Unwinding Replication

From basics into subtleties of binary logging and multi-threaded applier

Andrei Elkin, Senior MariaDB developer
Designed in MySQL even before transaction. Serves a number of critical missions including:

- Failover, Backup and point-in-time recovery
- Load balancer
- Auditing
- Error case analysis
Replication conceptually is 2PC
Transaction state transaction

- Executing
  - prepared
  - Binlogged
    - committed
    - Shipping dumping
  - Relay-logged
  - Executing slave
  - Committed on slave
Binlog group commit
Binlog Group Commit ordered
binary logging: event group

Transaction

DDL Query

Non-transactional DML

Replication event group
event group: example

# at 1283
#180223 19:37:35 server id 1 end_log_pos 1325 CRC32 0x0889dcf GTID 11-1-1 trans
/*!100001 SET @session.gtid_domain_id=11*//*!*/;
/*!100001 SET @session.gtid_seq_no=1*//*!*/;
BEGIN
/*!*/;
# at 1325
#180223 19:37:35 server id 1 end_log_pos 1414 CRC32 0xf1ca2d36 Query thread_id=9 \
exec_time=0 error_code=0
SET TIMESTAMP=1519411055/*!*/;
insert into t set a=11
/*!*/;
# at 1414
#180223 19:37:35 server id 1 end_log_pos 1445 CRC32 0x2112e803 Xid = 42
COMMIT/*!*/;
Events shipping: Dump thread
Events receiving: IO thread

- Dump thread
- Slave IO
- Relaylog
- Slave driver aka SQL thread

- "SELECT from file:pos"
- send data to slave A
- append
- notify
GTID idea

Master

Binlog

Slave A

Slave B

retrieve starting from 200

retrieve starting from 100
GTID: required in Multi-Source replication

- Master 1
- Binlog

- Master 2
- Binlog

- Slave A
- Slave B

retrieve from 200

Retrieve from a "GTID"(2,200) ?!
GTID definition

- **GTID**
  - domain_id,
  - server_id,
  - seq_no

- Identifier
  - 1

- Event group
  - or
  - "transaction"
GTID and Multi-Sourced Replication

Transactions from different domains are executed independently.
Events execution on slave: single-threaded mode

Check flow control conditions:
- `skip`
- `delay`
- `filter (--replicate-*, GTID)`

execute
Events execution: parallel scheduling
Events execution: ordered commit
GTID execution: gtid_slave_pos table
Recovery: committed data may be lost

- Executing
- Prepared
- Commit in memory
- Commit on disk
- Stock replicatable queries as events into private binlog cache
- Create Binlog_checkpoint event
- CREATE and log GTID_BEGIN, binlog the cache to complete with Xid_log_event
Recovery without Binlog_checkpoint

- Commit on disk
- Commit in memory
- Xid
- data
- FLUSH BINARY LOG
- Create Rotate_log event
- Old
- Xid
- New
- write in Old
Recovery: server crash
Recovery: post-restart decisions

- Server startup process
- Binlog
- Engine

1. Post-failure restart by server
2. Compute XID e.g. 100
3. Get back prepared trx XIDs
4. Read XIDs from binlogged XID log events
5. Rollback or commit
Ongoing and perspective projects

1) XA replication MDEV-7974
2) Eager replication as follow-up XA replication
3) Parallel slave group commit
4) Relay-log-less slave
5) Committed GTID tracker by engine
6) Binlog-less “relay” slave in chain replication
7) Consensus protocol on membership in replication configuration
8) E.g Paxos-like mode semi-sync
Ongoing: XA replication MDEV-7974

XA replication and its follow-up
Follow-ups: active replication
MDEV-16404: slave group commit
References:

https://kristiannielsen.livejournal.com/16826.html
https://kristiannielsen.livejournal.com/16382.html

MariaDB for Advanced DBAs. MariaDB Training, MariaDB (c)
Database replication techniques: a three parameter classification
M. Wiesmann; F. Pedone; A. Schiper; B. Kemme; G. Alonso