facebook

facebook

Global Transaction ID at Facebook

Santosh Praneeth Banda, Evan Elias, and Yoshinori Matsunobu

Agenda

Introduction to GTID

GTID Concepts

- Traditional MySQL replication uses relative coordinates
  - Replication position is entirely relative to the current master
- With GTID a unique identifier is assigned to every transaction
  - Each instance tracks which GTIDs they have executed
  - Relative coordinates are also still available
- Each transaction with a given GTID will only be executed once per server

MySQL 5.6 Implementation

- Each master has a UUID
- Each GTID looks like source_uuid:transaction_id
- transaction_id is monotonically increasing per source
- Say server A has UUID `3E11FA47-71CA-11E1-9E33-C80AA9429562`, then GTID `3E11FA47-71CA-11E1-9E33-C80AA9429562:23` indicates the 23rd write transaction to originate here
- Transaction counter ordering is per source, not global!
    - If we then promote server B, its transaction IDs begin at 1
    - If we promote back to A, its transaction IDs resume at 24
    - Binlog is the source of truth for replaying transactions in order

MySQL 5.6 GTID sets

- Servers track which range of GTIDs they’ve executed
  - gtid_executed global variable
  - Also available in SHOW MASTER STATUS and SHOW SLAVE STATUS
  - Example: `3E11FA47-71CA-11E1-9E33-C80AA9429562:1-5`
- Server also keeps track of set of GTIDs in binlogs that have been purged

MySQL 5.6 auto-positioning

- CHANGE MASTER TO … MASTER_AUTO_POSITION = 1
- Omit MASTER_LOG_FILE and MASTER_LOG_POS
- Scans master's binlogs to find transactions that have not yet been executed on this replica
- If the replica is missing transactions that have already been purged from master's binlogs, I/O thread will error

Other implementations

- Google’s MySQL 5.0 GTID patch, by Justin Tolmer
  - Supported auto-positioning and crash-safe replication
  - Tracks latest GTID instead of using GTID sets
  - Didn’t support master-master topologies

- MariaDB 10.x GTID support
  - Tracks latest GTID instead of using GTID sets
  - Uses a notion of “domain ID” to support master-master topologies
  - Permits online rollout
  - Commands and syntax all differ from Oracle’s implementation

Benefits and Use-cases

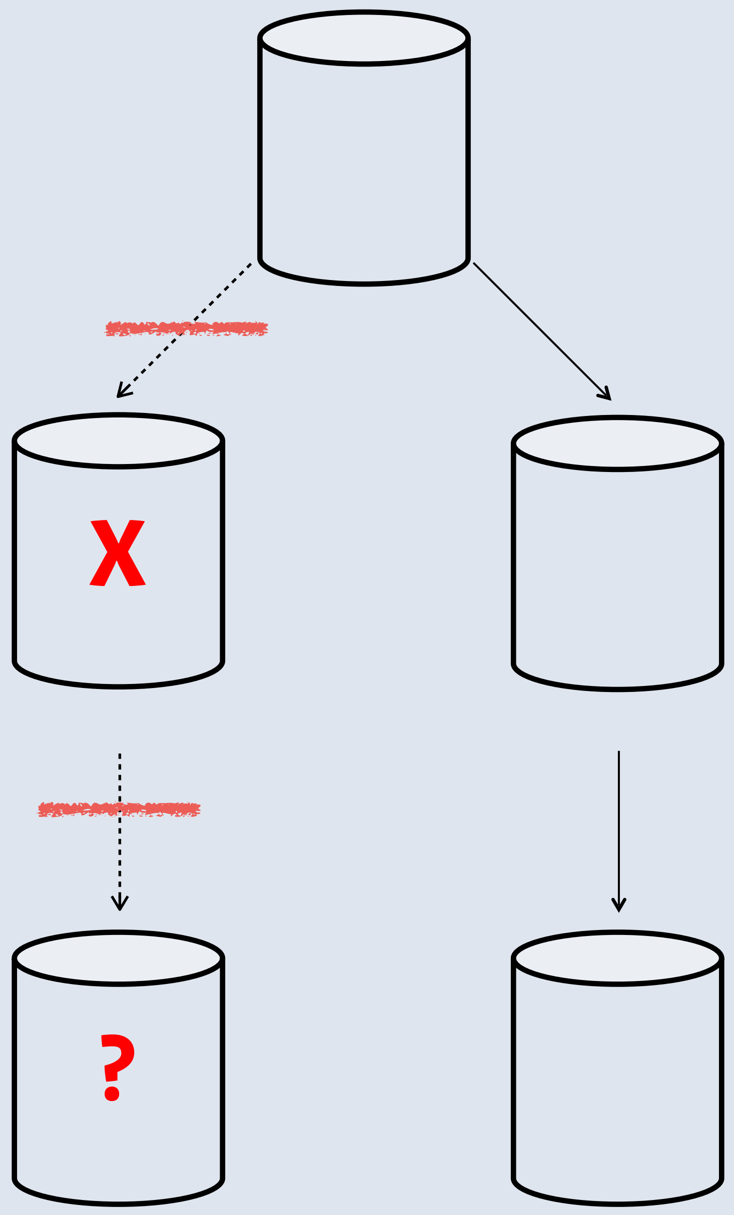
Major areas of benefit

- Failover
  - Simpler, faster, less error-prone
  - Trivial to get slaves in sync after master failure
  - Server will safely ignore transactions it has already executed
- Backups
  - Cornerstone for point-in-time recovery from a single binlog stream
  - No need for duplicate streams per replica
- Hierarchical replication
  - Slaves-of-slaves are now much easier to manage

Hierarchical replication chains

- Previously, if middle tier instance fails, no easy way to repoint its bottom-level replicas to another master

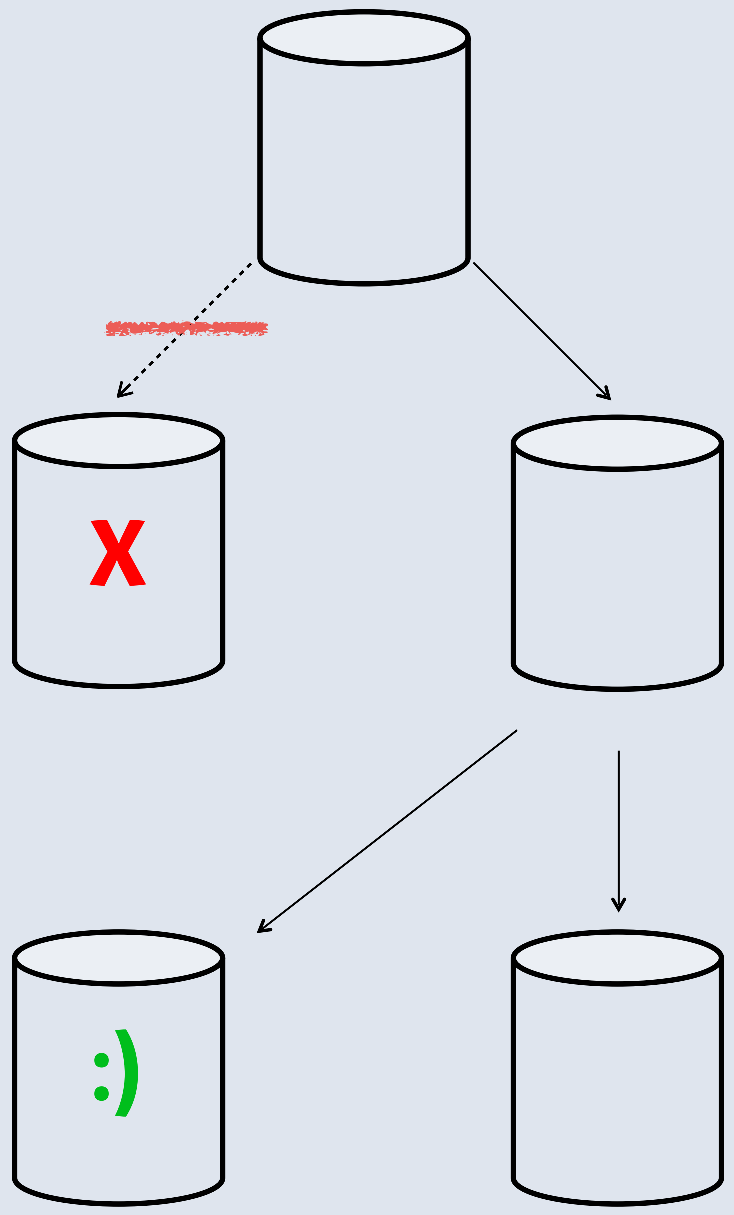
- Replication coordinates were always relative to the slave's immediate master!



Hierarchical replication chains

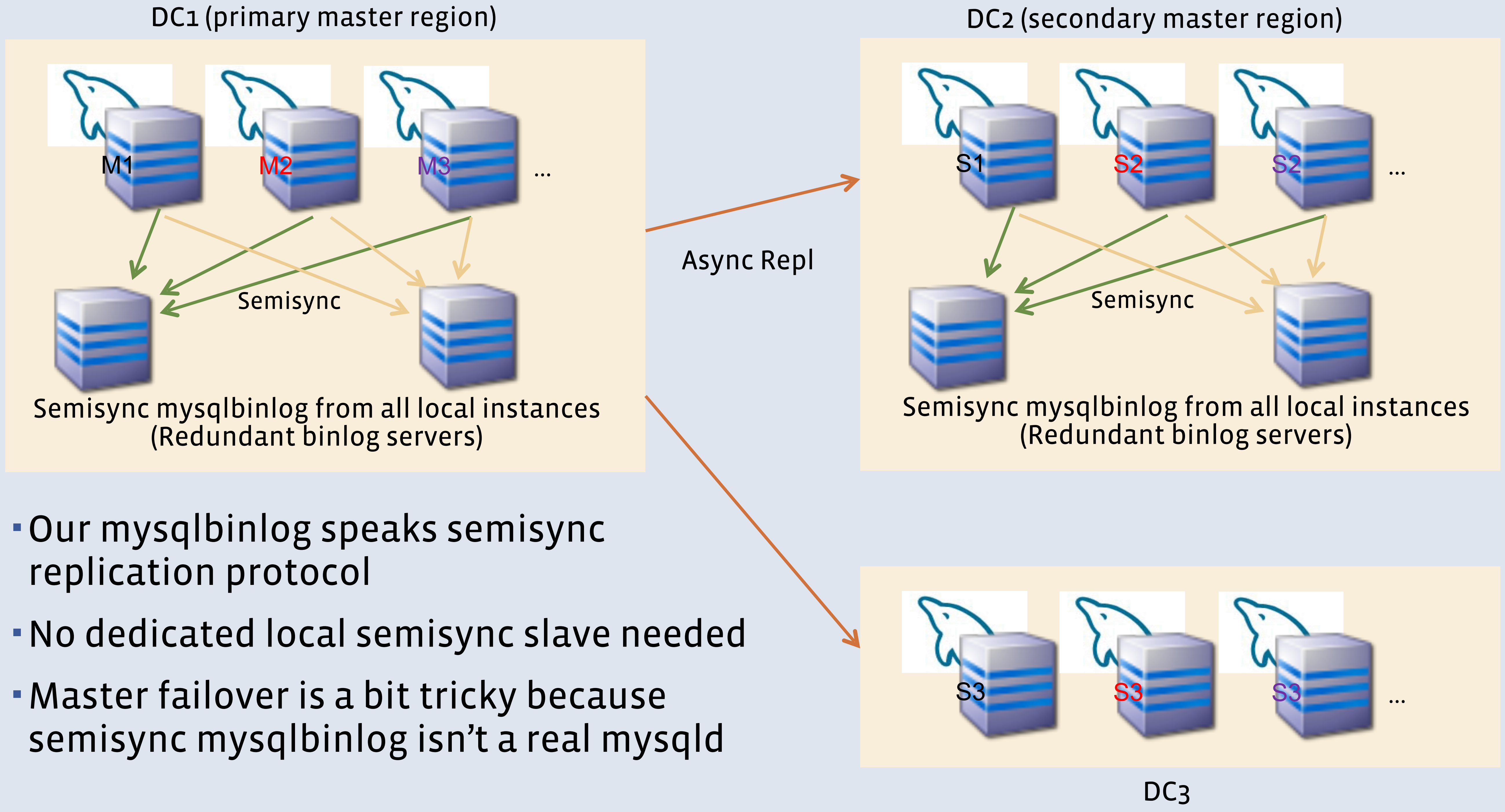
- GTID auto-positioning fixes this. Trivial to repoint any member of the replica set to any other.

- Hierarchical replication not currently used at Facebook, but perhaps we may find a use-case in the future



Failover, semi-sync, and MHA

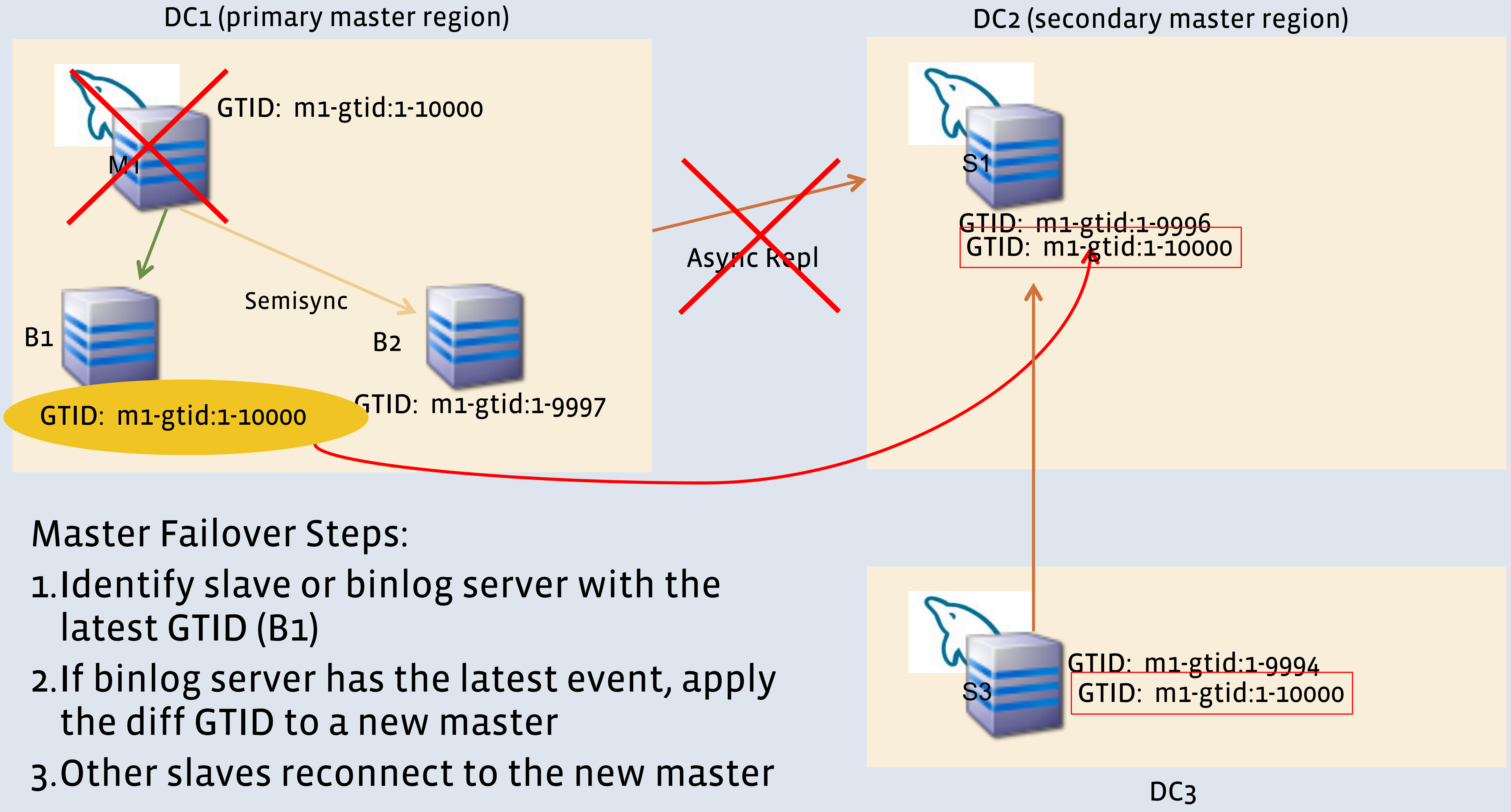
GTID and Loss-Less Semisync mysqlbinlog



DC1 (primary master region)
M1
M2
M3
...
Semisync
Semisync mysqlbinlog from all local instances
(Redundant binlog servers)
Async Repl
DC2 (secondary master region)
S1
S2
S2
...
Semisync
Semisync mysqlbinlog from all local instances
(Redundant binlog servers)
S3
S3
S3
...
DC3

- Our mysqlbinlog speaks semisync replication protocol
- No dedicated local semisync slave needed
- Master failover is a bit tricky because semisync mysqlbinlog isn't a real mysqld

Master failover with GTID + mysqlbinlog



Master Failover Steps:

1. Identify slave or binlog server with the latest GTID (B1)
2. If binlog server has the latest event, apply the diff GTID to a new master
3. Other slaves reconnect to the new master

Extending MHA for GTID and binlog server

- MHA is a fast master failover tool, working with official MySQL 5.0~5.6, even without GTID
- We use semisync binlog servers, which don't accept MySQL commands
- Official tool mysqlfailover covers master failover with GTID, but it doesn't cover binlog servers
- We extended MHA to support both GTID and binlog servers

- https://code.google.com/p/mysql-master-ha/

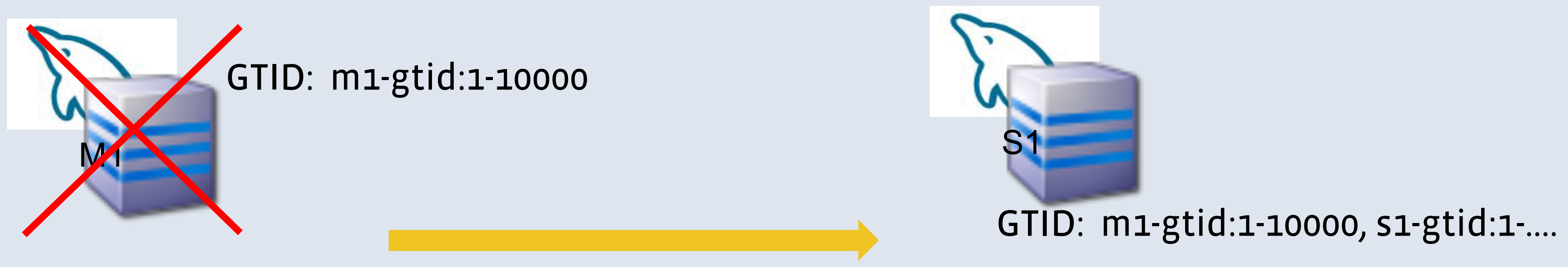
MHA and GTID failover

- MHA (Version 0.56) automatically detects whether to do GTID based failover
- GTID-based failover process:
  - Checking if binlog servers are available or not
  - Checking the latest GTID position
  - Recovering based on GTID (not using relay logs)

Crash-safe master via loss-less semisync

What is crash-safe master?

- When master is down and promoting a slave, after the crashed master's recovery, we want to add the crashed master as a new slave without rebuilding the whole crashed master instance
- Recovery (recovering from OS reboot etc) shouldn't take days. Applying a few hours of binlogs is much faster than rebuilding entire instance



After crashed master's recovery, continue replication by
CHANGE MASTER TO MASTER_HOST='S1', MASTER_AUTO_POSITION=1;

Extensions needed for crash-safe master

- Using InnoDB only
- Writing GTID to InnoDB (logfile or table) at transaction commit
- GTID and InnoDB must be consistent. Writing GTID to InnoDB logfile/ table guarantees this
- Using loss-less semisync (backporting from 5.7 semisync)
- Crashed master's binlog position must not be ahead any other slave (binlog reader). Loss-less semisync guarantees this.

FB MySQL Server improvements

Facebook GTID improvements

- Simplified deployment
- Restart and slave connect performance improvements
- Crash-safe slave
- GTID-incompatible statement counters

- Source code available on GitHub: https://github.com/facebook/mysql-5.6/commits/webscalesql-5.6.16-47

Simplify GTID deployment

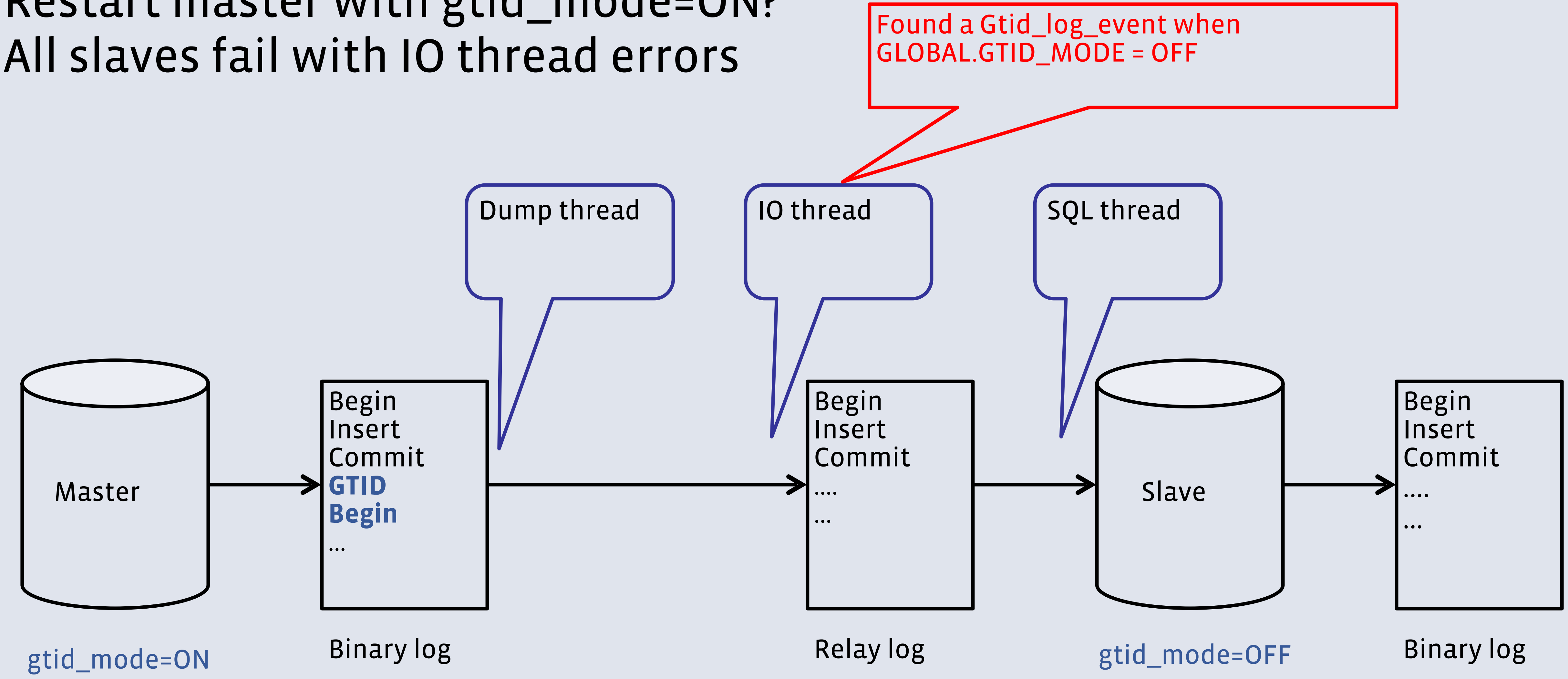
Why?

- No reasonable deployment plan exists in MySQL 5.6
- In MySQL documentation: Synchronize all the servers after stopping all the write traffic on master, restart all the servers in a replica cluster with gtid_mode=ON. This is not a viable option at all
- Need a smooth rollout plan to reduce downtime of services

Simplify GTID deployment

Try?

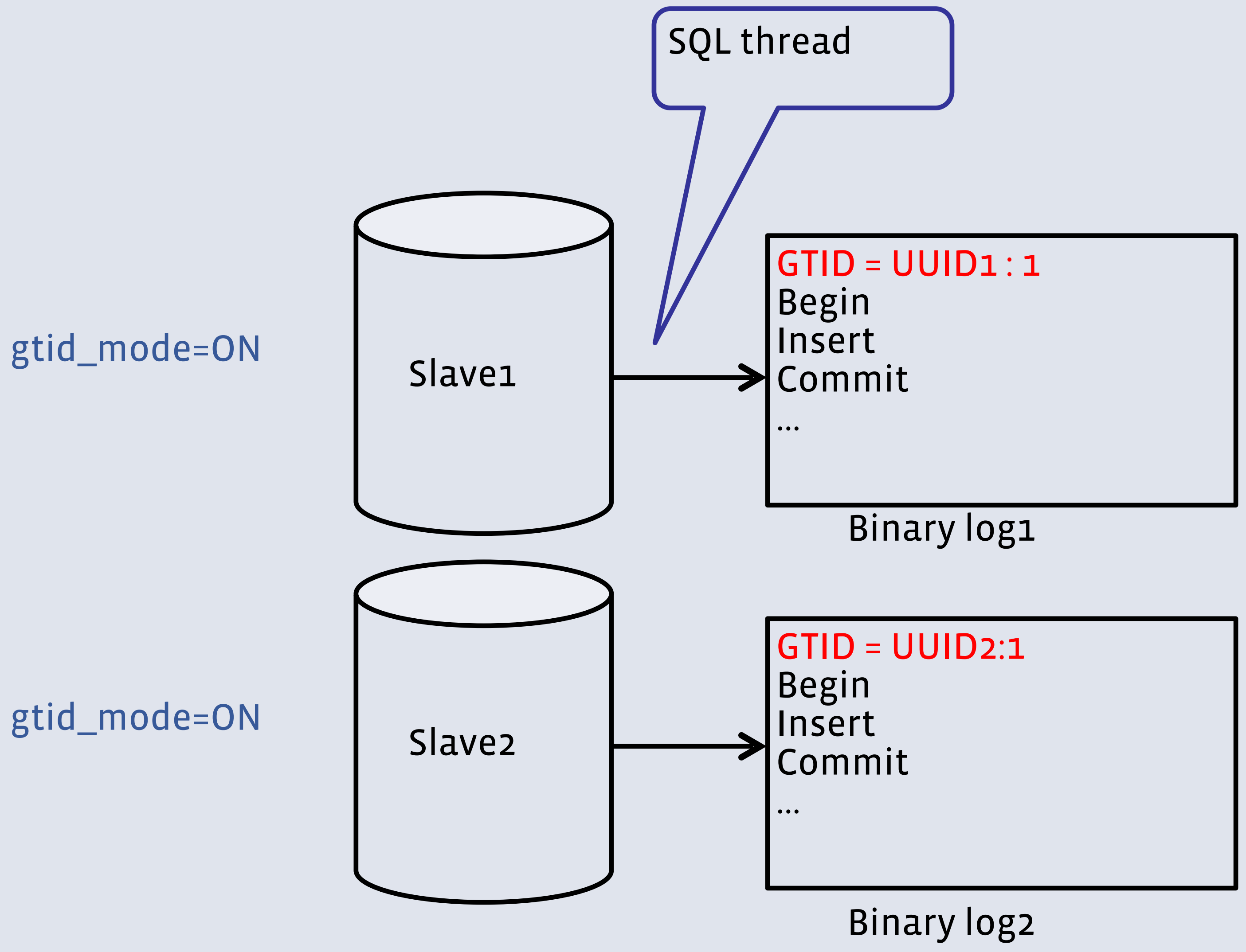
Restart master with gtid_mode=ON?
All slaves fail with IO thread errors



Simplify GTID deployment

Try?

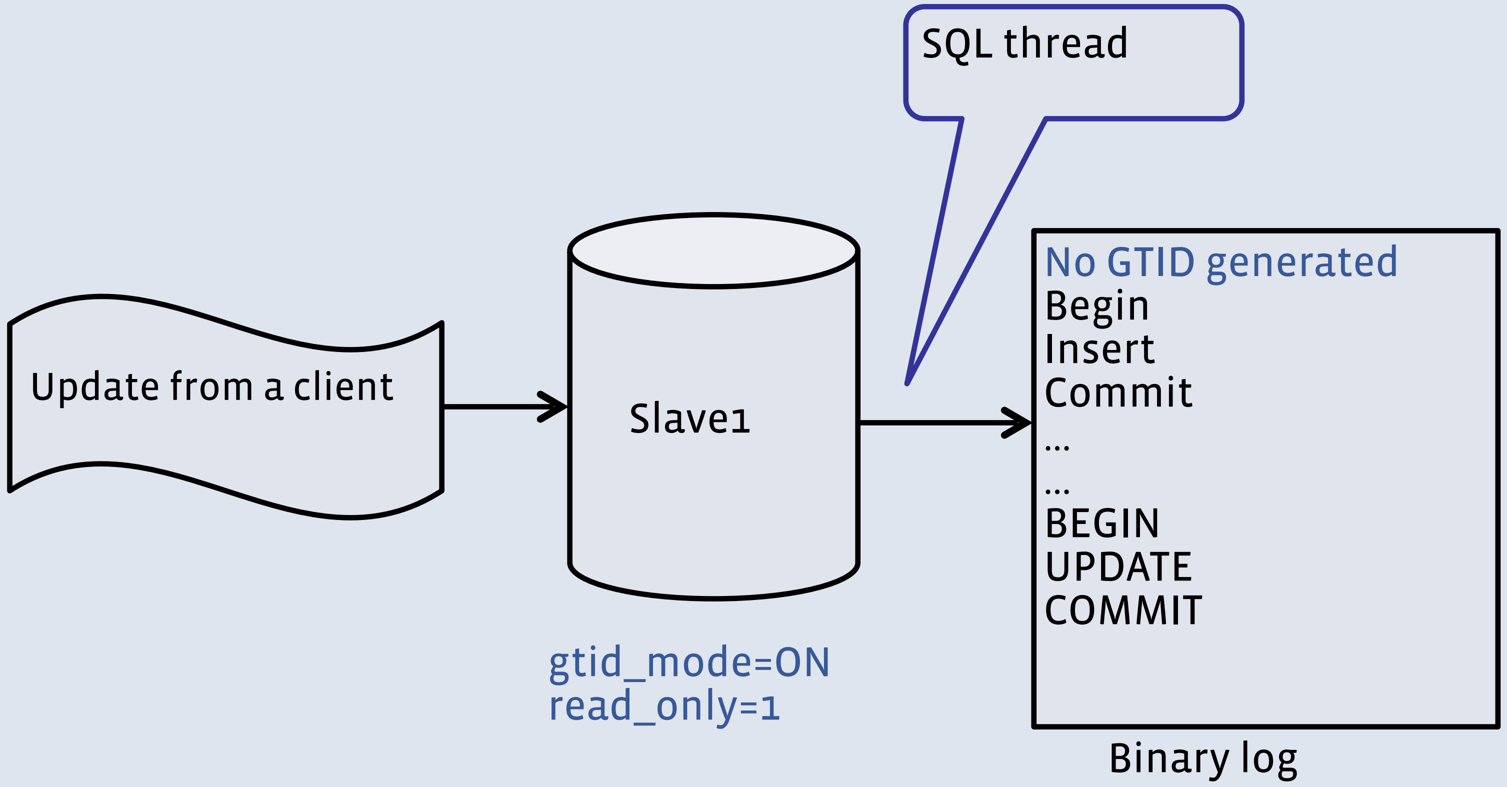
Restart slaves with --gtid_mode=ON? Gtid_executed will be different on each slave breaking replication with GTID protocol



Simplify GTID deployment

How?

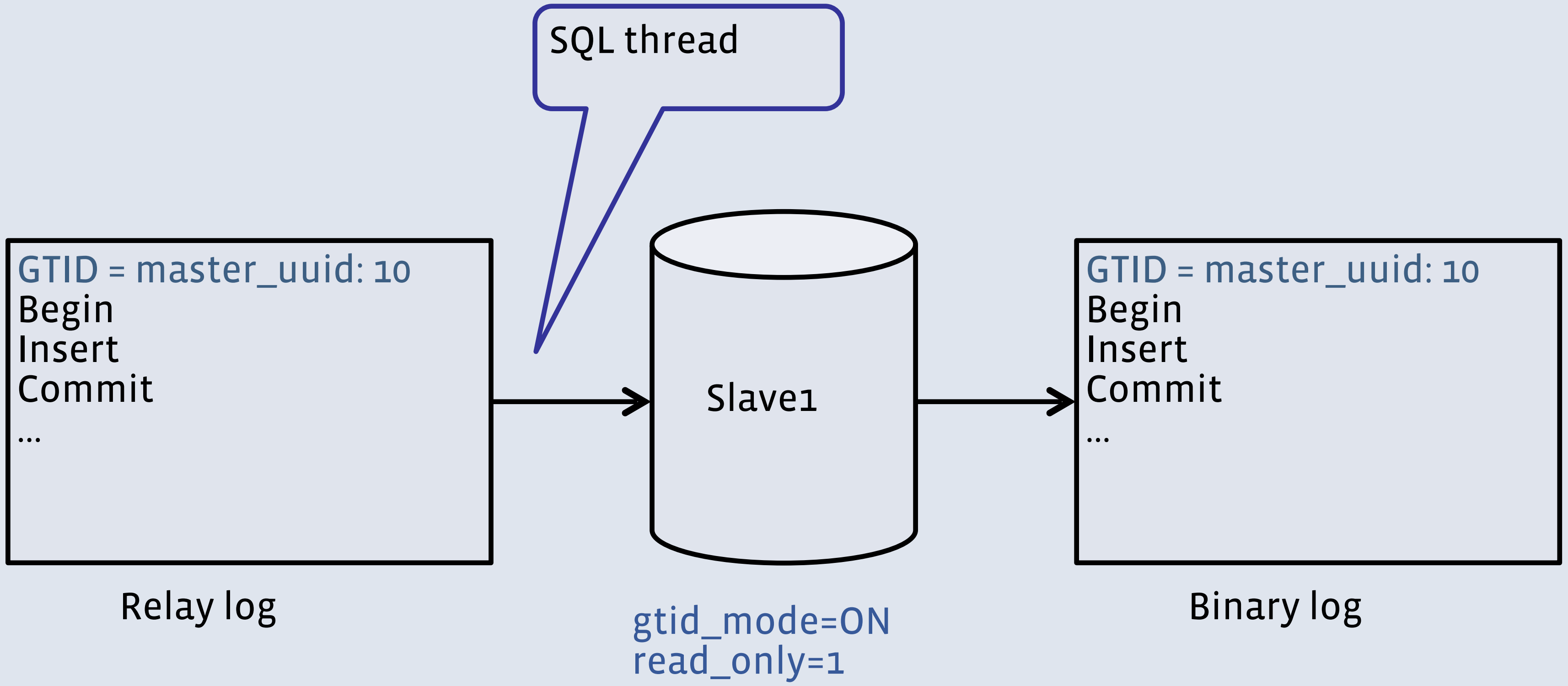
- Functionality to block generation of new GTIDs on slaves!
- Make sql_thread never generate GTIDs, but log GTIDs from master
- Use read_only=1 setting on slaves to block GTID generation
- Allow slave with –read_only=1 to do replication from a master with –gtid_mode=OFF



Simplify GTID deployment

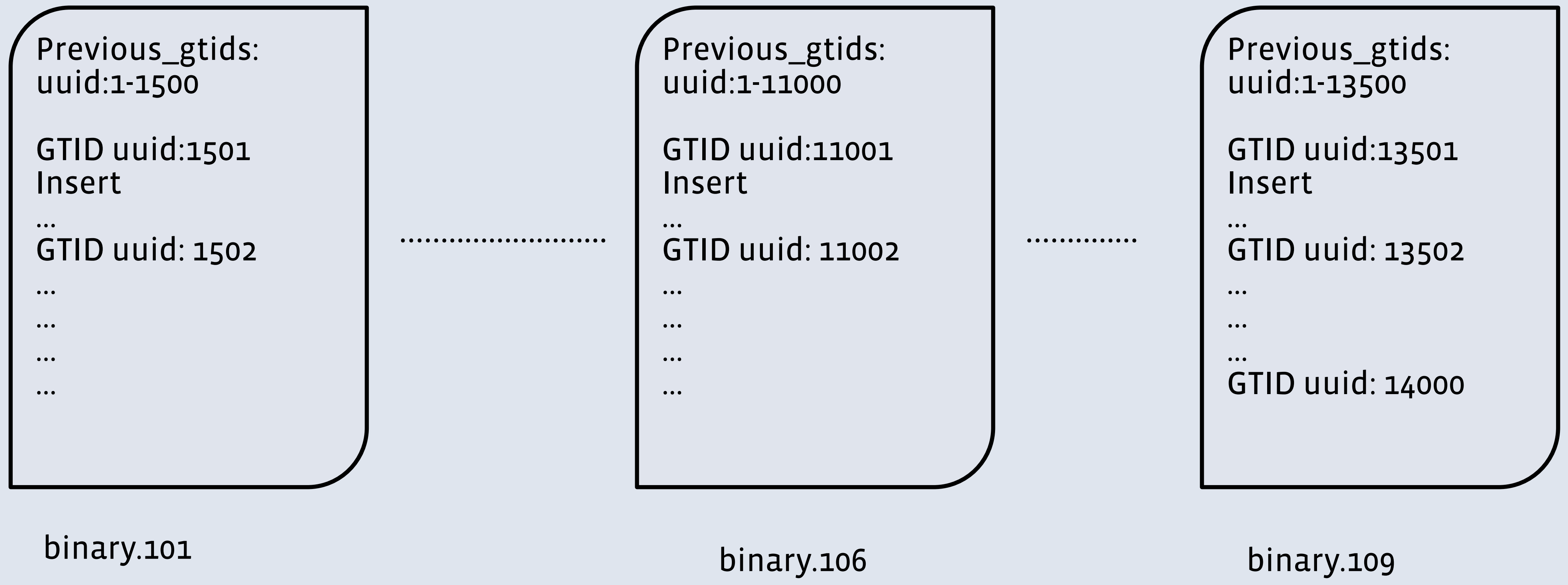
How?

- SQL_thread logs GTIDs received from master



Slave connect performance

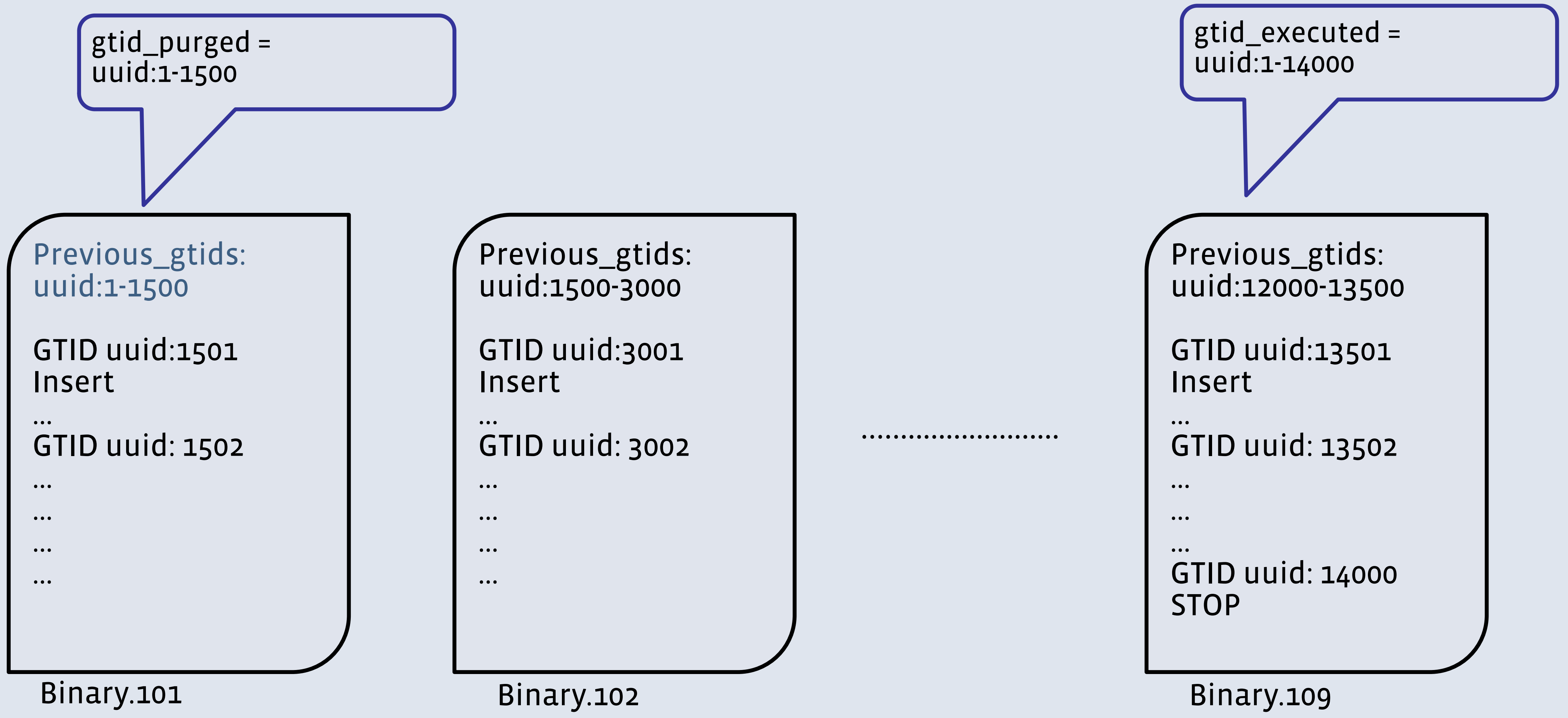
- In MySQL 5.6.10, dump thread scans all binary logs when slave connects with auto positioning
- After MySQL 5.6.10, dump thread opens binary logs in reverse order when slave connects with auto positioning. It opens logs from binary.109 to binary.106 if slave's gtid_executed='uuid:1-11000'



Binary logs on a master server

Restart performance

- Mysqld opens binary logs in forward order on restart or crash recovery to initiate gtid_set gtid_purged and in reverse order to initiate gtid_executed by using Previous_gtid_log_events which are logged in the beginning of every binary log
- Opens all binlogs if GTIDs are not enabled, or GTIDs are enabled recently



Change binary log index file format

- Keep the Previous_gtid_log_events in memory and log them in the binary log index file along with the file name
- Use the previous_gtid_log_events in memory to figure out the physical location when slave uses auto positioning
- On server restart, the Previous_gtid_log_events in the binary log index file are used to initiate gtid_sets
- Slave uses BINLOG_DUMP protocol instead of BINLOG_DUMP_GTID upon IO_thread reconnect!

```
Binlog.100 (size of previous_gtid_log_event in binary format)
uuid:1-1500
binlog.00100
uuid:1500-3000
......
Binlog.106
uuid:1-11000
......
......
Binlog.109
uuid:12000-13500
```

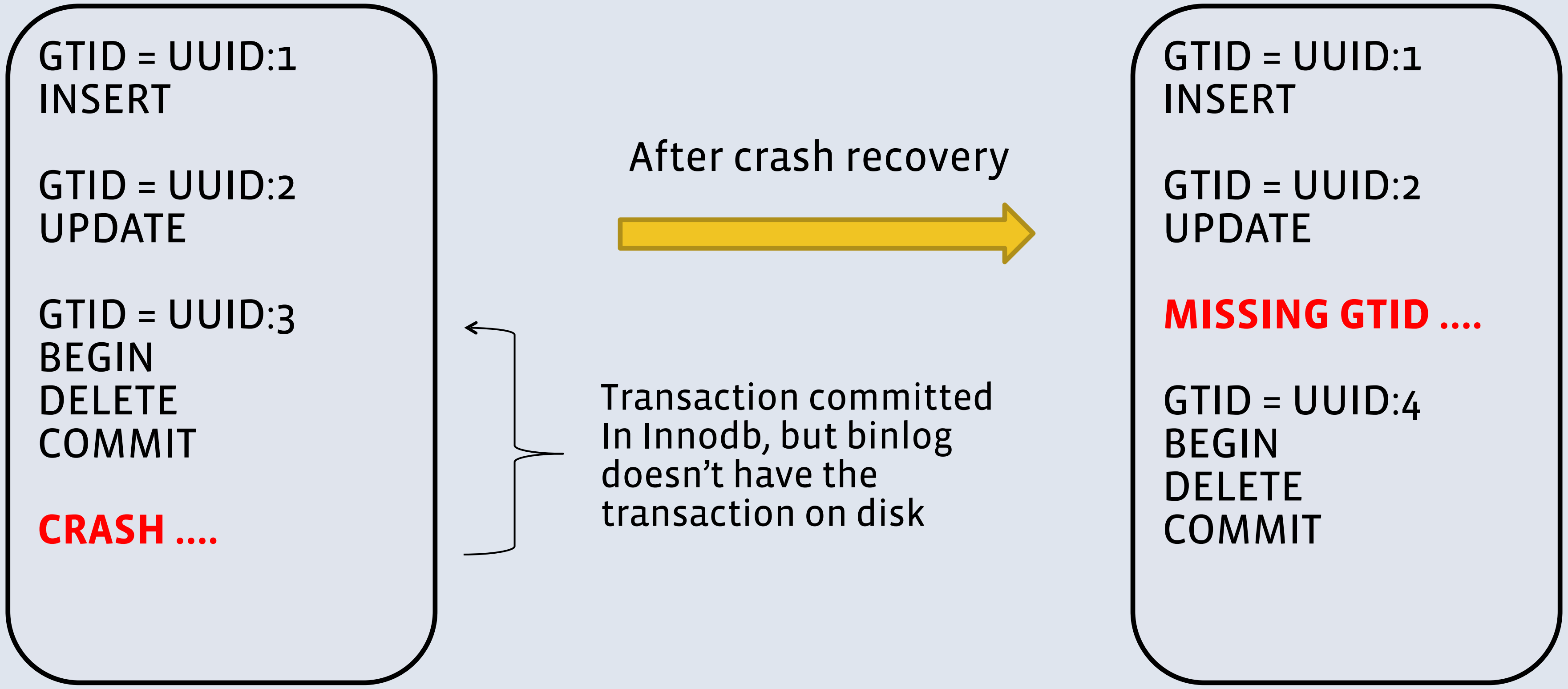
File format of binlog.index

Crash-safe slave

- Running with full durability (sync_binlog=1 and innodb_flush_log_at_trx_commit =1) negatively affects performance
- We want slaves running with reduced durability (sync_binlog != 1 and innodb_flush_log_at_trx_commit != 1) to be consistent even after a soft crash or a hard crash
- MySQL 5.6 has crash safe functionality with the use of relay_log_info_repository=TABLE. But it is not crash safe when GTIDs are enabled. Slave_relay_log_info table only stores master binary log file name and position but doesn't have any GTID information

Crash-safe slave: scenario 1

- If slave's binlog is behind Innodb transaction log during the crash, slave misses some GTIDs in binary logs that are committed in Innodb after crash recovery
- Slaves recovers if auto-positioning is OFF during the crash but will have missing GTIDs which will cause problems when auto-positioning is enabled later. But slave hits duplicate key error when auto-positioning is ON during the crash



slaves binary log is behind Innodb transaction log

Crash-safe slave: scenario 1 fix

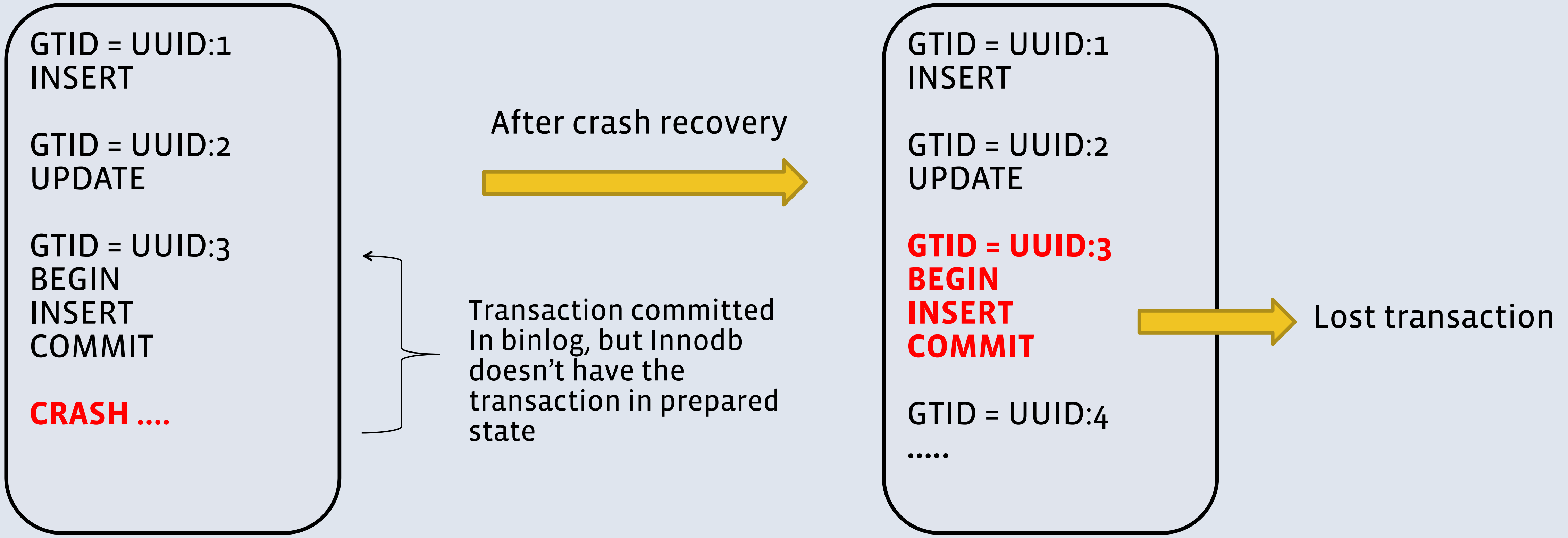
- Store the last committed GTID in a table slave_gtid_info. The transactional table is updated inside the SQL_thread transaction or a slave worker in case of multi threaded slave
- Missing GTIDs in the binlog are identified using the slave_gtid_info table by checking the condition if current GTID of the SQL_thread transaction is less than last committed GTID in the table. SQL_thread logs those transactions directly into the binary log instead of executing them

```
SET @@GTID_NEXT = 'UUID:1332492';
START TRANSACTION;
INSERT INTO db1.tbl(col1, col2, blob)
UPDATE mysql.slave_gtid_info
  SET Last_gtid = 'UUID:1332492' where Database_name = 'db1'; // Added by SQL thread
COMMIT;
```

SQL_thread executing a transaction

Crash-safe slave: scenario 2

- If slaves binary log is ahead of Innodb transaction log during the crash, slave loses transactions
- Since slaves gtid_executed contains UUID:3 after crash recovery, master skips sending the transaction corresponding to UUID:3



slaves binary log is ahead of Innodb transaction log

Crash-safe slave: scenario 2 fix

- After crash recovery slave uses these physical binary log coordinates from Innodb log header to trim gtid_executed before starting replication from master

GTID bug reports

- http://bugs.mysql.com/bug.php?id=69059 GTIDs lack a reasonable deployment strategy
- http://bugs.mysql.com/bug.php?id=69097 MySQL scans all binary logs on crash recovery
- http://bugs.mysql.com/bug.php?id=68386 Master scans all binary logs when slave use auto-position
- http://bugs.mysql.com/bug.php?id=70659 Make GTID crash safe
- http://bugs.mysql.com/bug.php?id=71575 Master logs consecutive Gtid events
- http://bugs.mysql.com/bug.php?id=71695 Concatenation of mysqlbinlog output does not work with GTID

Automation and Monitoring

Potential integration points

- Promotion logic
- Monitoring
- Config file generation
- Instance copying
- Backups (and restores!)
- Helper scripts for DBAs

Promotion logic

- Handle 3 cases of gtid_mode:
  - Traditional: gtid_mode=OFF on both old and new masters
  - GTID failover: gtid_mode=ON on both old and new masters
  - Rollout: gtid_mode=OFF on old master but ON for new master
- Many additional permutations to test
  - Which replica / binlog tailer is furthest ahead
  - Planned promotion (old master alive) vs failover (old master dead)

Monitoring

- Broken replication from mismatched gtid_mode (slave OFF, master ON)
- Replica sets using GTID but without auto-positioning — not crash-safe
- Gaps in gtid_executed
  - Should only occur if mysql server bug
  - Breaks auto-positioning
- Use of GTID-incompatible statements
  - Anything mixing DDL/nontransactional and transactional updates
  - FB-mysql adds user stat counters for these
- At Facebook, we use on-host agent plus external monitoring, and we also have external auto-remediation for solving common problems

Config file generation

- gtid_mode not dynamic in 5.6, so my.cnf is responsible for enabling
- Must support replica-set-at-a-time gtid_mode rollout
- Programmatically set gtid_mode and enforce_gtid_consistency
  - Enable if a flag has been set in asset tracker
  - Enable if gtid rollout complete for this entire logical data set
  - Enable if master already has gtid_mode=ON
  - Disable if version < 5.6

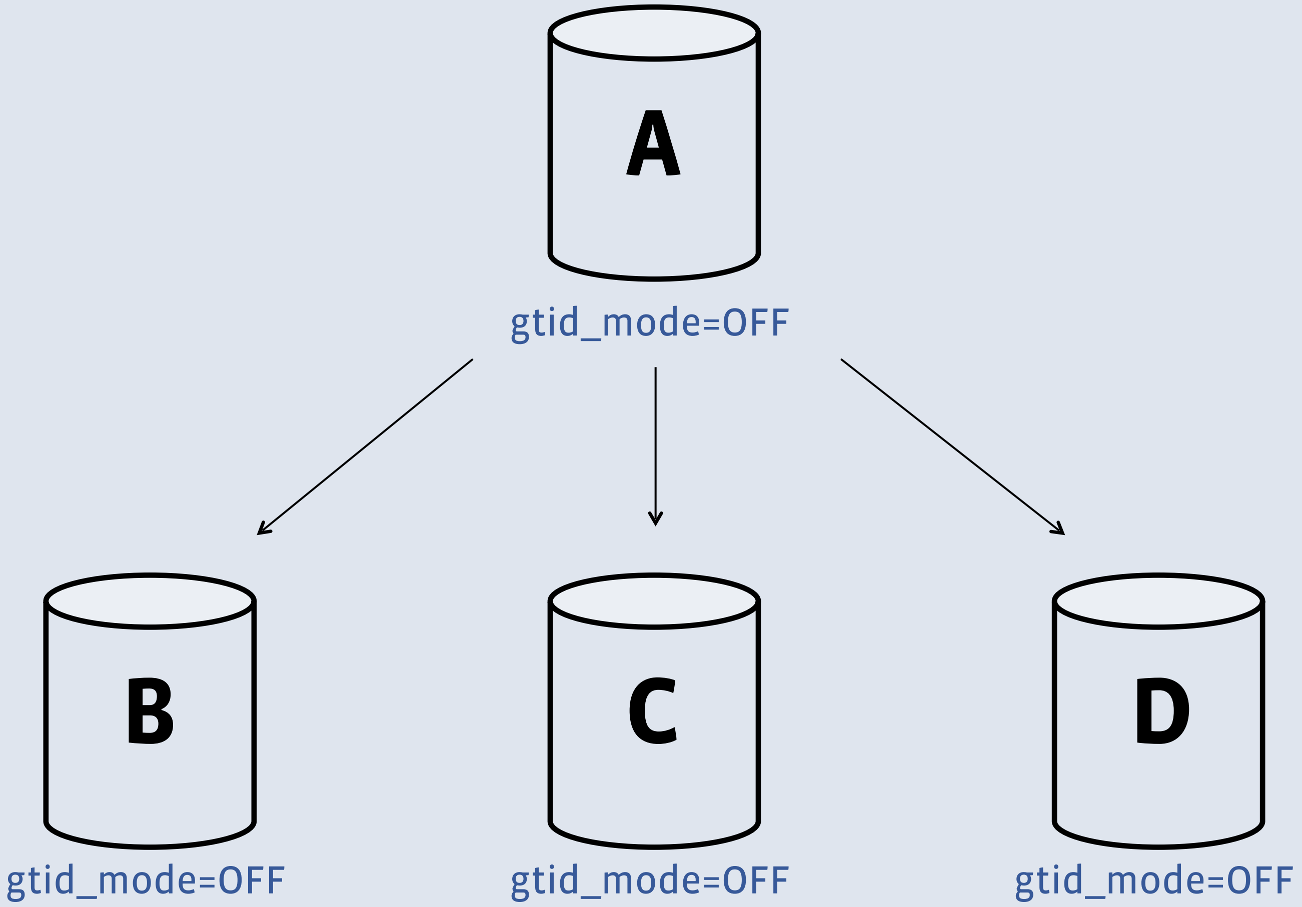
Helper scripts for DBAs

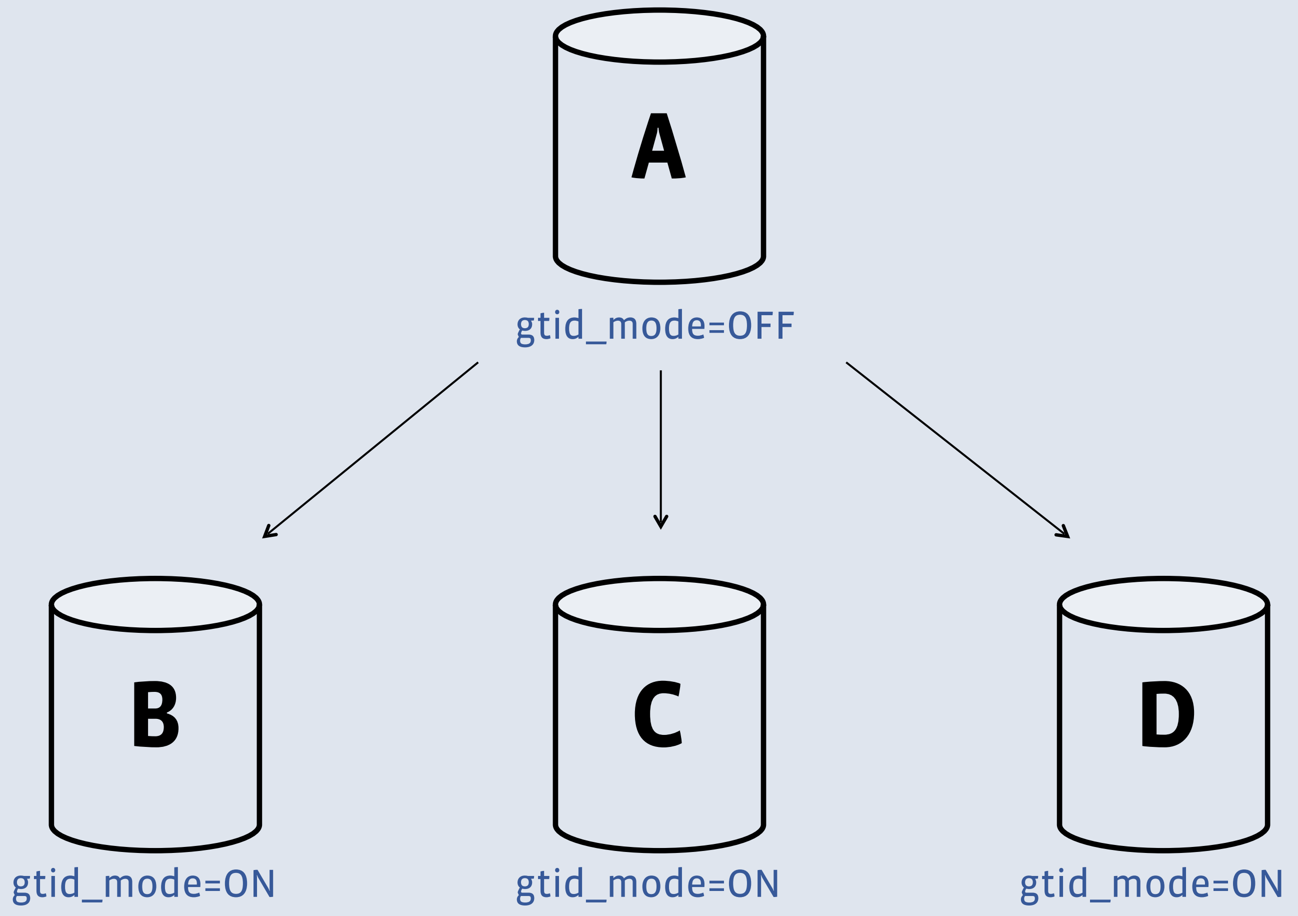
- Rollout script
- Rollback script? Rather painful
- Statement skipper
  - If gtid_mode=OFF, use sql_slave_skip_counter
  - If gtid_mode=ON, need to insert empty transaction instead

Deployment Strategy

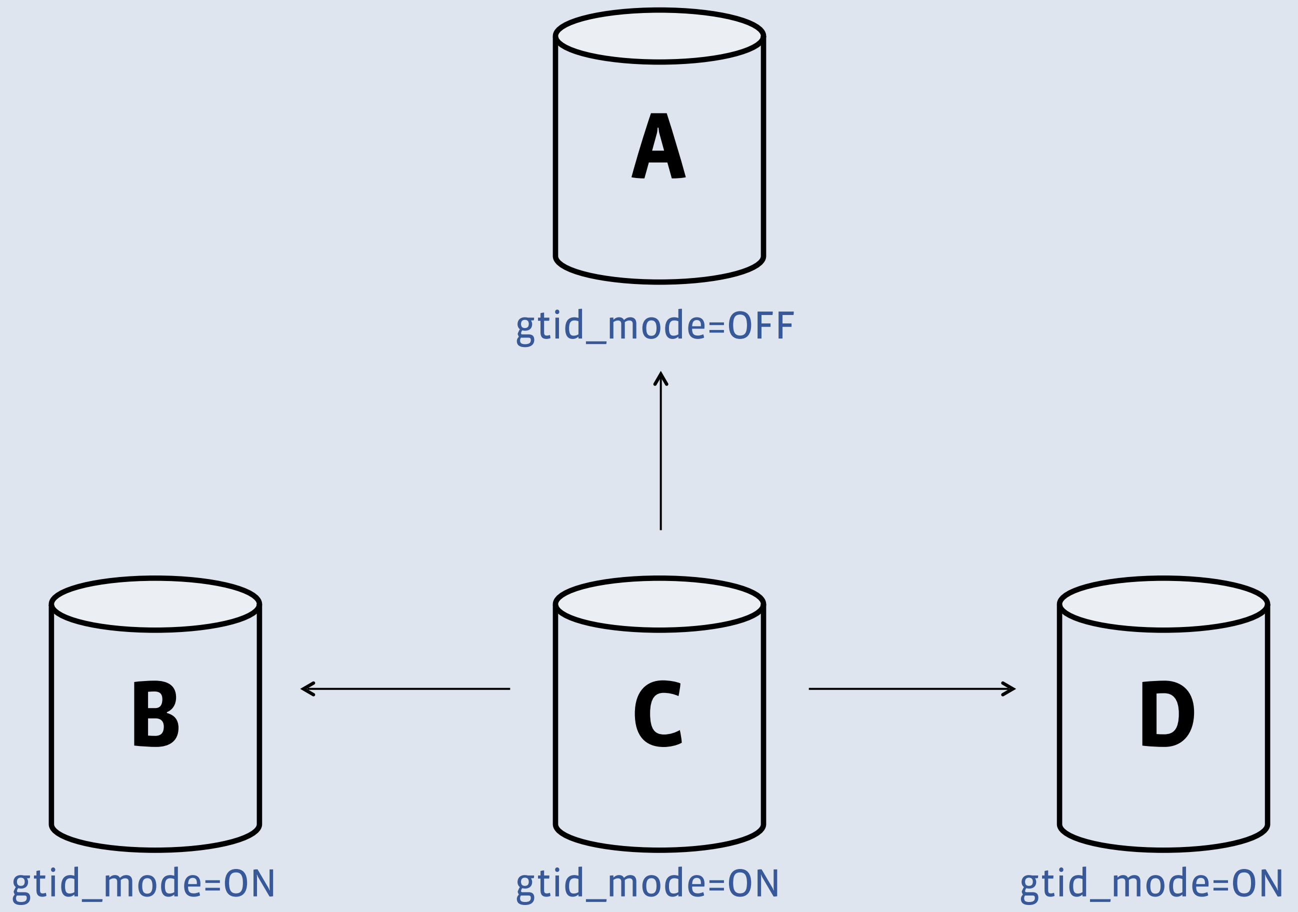
Our rollout process

1. Set flag on replica set to enable gtid_mode in my.cnf generator
2. Restart replicas one-at-a-time to enable gtid_mode
   - Stop client conns and suppress monitoring/automation first
   - Prep fast shutdown by setting innodb_max_dirty_pages_pct=0
3. Perform a promotion
   - Brief splay time, to avoid too many concurrent promotions
   - New master has gtid_mode=ON, so afterwards GTID fully in use
   - Still repoint old master, even though i/o thread will error upon start
4. Restart old master to enable gtid_mode, replication will work now

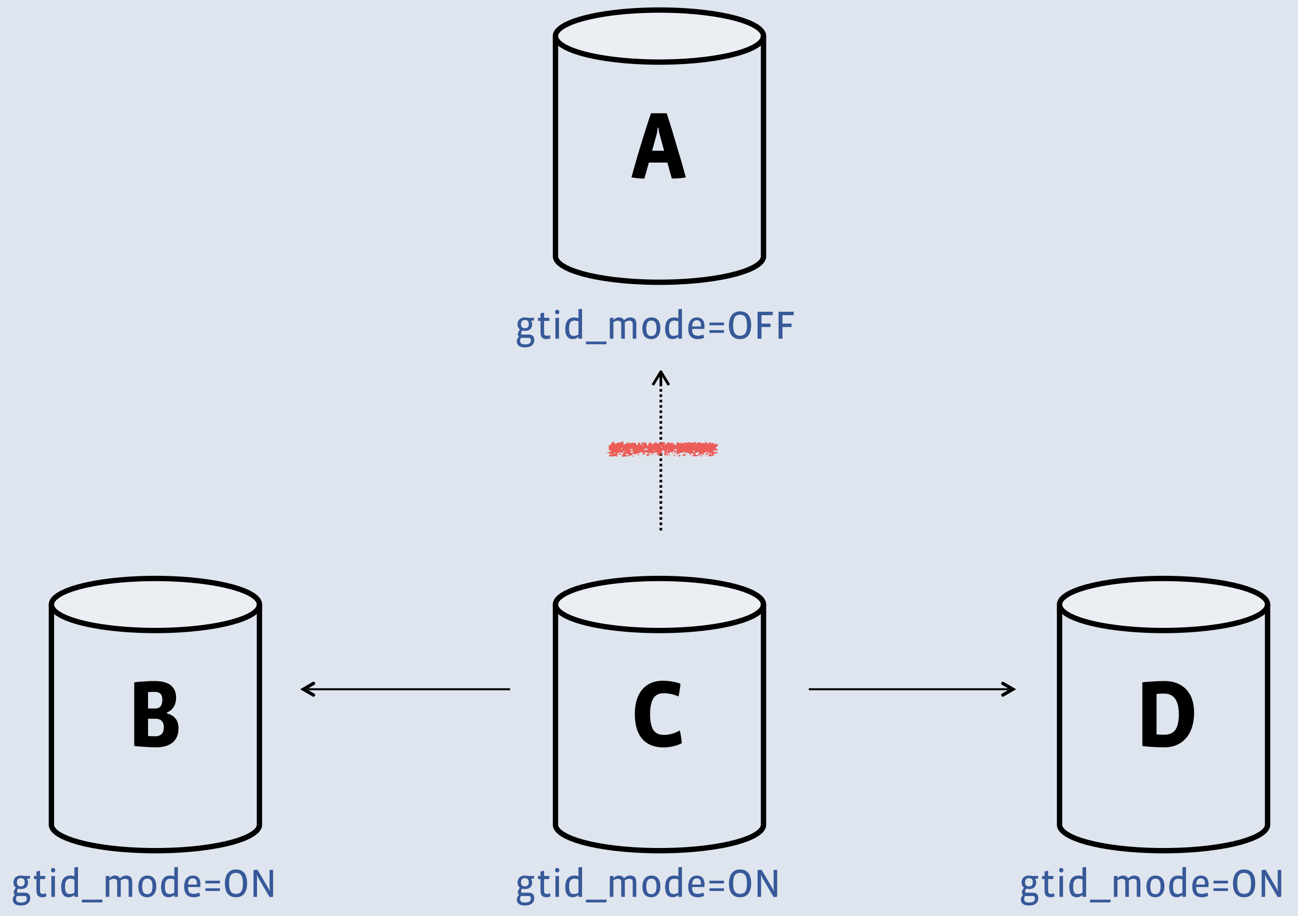

A
gtid_mode=OFF
B
gtid_mode=OFF
C
gtid_mode=OFF
D
gtid_mode=OFF



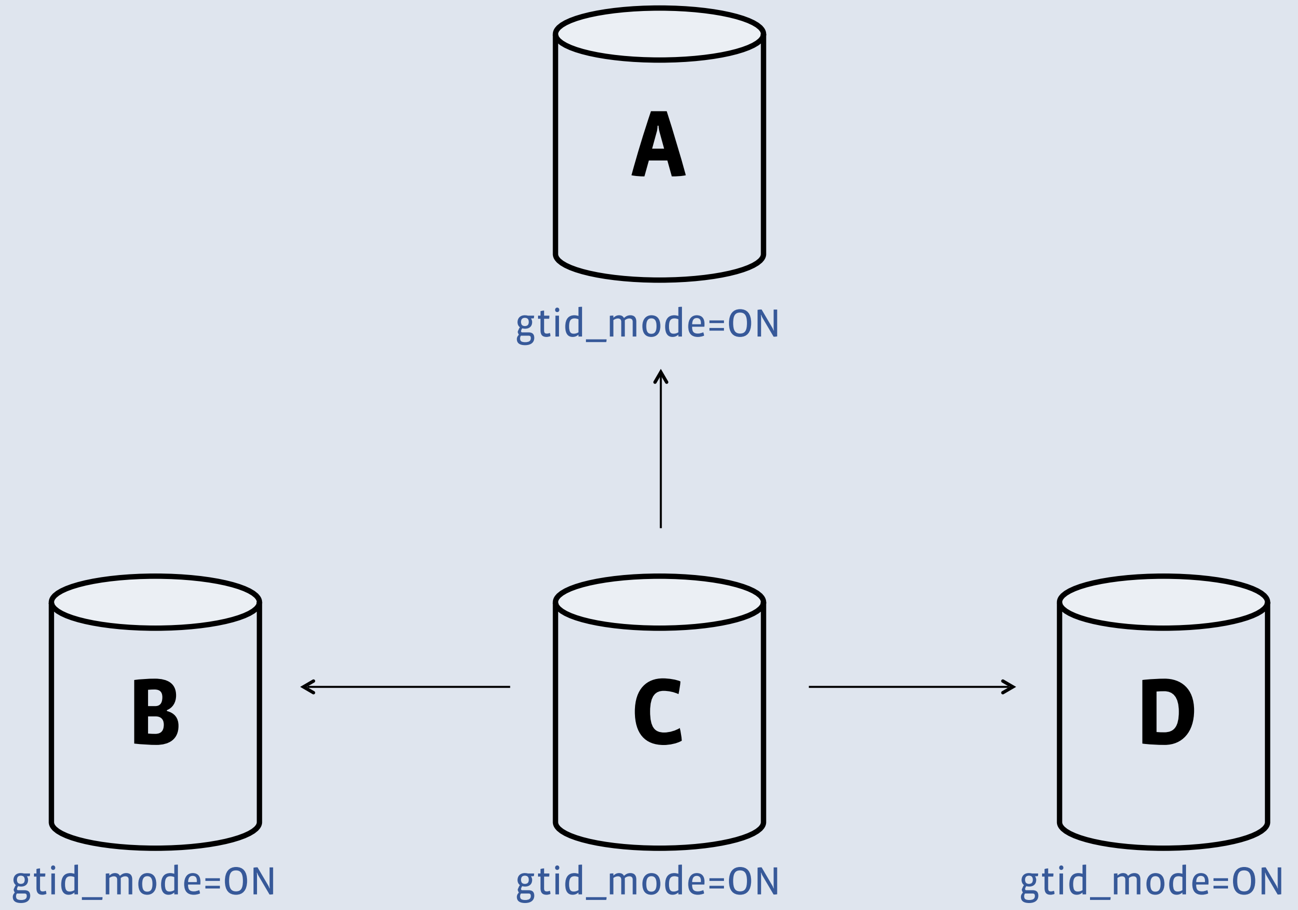
Enable gtid_mode on replicas, one at a time



Promote C to be the new master



I/O thread on A breaks immediately due to gtid_mode mismatch



Restart A to enable gtid_mode and fix replication

Rollout validations

- Locking to prevent other automation from touching replica set. This includes copies, promotions, shard migrations, upgrades
- Confirm valid failover replica that isn’t lagging
- User stat counters show no use of GTID-incompatible statements
- Replica set is reasonably healthy, no instances offline or broken
- All steps must succeed — confirm clean shutdown and startup without hitting timeouts
- Validate gtid_mode actually enabled after restart
- Confirm 5.6.x; if older 5.6.x, can upgrade instead of simple restart

Questions?

facebook