Unwinding Replication

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

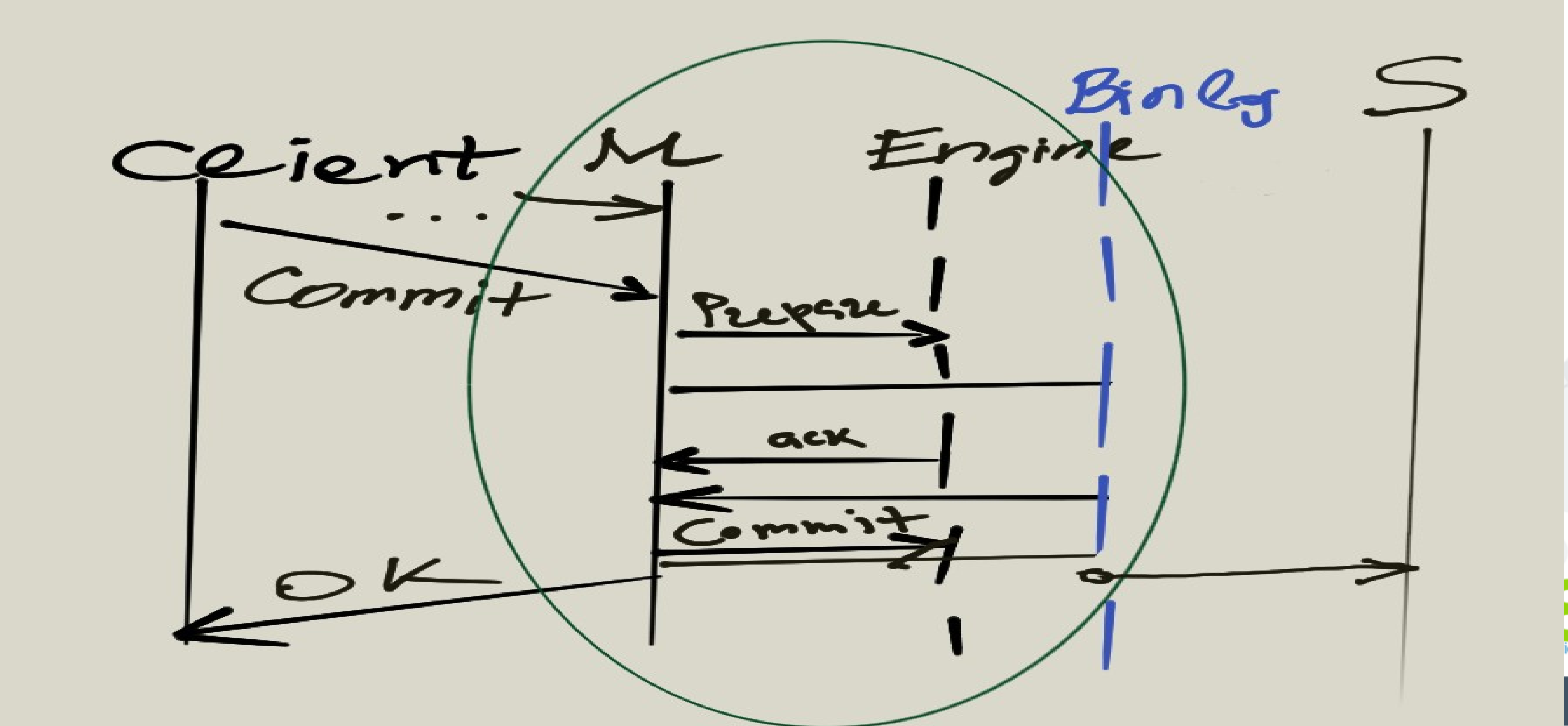
Andrei Elkin, Senior MariaDB
developer

Designed in MySQL even before transaction. Serves a number of critical missions including:

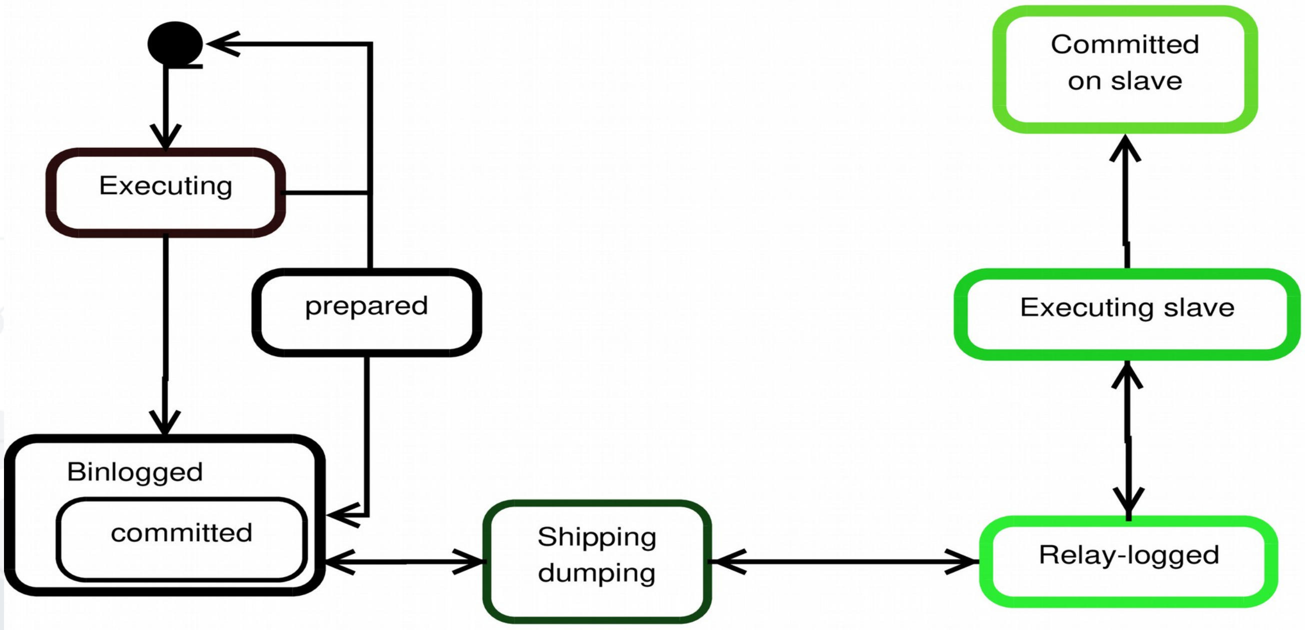
- Failover, Backup and point-in-time recovery
- Load balancer
- Auditing
- Error case analysis

USA
8
15
IAAF
4

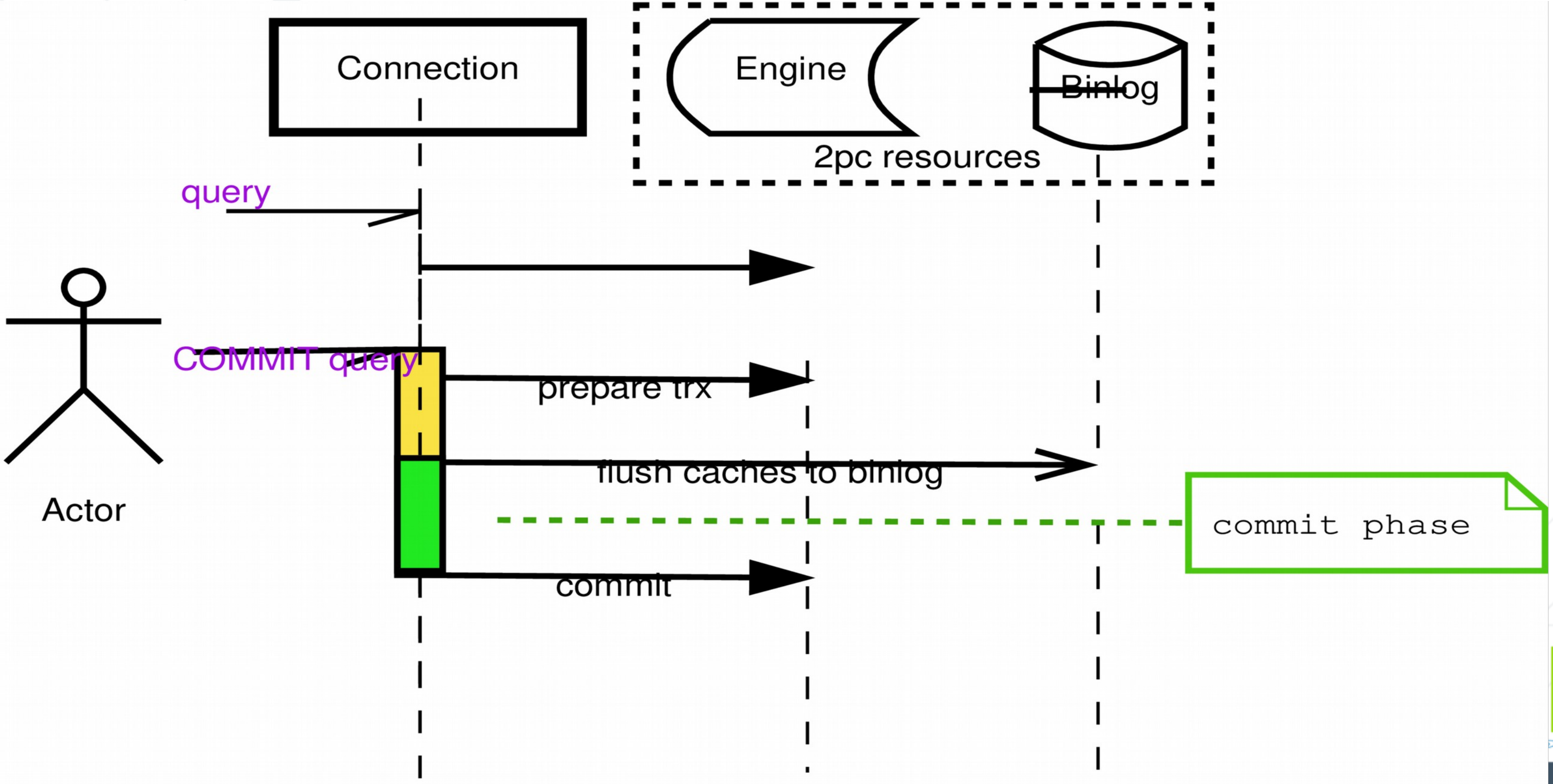
Replication conceptually is 2PC



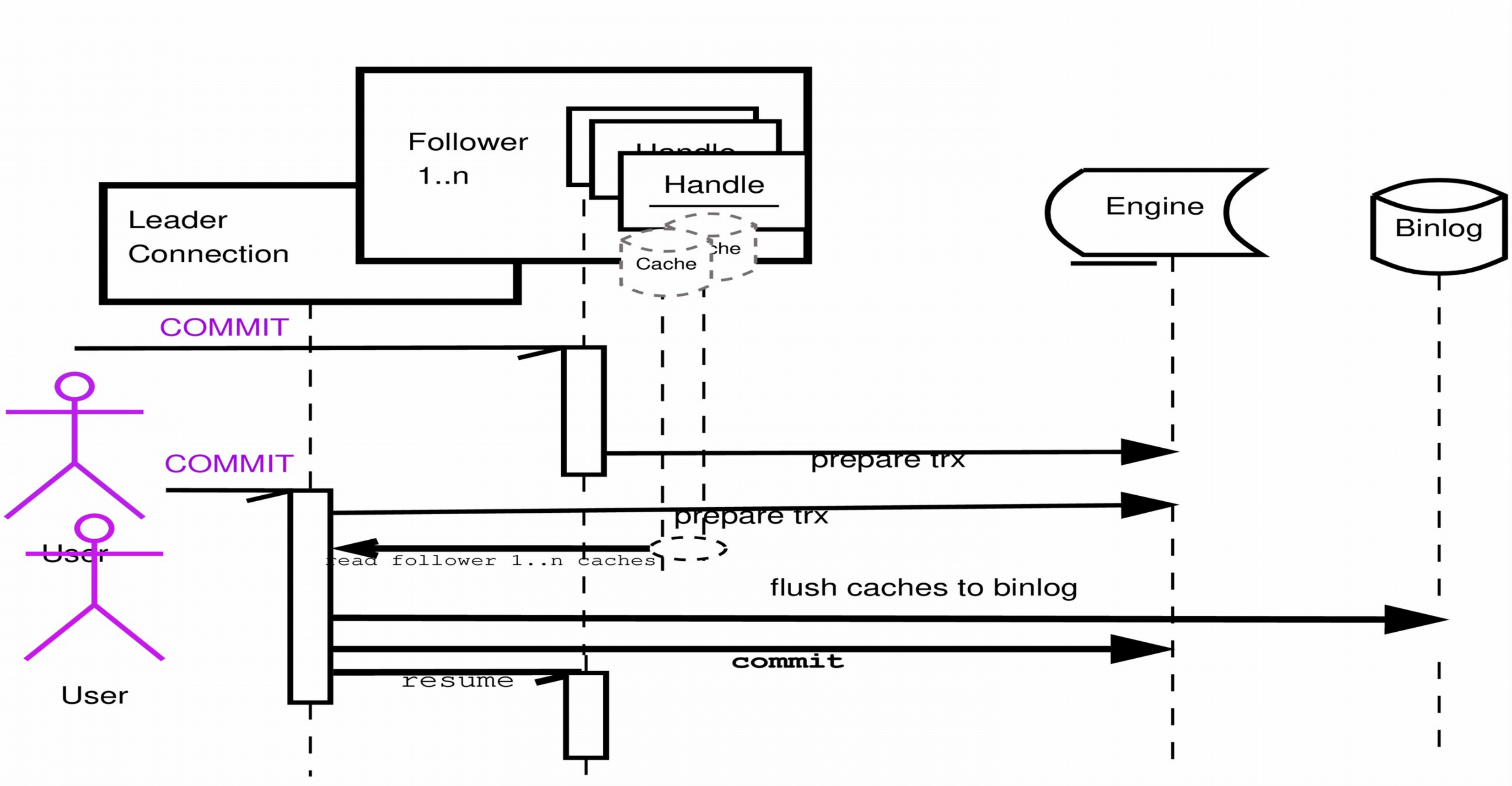
Transaction state transaction



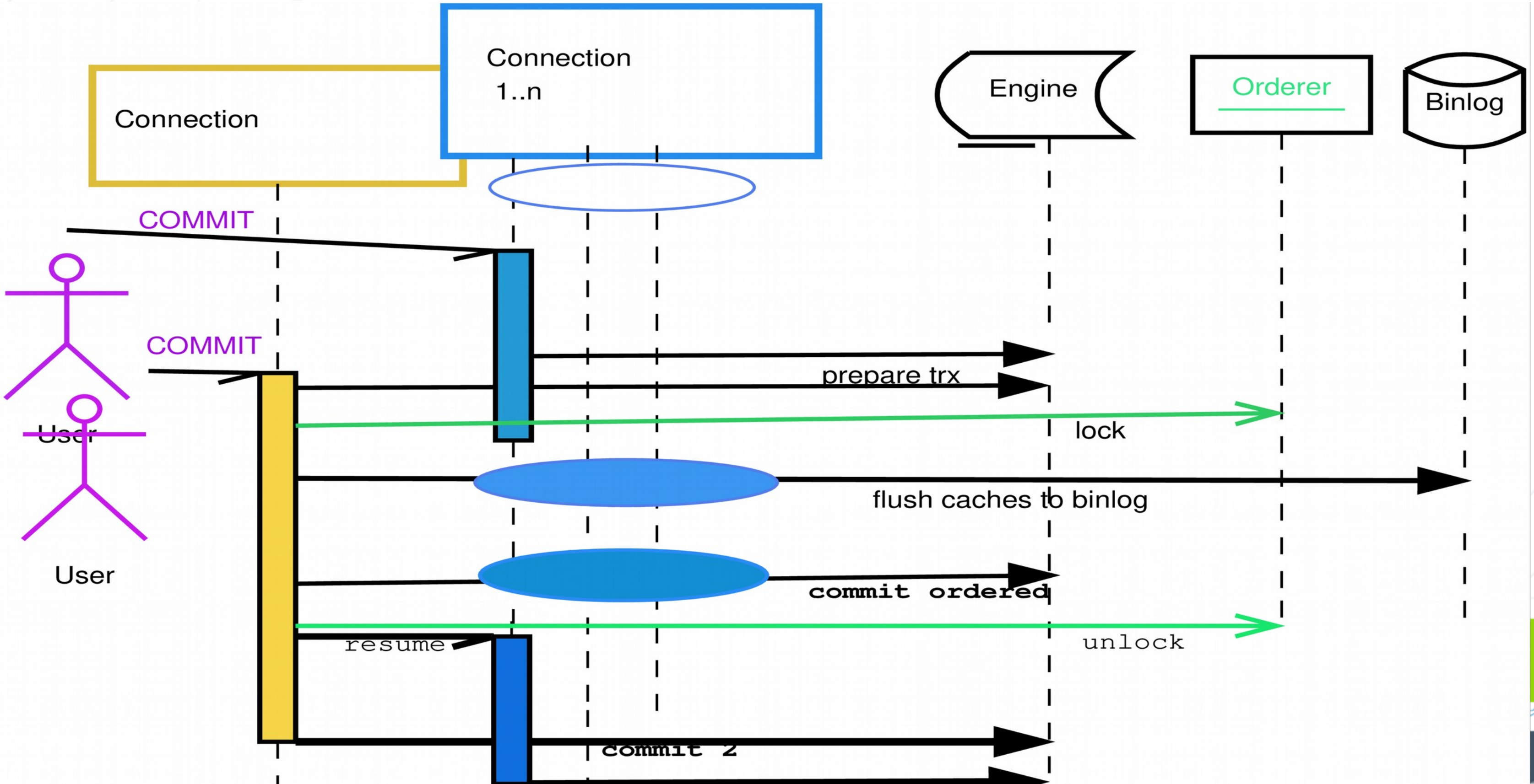
Binary logging



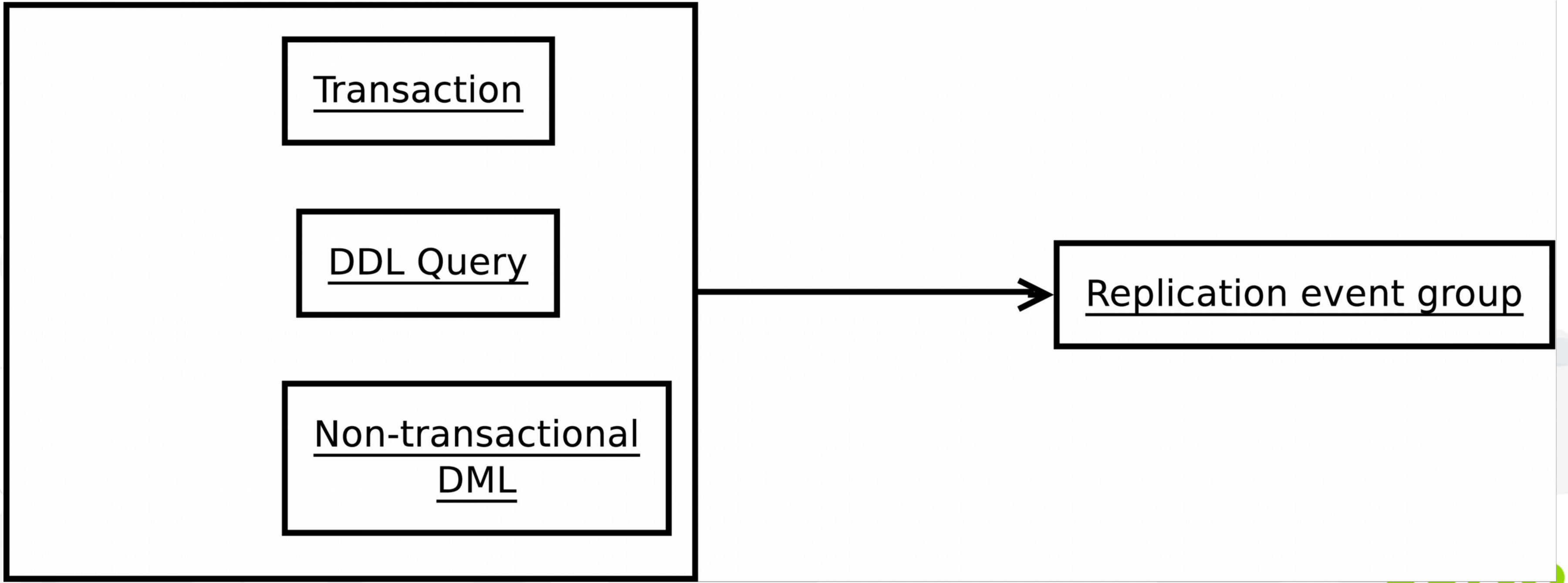
Binlog group commit



Binlog Group Commit ordered



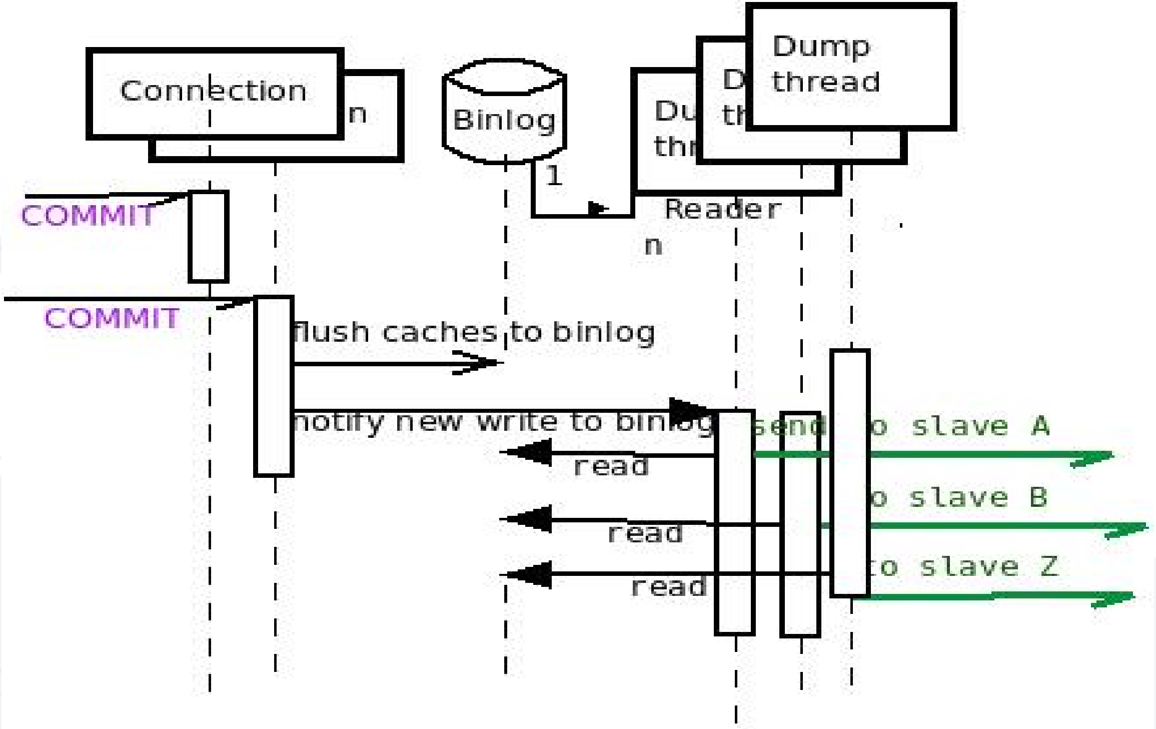
binary logging: event group



event group: example

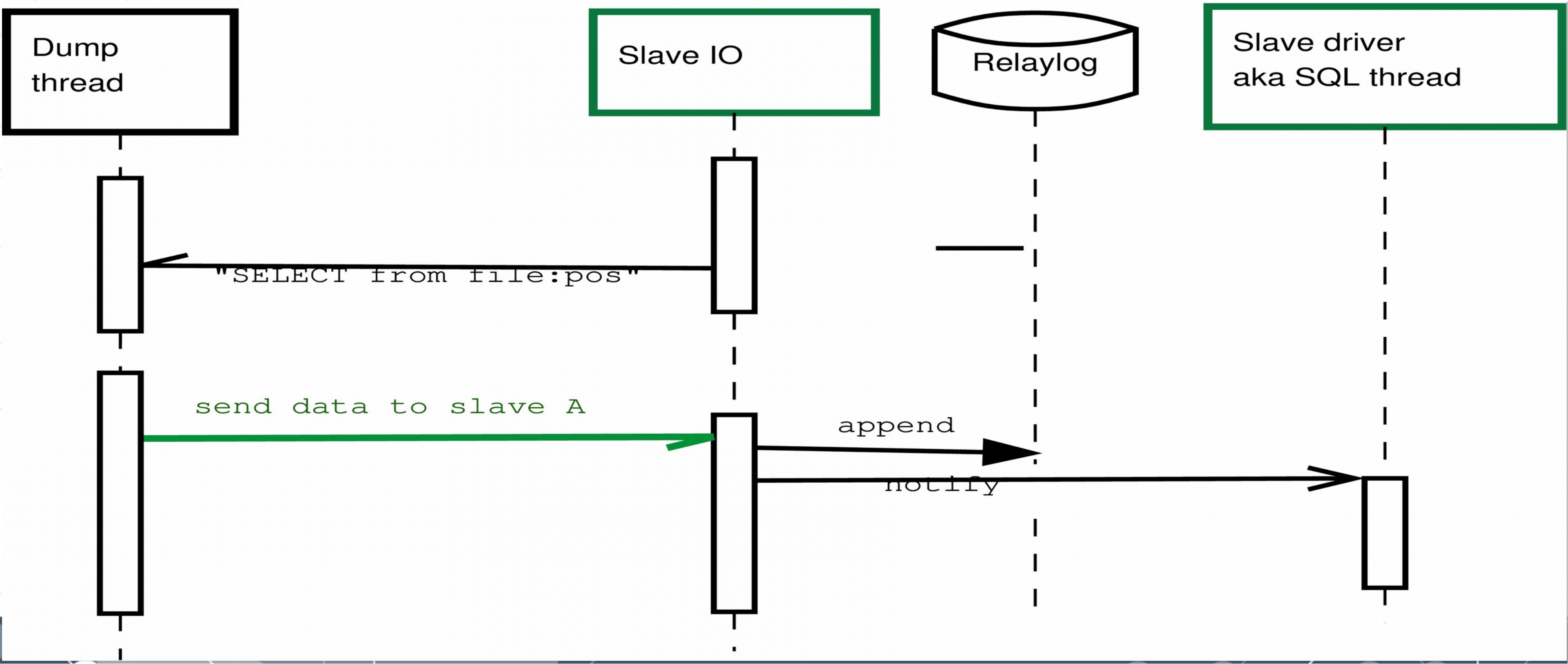
```
# at 1283
#180223 19:37:35 server id 1  end_log_pos 1325 CRC32 0x0889dcdf  GTID 11-1-1 trans
/*!100001 SET @@session.gtid_domain_id=11*//*!*/;
/*!100001 SET @@session.gtid_seq_no=1*//*!*/;
BEGIN
/*!*/;
# at 1325
#180223 19:37:35 server id 1  end_log_pos 1414 CRC32 0xf1ca2d36  Query thread_id=9 \
                                                                 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 0x21121803   Xid = 42
COMMIT/*!*/;
```

Events shipping: Dump thread

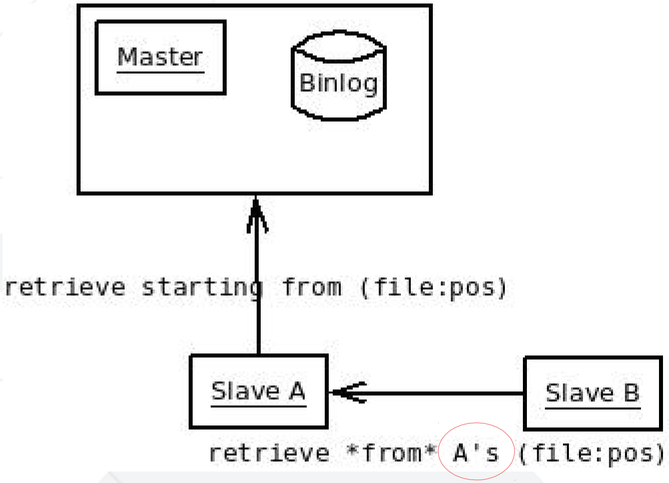


M|18 MariaDB

Events receiving: IO thread



No failover without ...



Master

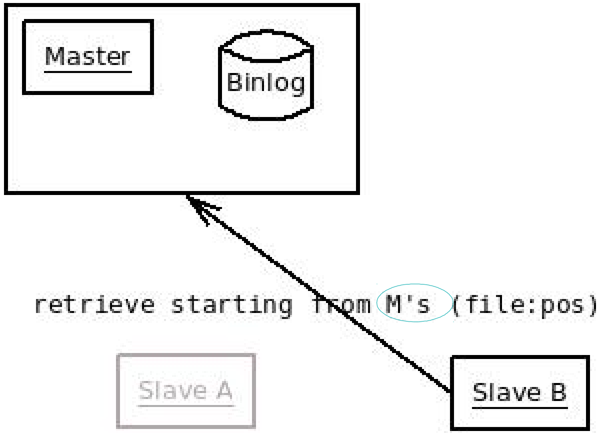
Binlog

retrieve starting from (file:pos)

Slave A

Slave B

retrieve *from* A's (file:pos)



Master

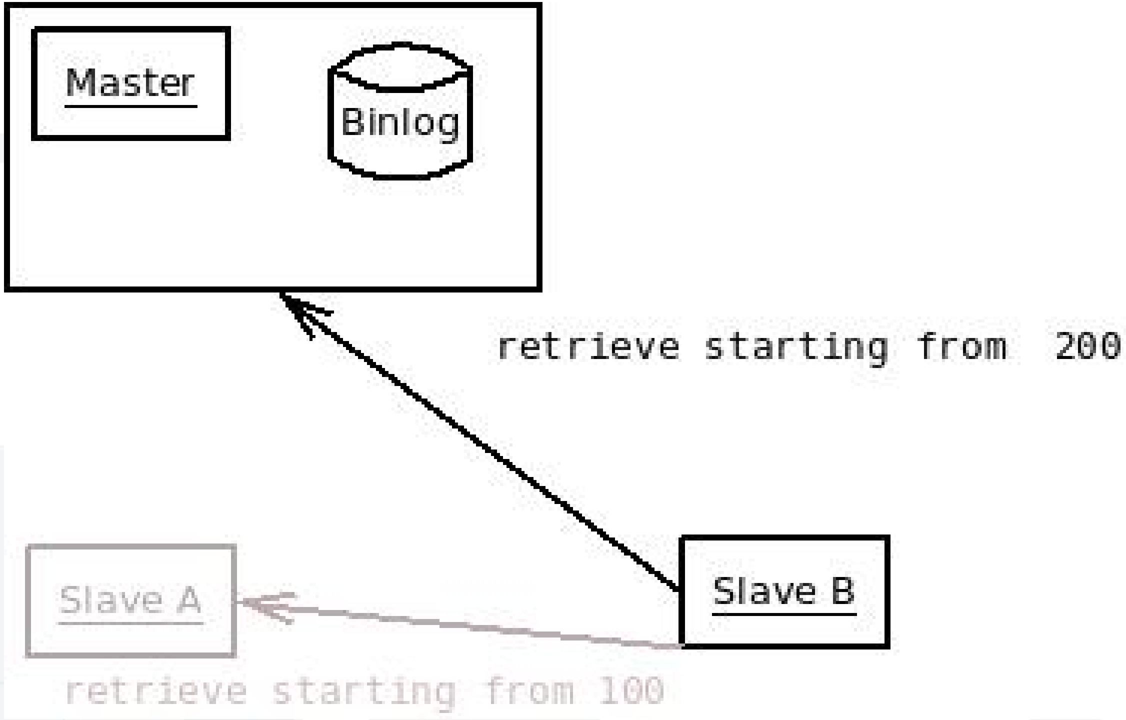
Binlog

retrieve starting from M's (file:pos)

Slave A

Slave B

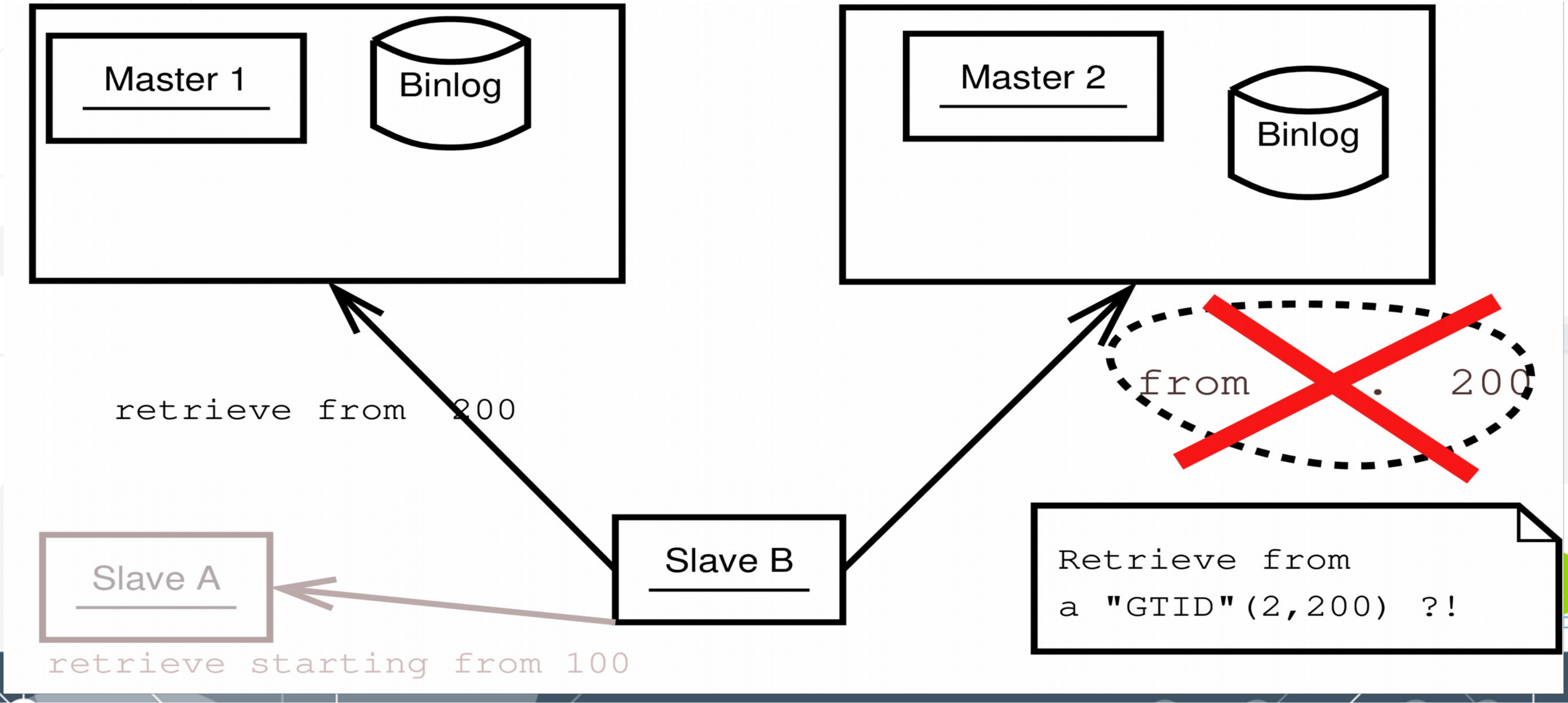
GTID idea



retrieve starting from 200

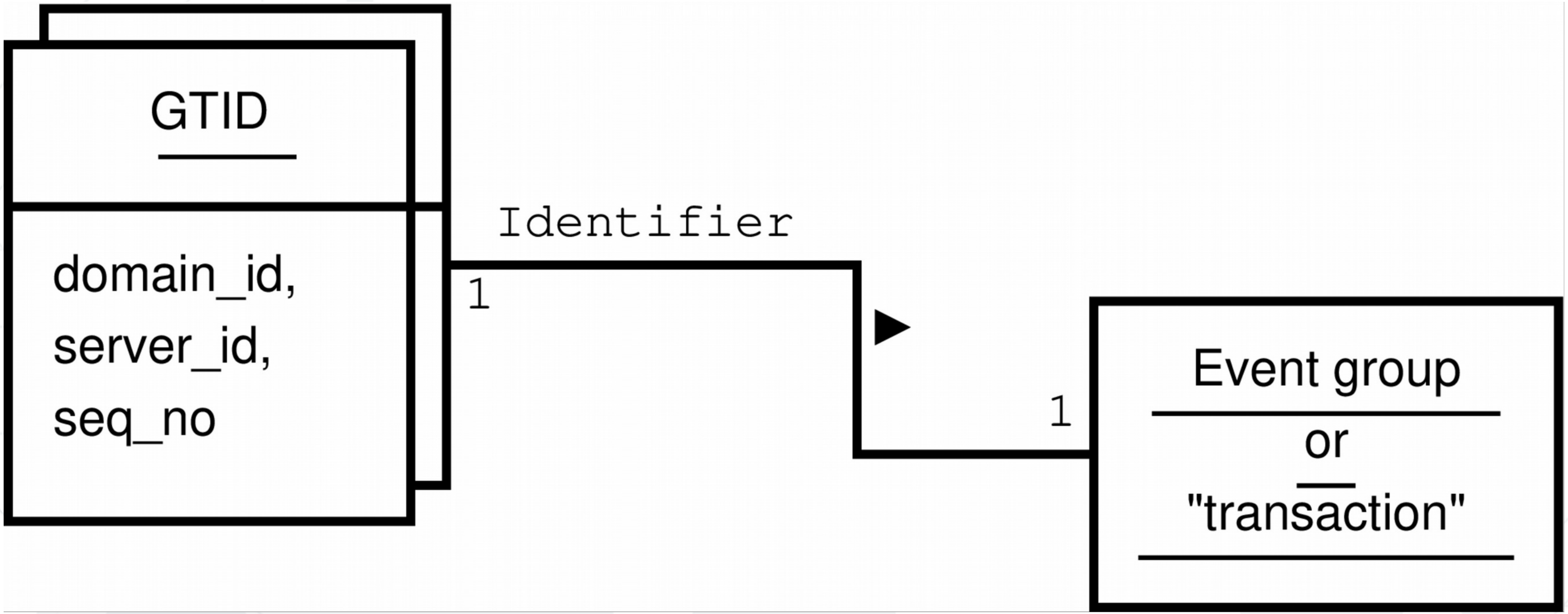
retrieve starting from 100

GTID: required in Multi-Source replication



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

GTID definition



GTID

domain_id,
server_id,
seq_no

Identifier

1

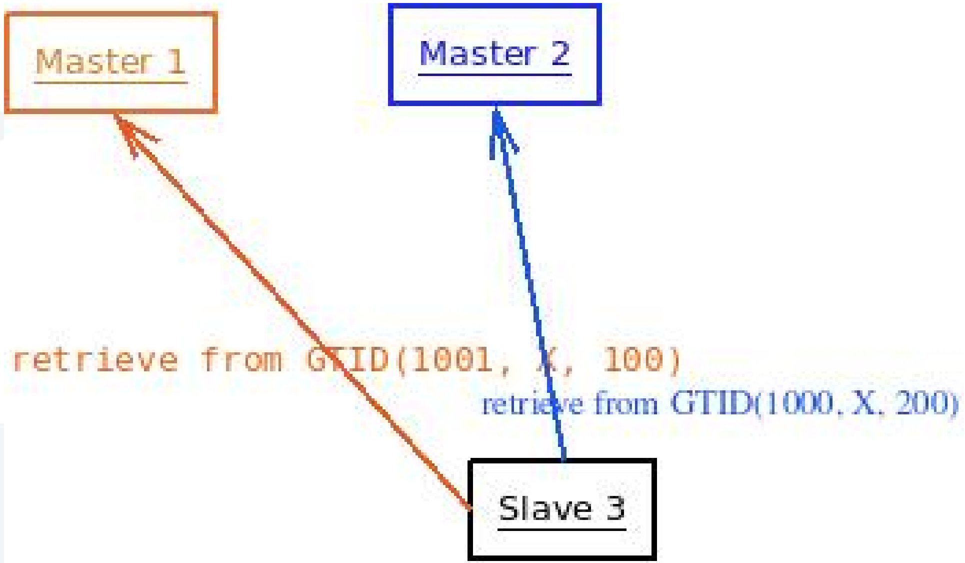
1

Event group

or

"transaction"

GTID and Multi-Sourced Replication



Master 1

Master 2

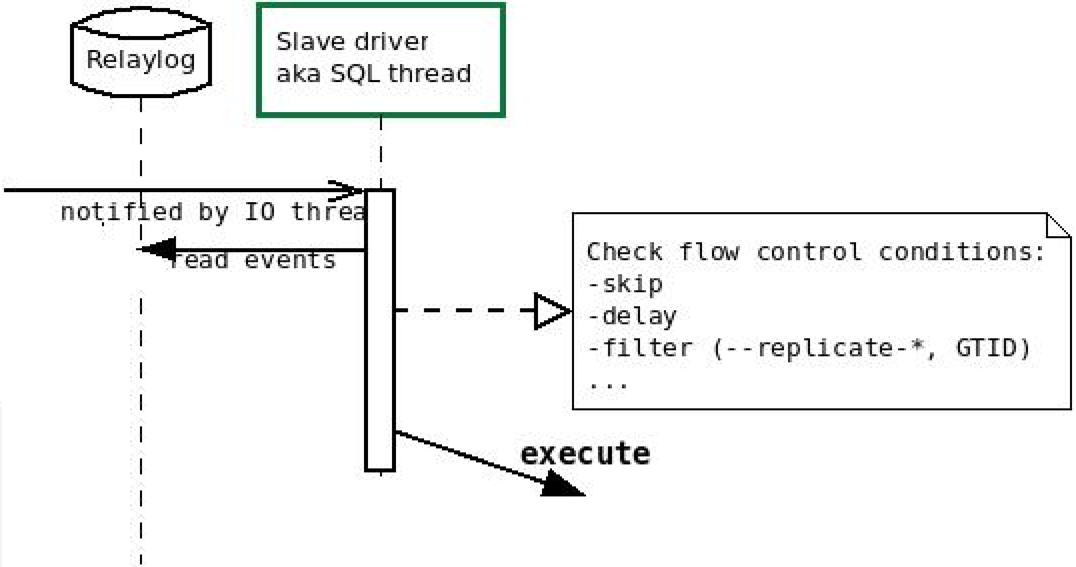
retrieve from GTID(1001, X, 100)

retrieve from GTID(1000, X, 200)

Slave 3

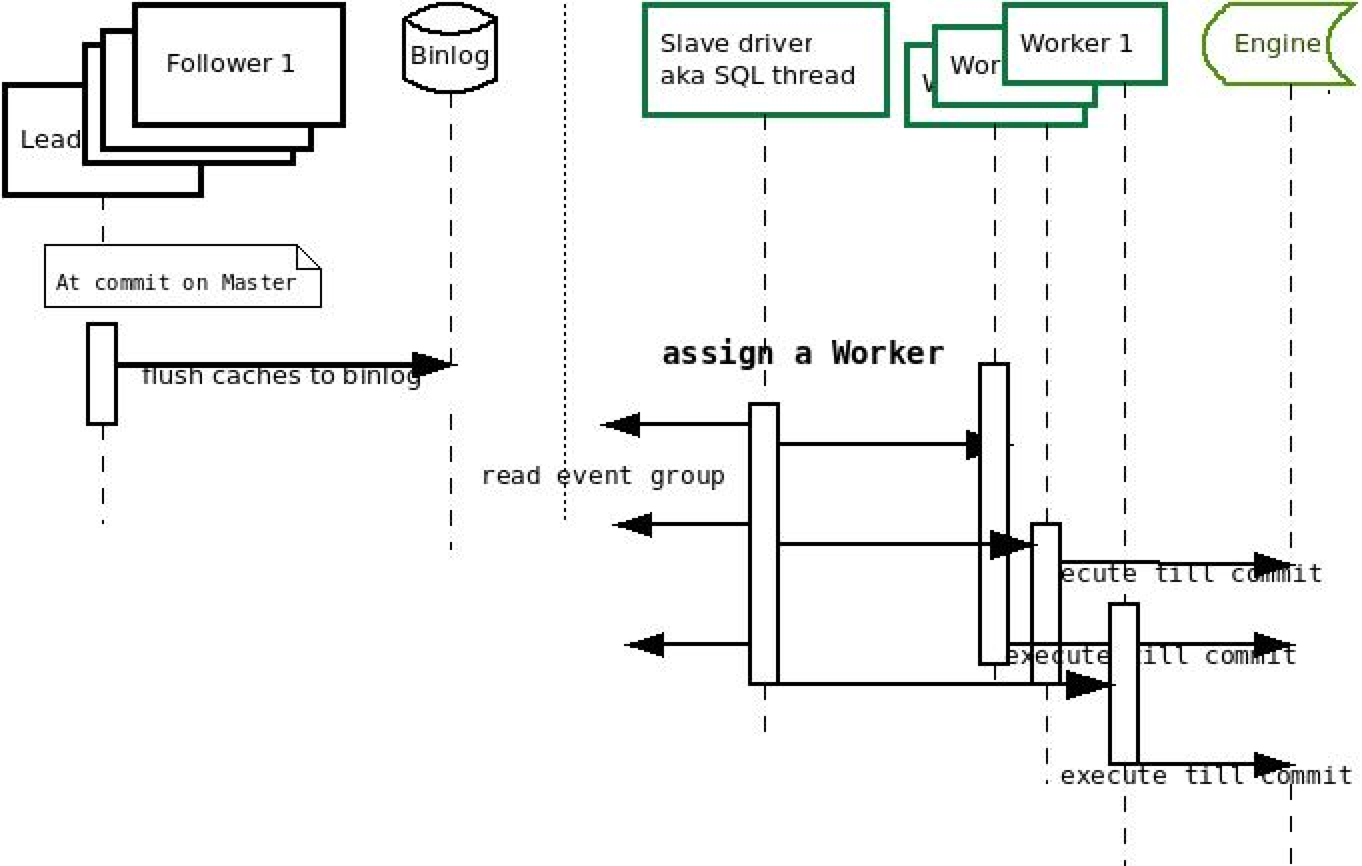
Transactions from different domains are executed independently

Events execution on slave: single-threaded mode

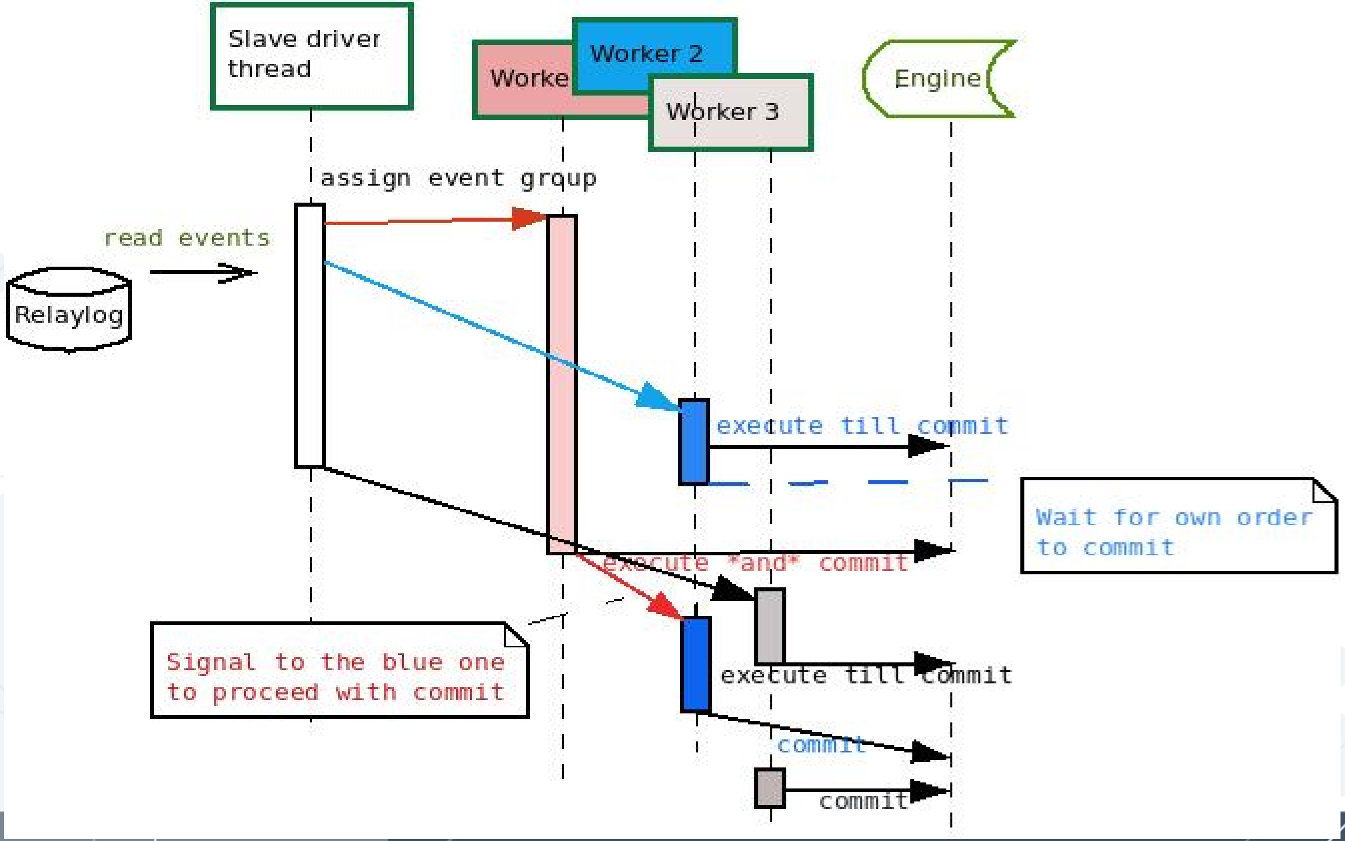


Check flow control conditions:
-skip
-delay
-filter (--replicate-*, GTID)
...

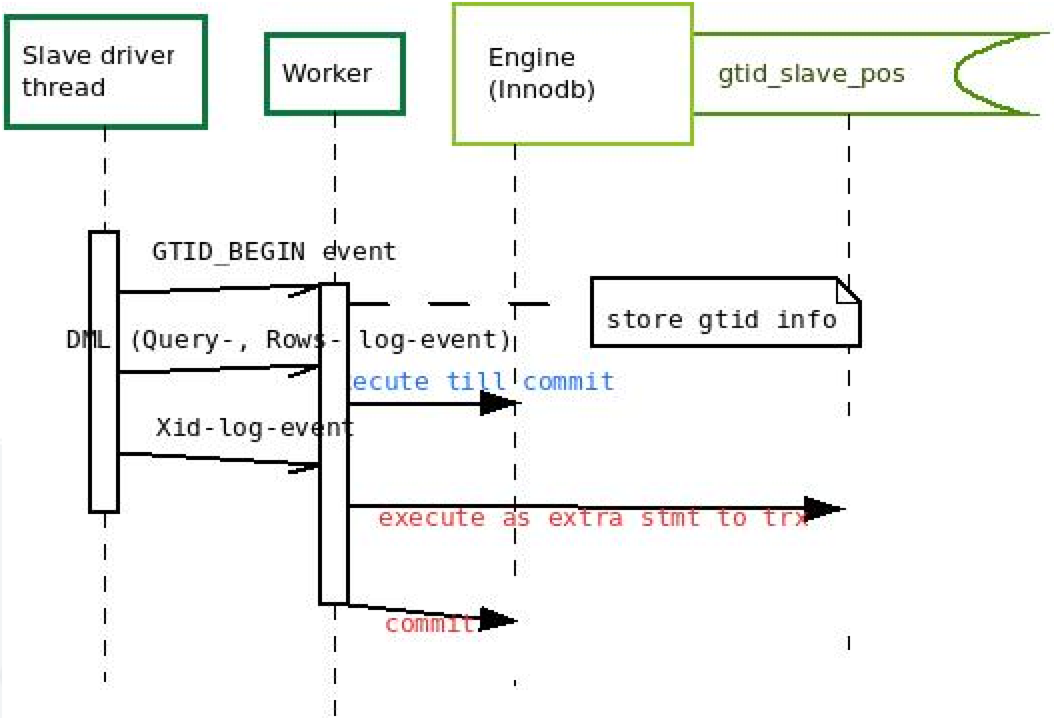
Events execution: parallel scheduling



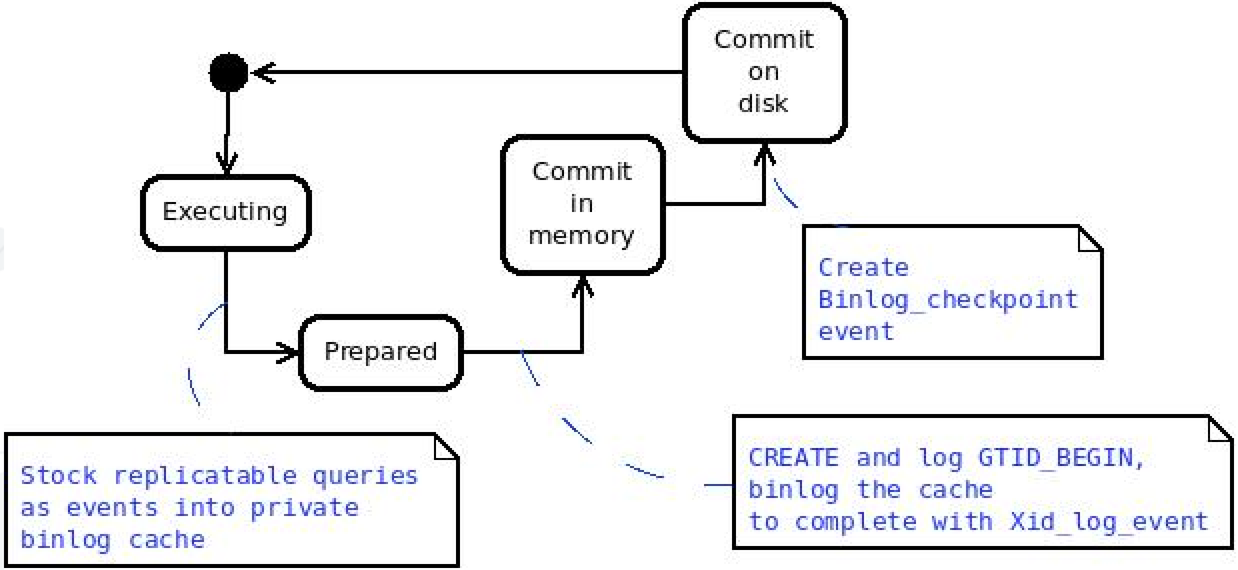
Events execution: ordered commit



GTID execution: gtid_slave_pos table



Recovery: committed data may be lost

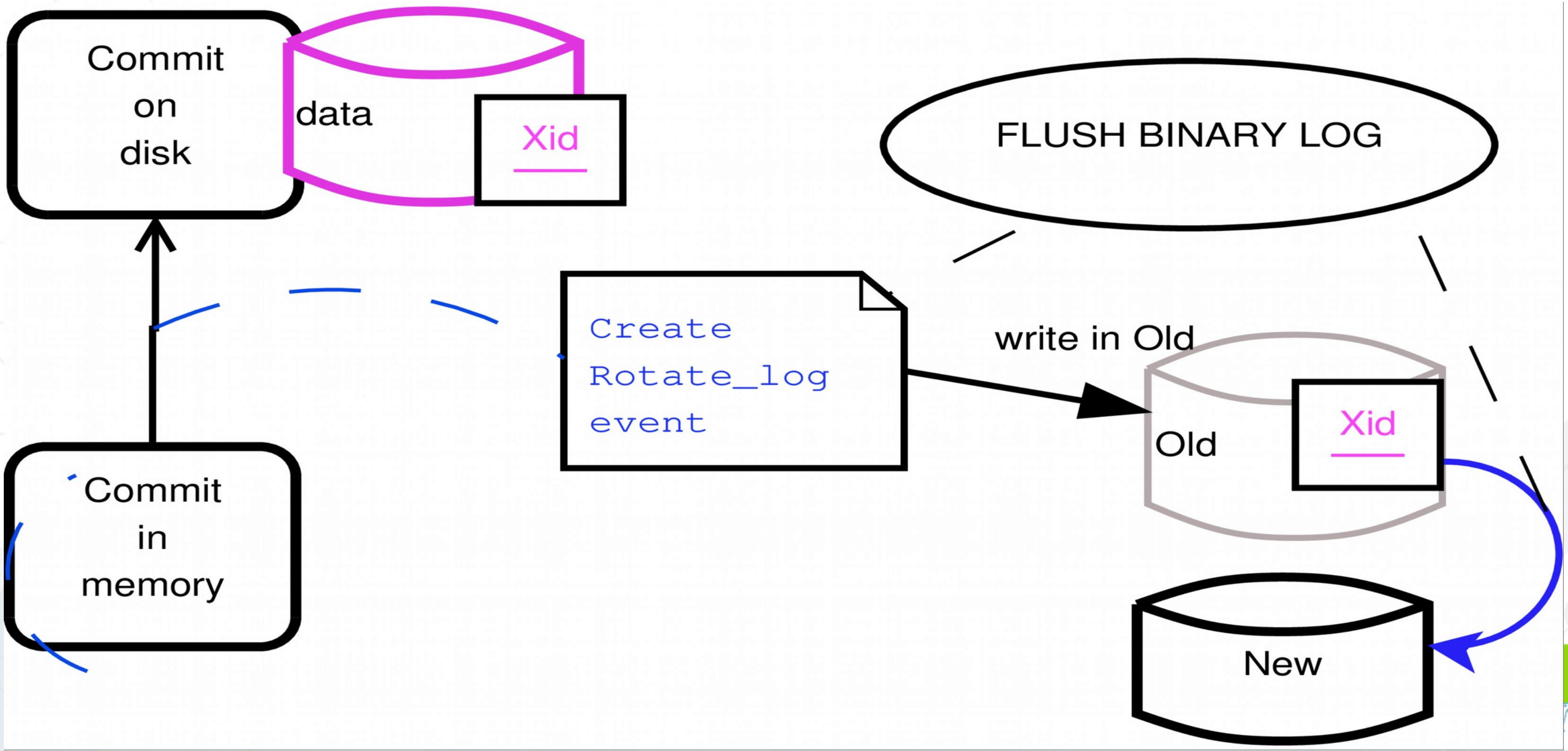


Create Binlog_checkpoint event

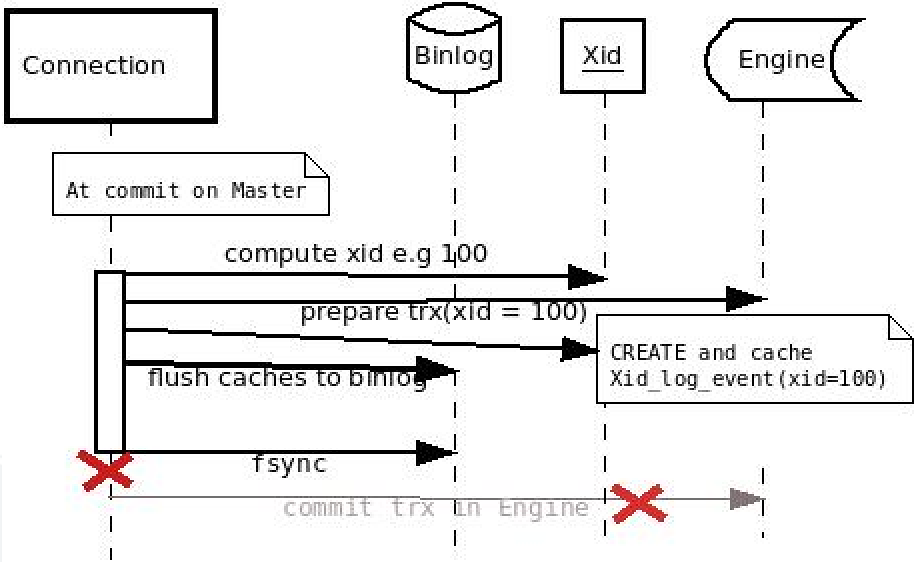
Stock replicatable queries as events into private binlog cache

CREATE and log GTID_BEGIN, binlog the cache to complete with Xid_log_event

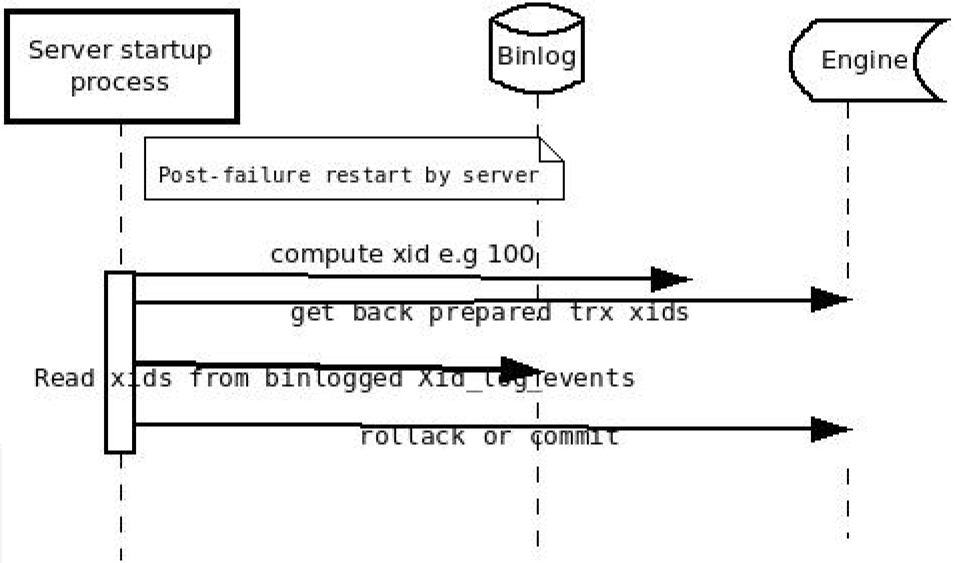
Recovery without Binlog_checkpoint



Recovery: server crash



Recovery: post-restart decisions



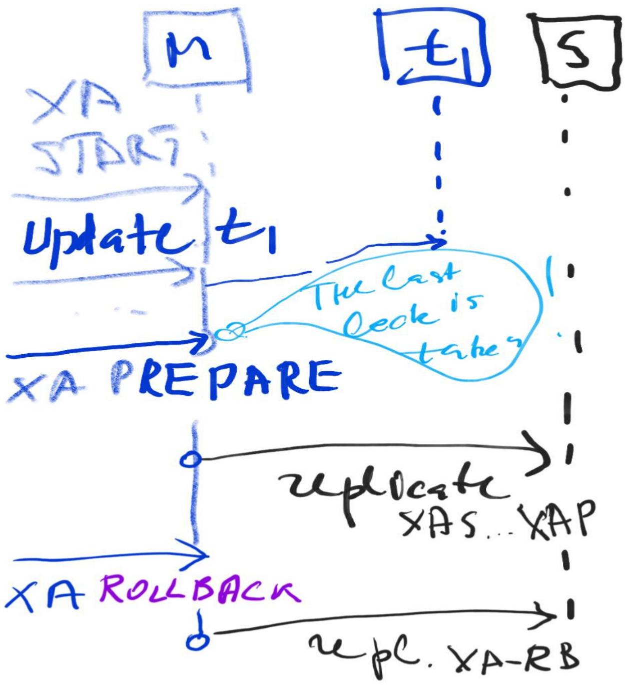
Server startup process
Binlog
Engine
Post-failure restart by server
compute xid e.g 100
get back prepared trx xids
Read xids from binlogged Xid_log_events
rollack or commit

Ongoing and perspective projects

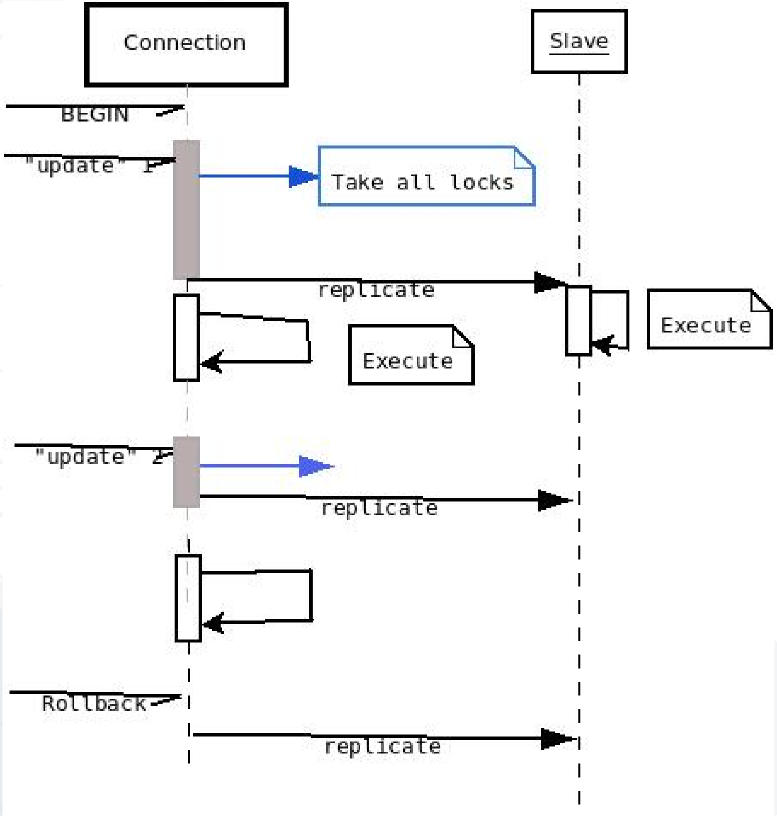
1) XA replication MDEV-7974
2) Eager replication as follow-up XA replication
3) Parallel slave group commit
4) Relay-log-less slave
5) Committed GTID tracker by engine
6) Binlog-less “relay” slave in chain replication
7) Consensus protocol on membership in replication configuration
8) E.g Paxos-like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

Ongoing: XA replication MDEV-7974

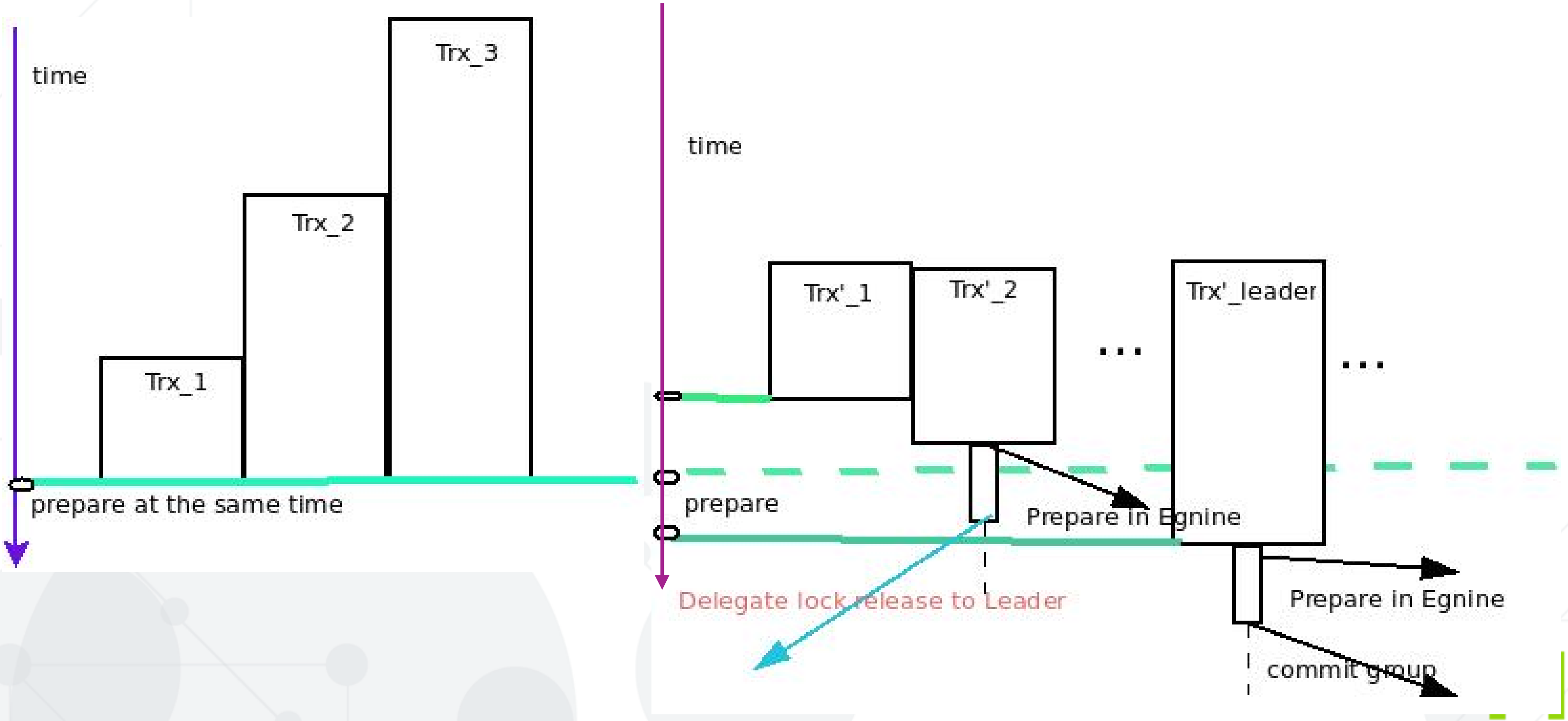
XA replication and its follow-up



Follow-ups: active replication



MDEV-16404: slave group commit



References:

https://mariadb.com/kb/en/library/replication-overview/
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