在从上执行时需要更多的行锁。

基于行的复制(RBR)

优点

可以复制所有更改。是最安全的复制形式。

主服务器上需要更少的行锁，从而实现更高的并发性。

缺点

基于行的复制会产生更多的二进制日志。

无法在从站上看到从主站接收和执行的语句。

二进制日志参数配置

参数	值	描述
log_bin	ON	是否启用二进制日志
server_id	2003306	MySQL服务器复制环境中的唯一ID，默认为1，建议使用IP地址+端口方式进行唯一标识
binlog_format	ROW	二进制日志格式，5.7默认ROW
binlog_cache_size	65536	二进制日志的事物缓存值，默认32K
expire_logs_days	7	删除二进制文件之前的保留天数，mysql8中被binlog_expire_logs_seconds参数替代
log_bin_basename	/mysqldata/binlog/mysql-bin	二进制日志文件的路径及名称，默认文件名为<host_name>-bin，默认目录为data目录。
log_bin_index	/mysqldata/binlog/mysql-bin.index	二进制日志文件索引文件的路径及名称
binlog_checksum	CRC32	二进制日志事件校验和，默认值CRC32
max_binlog_size	1073741824	二进制日志最大文件大小，默认1G，最大1G

二进制日志命名

mysqld将数字扩展名附加到二进制日志名称以生成二进制日志文件名。每次服务器创建新日志文件时，该数字都会增加。每次启动或刷新日志时，服务器都会在系列中创建一个新文件。当二进制文件达到最大时(max_binlog_size)，该文件会创建新文件。

```
root@localhost [ducw]>show binary logs;
+------------------+-----------+
| Log_name         | File_size |
+------------------+-----------+
| mysql-bin.000005 |       217 |
| mysql-bin.000006 |       938 |
+------------------+-----------+
```

查看二进制日志记录

SHOW BINLOG EVENTS [IN 'log_name'] [FROM pos] [LIMIT [offset,] row_count]

root@localhost [(none)]>show binlog events in 'mysql-bin.000004' from 721 limit 10;

Log_name	Pos	Event_type	Server_id	End_log_pos	Info
mysql-bin.000004	721	Query	103306	793	BEGIN
mysql-bin.000004	793	Rows_query	103306	841	# insert into t values (2)
mysql-bin.000004	841	Table_map	103306	885	table_id: 114 (ducw.t)
mysql-bin.000004	885	Write_rows	103306	925	table_id: 114 flags: STMT_END_F
mysql-bin.000004	925	Xid	103306	956	COMMIT /* xid=103 */

查看二进制日志记录

[root@skydb1 binlog]# mysqlbinlog -v --base64-output=decode-rows mysql-bin.000004 | tail -12

```
#181203 19:36:36 server id 103306  end_log_pos 925 CRC32 0x690ce736     Write_rows: table id 114 flags: STMT_END_F
### INSERT INTO `ducw`.`t`
### SET
###   @1=2
# at 925
#181203 19:36:36 server id 103306  end_log_pos 956 CRC32 0x67a2ded4     Xid = 103
COMMIT/*!*/;
SET @@SESSION.GTID_NEXT= 'AUTOMATIC' /* added by mysqlbinlog */ /*!*/;
DELIMITER ;
# End of log file
/*!50003 SET COMPLETION_TYPE=@OLD_COMPLETION_TYPE*/;
/*!50530 SET @@SESSION.PSEUDO_SLAVE_MODE=0*/;
```

二进制日志安全

控制MySQL服务器将二进制日志同步到磁盘的频率。

- sync_binlog=0：禁用MySQL服务器将二进制日志同步到磁盘。相反，MySQL服务器依赖操作系统不时地将二进制日志刷新到磁盘，就像对任何其他文件一样。此设置提供最佳性能，但在电源故障或操作系统崩溃的情况下，服务器可能存在已提交尚未同步到二进制日志的事务。

- **sync_binlog=1 (默认)**：在提交事务之前，允许将二进制日志同步到磁盘。这是最安全的设置，但由于磁盘写入次数增加，可能会对性能产生负面影响。在电源故障或操作系统崩溃的情况下，二进制日志中缺少的事务仅处于准备状态。这允许自动恢复例程回滚事务，这保证了二进制日志中没有丢失任何事务。

- 当sync_binlog =N (N>0)： MySQL 在每写 N次 二进制日志binary log时，会将二进制日志binary log同步到磁盘中去。

二进制日志安全

控制MySQL服务器将log buffer写入log file的频率。

- innodb_flush_log_at_trx_commit = 0，log buffer将每秒一次地写入log file中，并且log file的刷到磁盘操作同时进行.该模式下，在事务提交的时候，不会主动触发写入磁盘的操作。

- **innodb_flush_log_at_trx_commit = 1 (默认)**，每次事务提交时MySQL都会把log buffer的数据写入log file，并且刷到磁盘中去.

- innodb_flush_log_at_trx_commit = 2，每次事务提交时MySQL都会把log buffer的数据写入log file.但是flush(刷到磁盘)操作并不会同时进行。该模式下,MySQL会每秒执行一次 刷到磁盘操作。

二进制日志安全

默认情况下，从属服务器状态日志存储在文件中，文件名是 `master.info` 和 `relay-log.info`，都存储在数据目录中。

```
[root@skydb2 data]# ls -la | grep info$
-rw-r-----. 1 mysql mysql       131 Dec  3 20:27 master.info
-rw-r-----. 1 mysql mysql        62 Dec  3 20:17 relay-log.info
```

如果将从属服务器状态日志存储在文件中，在记录事件与在状态日志中记录该事件的日志坐标之间的某点会发生故障。服务器在此类事件后重新启动时，状态文件和二进制日志将不一致，恢复变得困难。

通过将 master_info_repository 和 relay_log_info_repository 的值从 FILE 更改为 TABLE (8.0.2以后，默认为TABLE)，将状态日志存储在事务表中以提高性能并确保故障安全复制。该表称为 mysql.slave_master_info 和 mysql.slave_relay_log_info。

中继日志(Relay Log)

从服务器I/O线程将主服务器的二进制日志读取过来记录到从服务器本地文件，然后SQL线程会读取中继日志的内容并应用到从服务器。

默认情况下，中继日志文件名为 `<host_name>-relay-bin.<nnnnnn>`，索引文件名为 `<host_name>-relay-bin.index`。

root@localhost [(none)]>**show variables like 'relay_log%'**;

Variable_name	Value
relay_log	
relay_log_basename	/mysqldata/data/skydb2-relay-bin
relay_log_index	/mysqldata/data/skydb2-relay-bin.index
relay_log_info_file	relay-log.info
relay_log_info_repository	FILE
relay_log_purge	ON
relay_log_recovery	OFF
relay_log_space_limit	0

中继日志

root@localhost [(none)]>**show relaylog events in 'skydb2-relay-bin.000014'**;

```
root@localhost [(none)]>show relaylog events in 'skydb2-relay-bin.000014';
+-------------------------+------+----------------+-----------+-------------+-----------------------------------------------------+
| Log_name                | Pos  | Event_type     | Server_id | End_log_pos | Info                                                |
+-------------------------+------+----------------+-----------+-------------+-----------------------------------------------------+
| skydb2-relay-bin.000014 |    4 | Format_desc    |    203306 |         123 | Server ver: 5.7.24-log, Binlog ver: 4               |
| skydb2-relay-bin.000014 |  123 | Previous_gtids |    203306 |         194 | afa4a9cc-e184-11e8-9e89-0242aca8380a:1-9            |
| skydb2-relay-bin.000014 |  194 | Rotate         |    103306 |           0 | mysql-bin.000004;pos=4                              |
| skydb2-relay-bin.000014 |  241 | Format_desc    |    103306 |         123 | Server ver: 5.7.24-log, Binlog ver: 4               |
| skydb2-relay-bin.000014 |  360 | Rotate         |         0 |         407 | mysql-bin.000004;pos=194                            |
| skydb2-relay-bin.000014 |  472 | Query          |    103306 |         356 | use `ducw`; create table t (id int)                 |
| skydb2-relay-bin.000014 |  634 | Query          |    103306 |         493 | BEGIN                                               |
| skydb2-relay-bin.000014 |  869 | Gtid           |    103306 |         721 | SET @@SESSION.GTID_NEXT= '......-0242aca8380a:12'   |
| skydb2-relay-bin.000014 |  934 | Query          |    103306 |         793 | BEGIN                                               |
| skydb2-relay-bin.000014 | 1006 | Rows_query     |    103306 |         841 | # insert into t values (2)                          |
| skydb2-relay-bin.000014 | 1054 | Table_map      |    103306 |         885 | table_id: 114 (ducw.t)                              |
| skydb2-relay-bin.000014 | 1098 | Write_rows     |    103306 |         925 | table_id: 114 flags: STMT_END_F                     |
| skydb2-relay-bin.000014 | 1138 | Xid            |    103306 |         956 | COMMIT /* xid=103 */                                |
+-------------------------+------+----------------+-----------+-------------+-----------------------------------------------------+
```

全局事务标识符(GTID)

全局事务标识符 (Global Transaction Identifier，GTID) 唯一的标识复制网络中的每个事务。使用GTID时，每个事务在主库和从库上提交时都会进行标识和跟踪; 这意味着在启动新从库或故障转移到新主库时，使用GTID不必再引用日志文件和位置，这极大地简化了这些任务。

每个服务器的 UUID 存储在数据目录中的 auto.cnf 文件中。如果该文件不存在，MySQL 会创建该文件并生成新的 UUID，将其放在该新文件中。

[root@skydb1 data]# cat auto.cnf

[auto]

server-uuid=afa4a9cc-e184-11e8-9e89-0242aca8380a

root@localhost [(none)]>select @@server_uuid;

开启GTID

每个 GTID 的形式为 <source-uuid>:<transaction-id>。
afa4a9cc-e184-11e8-9e89-0242aca8380a:1

GTID 集包含一系列 GTID：
afa4a9cc-e184-11e8-9e89-0242aca8380a:1-15

gtid_mode=ON：与每个事务一起记录唯一的 GTID
enforce_gtid_consistency=ON：禁止无法以事务安全方式记录的事件
log_slave_updates=TRUE：从库将复制的事件记录到从属服务器的二进制日志

03

复制配置

参数设置

```
[root@skydb4 mysqldata]# cat /etc/my.cnf
[client]
port                                = 3306
socket                              = /mys
user                                = root
password                            = orac

[mysql]
auto-rehash
prompt                              = "\\u

[mysqld]
user                                = mysq
basedir                             = /opt
datadir                             = /mys
socket                              = /mys
server_id                           = 5640
port                                = 3306
character_set_server                = utf8
log_timestamps                      = syst
default_time_zone                   = '+8:
explicit_defaults_for_timestamp     = ON
####: for binlog
binlog_format                       = ROW
log_bin                             = /mysqldata/binlog/skydb4-bin
log_slave_updates                   = ON
expire_logs_days                    = 7
sync_binlog                         = 1

####: for error-log
log_error                           = /mysqldata/log/skydb4.err

####: for slow query log
slow_query_log                      = ON
slow_query_log_file                 = /mysqldata/log/skydb4.slow
long_query_time                     = 1.000000

####: replication
master_info_repository              = TABLE
relay_log_info_repository           = TABLE

####: for gtid
gtid_mode                           = ON
enforce_gtid_consistency            = ON
```

参数设置

为每个服务器配置唯一 server_id。

主服务器：

- 启用二进制日志并启用 TCP/IP 网络。
- 创建具有 REPLICATION SLAVE 特权的新用户。

每个从属服务器：

- 从主服务器恢复备份。
- 执行 CHANGE MASTER TO 语句。
- 使用已经包含数据库的主服务器创建复制，必须首先为从属服务器创建该数据库的副本。
- 使用全局事务标识符，不需要记录日志坐标。

主库配置复制

必要参数：
server_id|binlog_format|log_bin|gtid_mode|enforce_gtid_consistency

创建用户：
mysql> CREATE USER 'repl'@'%' IDENTIFIED BY 'repl4slave';
mysql> GRANT REPLICATION SLAVE ON *.* TO 'repl'@'%';

用户验证：
[root@skydb5 ~]# mysql -h skydb4 -urepl -p

从库配置复制

配置复制：

```
root@localhost [(none)]>HELP CHANGE MASTER TO;
root@localhost [(none)]>
CHANGE MASTER TO MASTER_HOST='172.168.56.40' , \
    -> MASTER_PORT=3306, \
    -> MASTER_USER='repl', \
    -> MASTER_PASSWORD='repl4slave', \
    -> MASTER_AUTO_POSITION=1;
```

启动复制线程并查看状态

```
root@localhost [(none)]>start slave;
root@localhost [(none)]>show slave status\G
```

CHANGE MASTER TO 详解

- MASTER_HOST 和 MASTER_PORT 值指定主服务器的主机名和 TCP 端口号。
- MASTER_USER 和 MASTER_PASSWORD 值指定具有 REPLICATION SLAVE 特权的主服务器上帐户的帐户详细信息。
- MASTER_LOG_FILE 和 MASTER_LOG_POS 值包含从属服务器开始进行复制的二进制日志位置的日志坐标。可以通过执行 SHOW MASTER STATUS 语句从主服务器获取文件和位置：
- 使用MASTER_AUTO_POSITION = 1，slave连接master将使用基于GTID的复制协议，这种方式不需要MASTER_LOG_FILE 和 MASTER_LOG_POS。

测试验证

主库：

```
[root@skydb4 ~]# mysql
root@localhost [(none)]>create database ducw;
Query OK, 1 row affected (0.01 sec)

root@localhost [(none)]>use ducw;
Database changed
root@localhost [ducw]>create table t (id int);
Query OK, 0 rows affected (0.03 sec)

root@localhost [ducw]>insert into t values (1);
Query OK, 1 row affected (0.04 sec)
```

从库：

```
[root@skydb5 ~]# mysql
root@localhost [(none)]>show databases;

root@localhost [(none)]>use ducw;

root@localhost [ducw]>show tables;

root@localhost [ducw]>select * from t;
```

半同步复制注册插件

root@localhost [(none)]>show variables like '%plugin%';

[root@skydb4 ~]# ls -la /opt/mysql5.7/lib/plugin/ | grep semi

-rwxr-xr-x. 1 7161 31415   708290 Oct 4 06:15 semisync_master.so

-rwxr-xr-x. 1 7161 31415   152309 Oct 4 06:15 semisync_slave.so

主库注册插件：

mysql> INSTALL PLUGIN rpl_semi_sync_master SONAME 'semisync_master.so';

从库注册插件：

mysql> INSTALL PLUGIN rpl_semi_sync_slave SONAME 'semisync_slave.so';

半同步复制参数修改

root@localhost [(none)]>show variables like 'rpl_semi%';

主库参数设置：

```
mysql> SET GLOBAL rpl_semi_sync_master_enabled = 1;
mysql> SET GLOBAL rpl_semi_sync_master_timeout = 100000; # 100 seconds
```

从库参数设置：

```
mysql> SET GLOBAL rpl_semi_sync_slave_enabled = 1;
mysql> STOP SLAVE IO_THREAD;
mysql> START SLAVE IO_THREAD;
```

半同步降级

主从连接超时的情况下，半同步会降级成异步模式：

2018-12-05T08:51:17.500123-00:00 9 [Warning] Timeout waiting for reply of binlog (file: skydb4-bin.000014, pos: 1460), semi-sync up to file , position 4.

2018-12-05T08:51:17.500175-00:00 9 [Note] Semi-sync replication switched OFF.

连接恢复后，异步会恢复成半同步：

2018-12-05T08:52:41.773742-00:00 14 [Note] Start binlog_dump to master_thread_id(14) slave_server(56503306), pos(, 4)

2018-12-05T08:52:41.773786-00:00 14 [Note] Start semi-sync binlog_dump to slave (server_id: 56503306), pos(, 4)

2018-12-05T08:52:41.799793-00:00 0 [Note] Semi-sync replication switched ON at (skydb4-bin.000014, 1460)

04

复制线程

复制线程

MySQL复制功能使用三个线程实现，一个在主服务器上，两个在从服务器上：

Binlog转储线程：主设备创建一个线程，以便在从设备连接时将二进制日志内容发送到从设备。可以SHOW PROCESSLIST在主服务器的输出中将此线程标识为Binlog Dump线程。

二进制日志转储线程获取主机二进制日志上的锁，用于读取要发送到从机的每个事件。一旦读取了事件，即使在事件发送到从站之前，锁也会被释放。

从属I/O线程：在START SLAVE从属服务器上发出语句时，从属服务器会创建一个I/O线程，该线程连接到主服务器并要求它发送二进制日志中记录的更新。从属I/O线程读取主Binlog Dump线程发送的更新并将它们复制到包含从属中继日志的本地文件。此线程的状态显示为 Slave_IO_Running 输出 SHOW SLAVE STATUS。

从属SQL线程：从站创建一个SQL线程来读取由从I/O线程写入的中继日志，并执行其中包含的事件。

控制从属服务器线程

控制从属服务器线程：

- START SLAVE;
- STOP SLAVE;

单独控制线程：

- START SLAVE IO_THREAD;
- STOP SLAVE SQL_THREAD;

启动线程直到指定的条件：

- START SLAVE UNTIL SQL_AFTER_MTS_GAPS;
- START SLAVE IO_THREAD UNTIL SQL_AFTER_GTIDS = 0ed18583-47fd-11e2-92f3-0019b944b7f7:338;

复制从属服务器 I/O 线程状态

Connecting to master：线程正尝试连接到主服务器。

Waiting for master to send event：线程已经连接到主服务器并且正在等待二进制日志事件到达。如果主服务器处于空闲状态，此状态可能持续很长时间。如果等待持续 slave_read_timeout 秒，则发生超时。此时，线程考虑断开连接并尝试重新连接。

Queueing master event to the relay log：线程已经读取事件并且正在将其复制到中继日志，从而 SQL 线程可以处理该事件。

Waiting to reconnect after a failed binlog dump request：如果二进制日志转储请求失败（由于断开连接），线程在其休眠时转入此状态，然后定期尝试重新连接。

复制从属服务器 I/O 线程状态

Reconnecting after a failed binlog dump request：线程正尝试重新连接到主服务器。

Waiting to reconnect after a failed master event read：读取时出错（由于断开连接）。线程在尝试重新连接之前休眠 master-connect-retry 秒。

Reconnecting after a failed master event read：线程正尝试重新连接到主服务器。重新建立连接后，状态变为 Waiting for master to send event。

Waiting for the slave SQL thread to free enough relay log space：该状态显示正在使用非零 relay_log_space_limit 值，并且中继日志已经增加得足够大，以至其合并大小超过了此值。I/O 线程正在等待，直到 SQL 线程通过处理中继日志内容以便可以删除一些中继日志文件来释放足够空间。

复制从属服务器 SQL 线程状态

Waiting for the next event in relay log：这是 Reading event from the relay log 之前的初始状态。

Reading event from the relay log：线程已经从中继日志中读取了事件，从而可以处理该事件。

Making temp file：线程正在执行 LOAD DATA INFILE 语句，并且正在创建包含数据的临时文件，从属服务器从该数据中读取行。

Slave has read all relay log; waiting for the slave I/O thread to update it：线程已经处理了中继日志文件中的所有事件，现在正在等待 I/O 线程将新事件写入中继日志。

Waiting until MASTER_DELAY seconds after master executed event：SQL 线程已经读取事件，但是正等待从属服务器延迟结束。通过 CHANGE MASTER TO 的 MASTER_DELAY 选项设置此延迟。

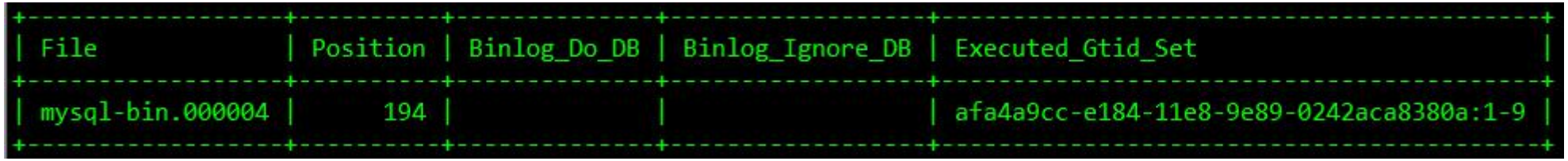
Waiting for an event from Coordinator：在多线程从属服务器上，工作线程正等待协调线程向工作队列分配作业。

05

复制故障排查

主库复制状态

```
root@localhost [(none)]>show master status\G
*************************** 1. row ***************************
             File: mysql-bin.000004
         Position: 194
     Binlog_Do_DB:
 Binlog_Ignore_DB:
Executed_Gtid_Set: ......
```



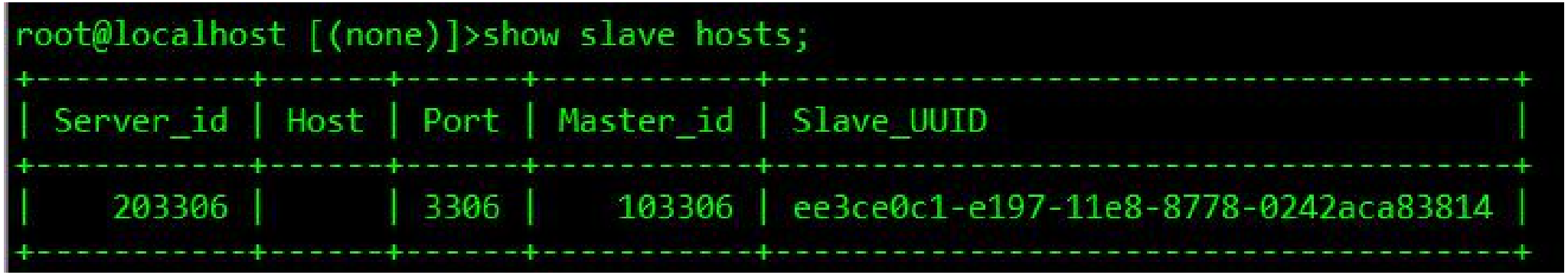
File	Position	Binlog_Do_DB	Binlog_Ignore_DB	Executed_Gtid_Set
mysql-bin.000004	194			afa4a9cc-e184-11e8-9e89-0242aca8380a:1-9

主库复制进程信息

```
root@localhost [(none)]>show processlist\G
*************************** 1. row ***************************
     Id: 2
   User: repl
   Host: 172.168.56.20:42444
     db: NULL
Command: Binlog Dump GTID
   Time: 881
  State: Master has sent all binlog to slave;
   Info: NULL
```

从库注册信息

```
root@localhost [(none)]>show slave hosts\G
*************************** 1. row ***************************
 Server_id: 203306
      Host:
      Port: 3306
 Master_id: 103306
Slave_UUID: ee3ce0c1-e197-11e8-8778-0242aca83814
```



```
root@localhost [(none)]>show slave hosts;
+-----------+------+------+-----------+--------------------------------------+
| Server_id | Host | Port | Master_id | Slave_UUID                           |
+-----------+------+------+-----------+--------------------------------------+
|    203306 |      | 3306 |    103306 | ee3ce0c1-e197-11e8-8778-0242aca83814 |
+-----------+------+------+-----------+--------------------------------------+
```

从库复制状态 (主库信息)

```
root@localhost [(none)]>show slave status\G
*************************** 1. row ***************************
               Slave_IO_State: Waiting for master to send event
                  Master_Host: 172.168.56.10
                  Master_User: repl
                  Master_Port: 3306
                Master_Log_File: mysql-bin.000004
            Read_Master_Log_Pos: 194
                 Relay_Log_File: skydb2-relay-bin.000011
                  Relay_Log_Pos: 407
          Relay_Master_Log_File: mysql-bin.000004
```

从库复制状态 (主库信息)

- **主服务器日志坐标**：Master_Log_File 和 Read_Master_Log_Pos 列标识主服务器二进制日志中 I/O 线程已经传输的最近事件的坐标。这些列可与您在主服务器上执行 SHOW MASTER STATUS 时显示的坐标进行比较。如果 Master_Log_File 和 Read_Master_Log_Pos 的值远远落后于主服务器上的那些值，这表示主服务器与从属服务器之间事件的网络传输存在延迟。
- **中继日志坐标**：Relay_Log_File 和 Relay_Log_Pos 列标识从属服务器中继日志中 SQL 线程已经执行的最近事件的坐标。这些坐标对应于 Relay_Master_Log_File 和 Exec_Master_Log_Pos 列标识的主服务器二进制日志中的坐标。
- 如果 Relay_Master_Log_File 和 Exec_Master_Log_Pos 列的输出远远落后于 Master_Log_File 和 Read_Master_Log_Pos 列（表示 I/O 线程的坐标），这表示 SQL 线程（而不是 I/O 线程）中存在延迟。即，它表示复制日志事件快于执行这些事件。

从库复制状态 (复制进程及对象信息)

```
root@localhost [(none)]>show slave status\G
*************************** 1. row ***************************
               Slave_IO_Running: Yes
              Slave_SQL_Running: Yes
                Replicate_Do_DB:
            Replicate_Do_Table:
        Replicate_Wild_Do_Table:
    Replicate_Wild_Ignore_Table:
                     Last_Errno: 0
                     Last_Error:
                   Skip_Counter: 0
```

从库复制状态 (复制进程及对象信息)

Slave_*_Running：Slave_IO_Running 和 Slave_SQL_Running 列标识从属服务器的 I/O 线程和 SQL 线程当前正在运行、未运行还是正在运行但尚未连接到主服务器。可能值分别为 Yes、No 或 Connecting。

```
root@localhost [(none)]>stop slave sql_thread;
root@localhost [(none)]>stop slave io_thread;

root@localhost [(none)]>show slave status\G
             Slave_IO_Running: No
            Slave_SQL_Running: No
```

从库复制状态 (复制进程及对象信息)

- Last_IO_Error、Last_SQL_Error：分别导致 I/O 线程或 SQL 线程停止的最新错误的错误消息。在正常复制过程中，这些字段是空的。如果发生错误并导致消息显示在以上任一字段中，则错误值也显示在错误日志中。
- Last_IO_Errno、Last_SQL_Errno：与分别导致 I/O 线程或 SQL 线程停止的最新错误关联的错误编号。在正常复制过程中，这些字段包含编号 0。
- Last_IO_Error_Timestamp、Last_SQL_Error_Timestamp：分别导致 I/O 线程或 SQL 线程停止的最新错误的时间戳，格式为 YYMMDD HH:MM:SS。在正常复制过程中，这些字段是空的。

从库复制状态 (复制状态及事物信息)

```
root@localhost [(none)]>show slave status\G
*************************** 1. row ***************************
      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: afa4a9cc-e184-11e8-9e89-0242aca8380a:1-9
             Executed_Gtid_Set: afa4a9cc-e184-11e8-9e89-0242aca8380a:1-9
                 Auto_Position: 1
```

从库进程信息

```
root@localhost [(none)]>start slave io_thread;
```

```
+----+-------------+-----------+------+---------+------+-----------------------------------+------------------+
| Id | User        | Host      | db   | Command | Time | State                             | Info             |
+----+-------------+-----------+------+---------+------+-----------------------------------+------------------+
|  2 | root        | localhost | NULL | Query   |    0 | starting                          | show processlist |
|  3 | system user |           | NULL | Connect |    1 | Waiting for master to send event | NULL             |
+----+-------------+-----------+------+---------+------+-----------------------------------+------------------+
```

```
root@localhost [(none)]>start slave sql_thread;
```

```
root@localhost [(none)]>show processlist;
+----+-------------+-----------+------+---------+------+-----------------------------------------------------------------+------------------+
| Id | User        | Host      | db   | Command | Time | State                                                           | Info             |
+----+-------------+-----------+------+---------+------+-----------------------------------------------------------------+------------------+
|  2 | root        | localhost | NULL | Query   |    0 | starting                                                        | show processlist |
|  3 | system user |           | NULL | Connect |   86 | Waiting for master to send event                                | NULL             |
|  4 | system user |           | NULL | Connect |   16 | Slave has read all relay log; waiting for more updates          | NULL             |
|  5 | system user |           | NULL | Connect |   16 | Waiting for an event from Coordinator                           | NULL             |
|  6 | system user |           | NULL | Connect |   16 | Waiting for an event from Coordinator                           | NULL             |
|  7 | system user |           | NULL | Connect |   16 | Waiting for an event from Coordinator                           | NULL             |
|  8 | system user |           | NULL | Connect |   16 | Waiting for an event from Coordinator                           | NULL             |
+----+-------------+-----------+------+---------+------+-----------------------------------------------------------------+------------------+
```

从库错误日志

root@localhost [(none)]>**show variables like 'log_error'**;

错误日志包含关于从库发生的所有的日志信息：

2018-12-03T15:18:02.092934-00:00 10 [Note] Slave I/O thread: **Start semi-sync** replication to master 'repl@172.168.56.10:3306' in log 'mysql-bin.000004' at position 194

2018-12-03T15:18:02.094212-00:00 10 [Note] Slave I/O thread for channel '': **connected to master** 'repl@172.168.56.10:3306'，replication started in log 'mysql-bin.000004' at position 194

谢谢！