PERCONA

New Replication Features In MySQL 5.6

Stéphane Combaudon
October 23rd, 2013

Agenda

- Introduction
- Binlog checksums
- Crash safe replication
- Multi threaded slaves
- Global Transactions IDs

Introduction

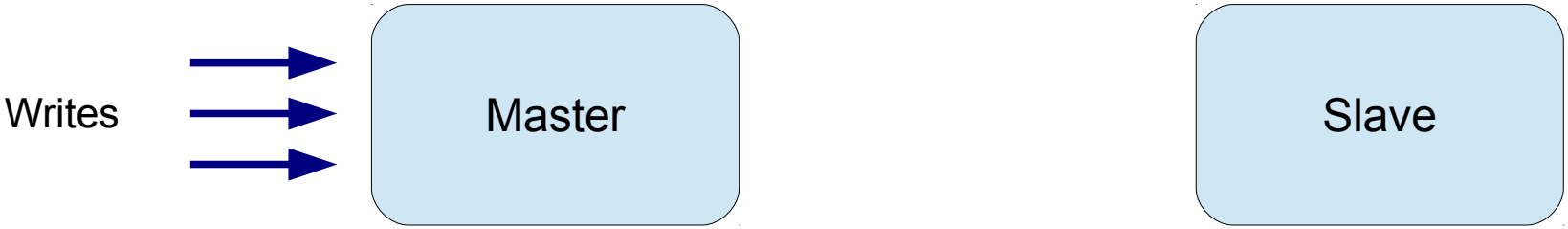
Why using replication?

- Scaling out
  - Offload some of the reads to slaves
- High availability
  - If master fails, you can promote a slave
- Backups
  - Impact of backups is less problematic on slaves
- Tests & Upgrades
  - Run the application load on a slave first

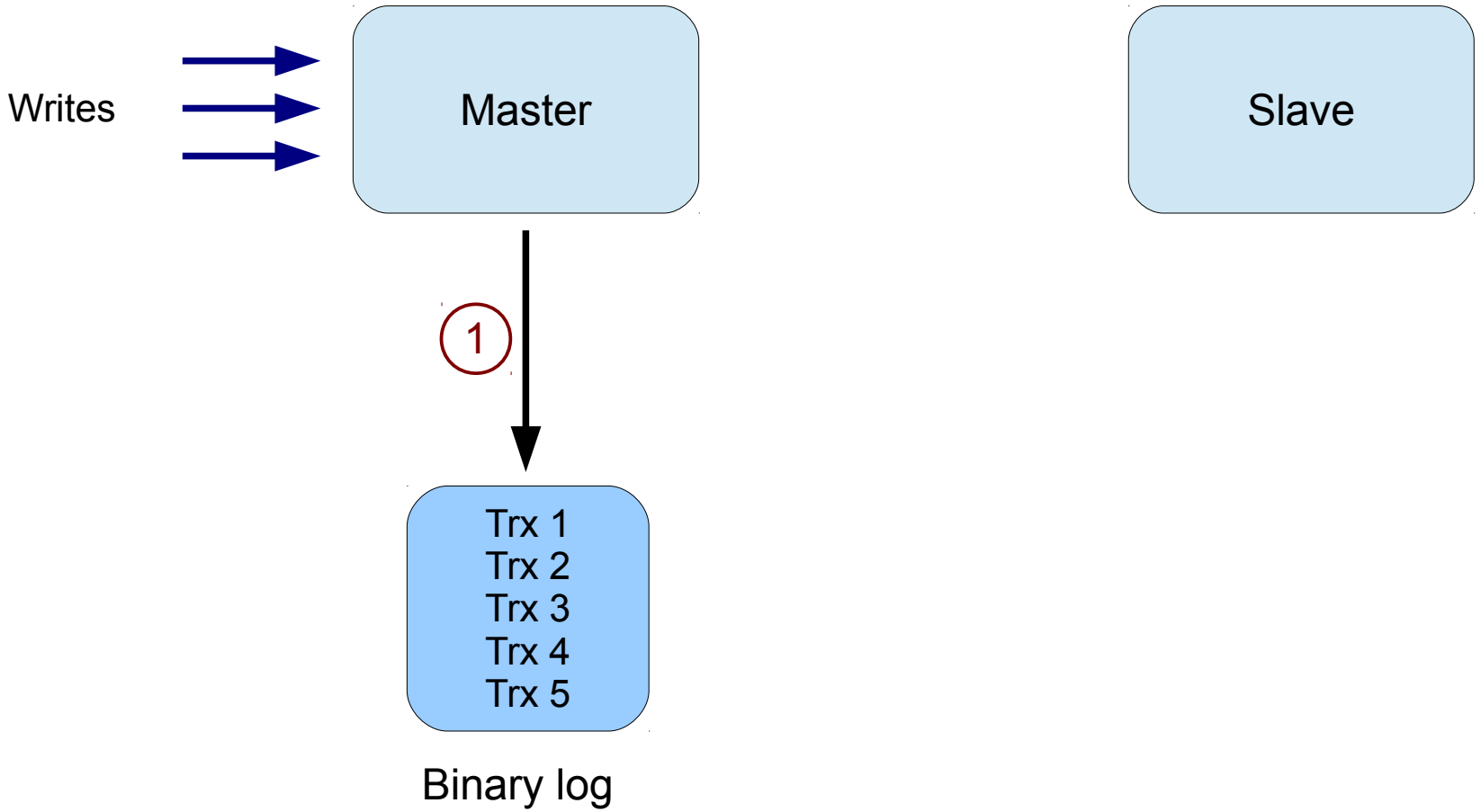
High-level description

High-level description

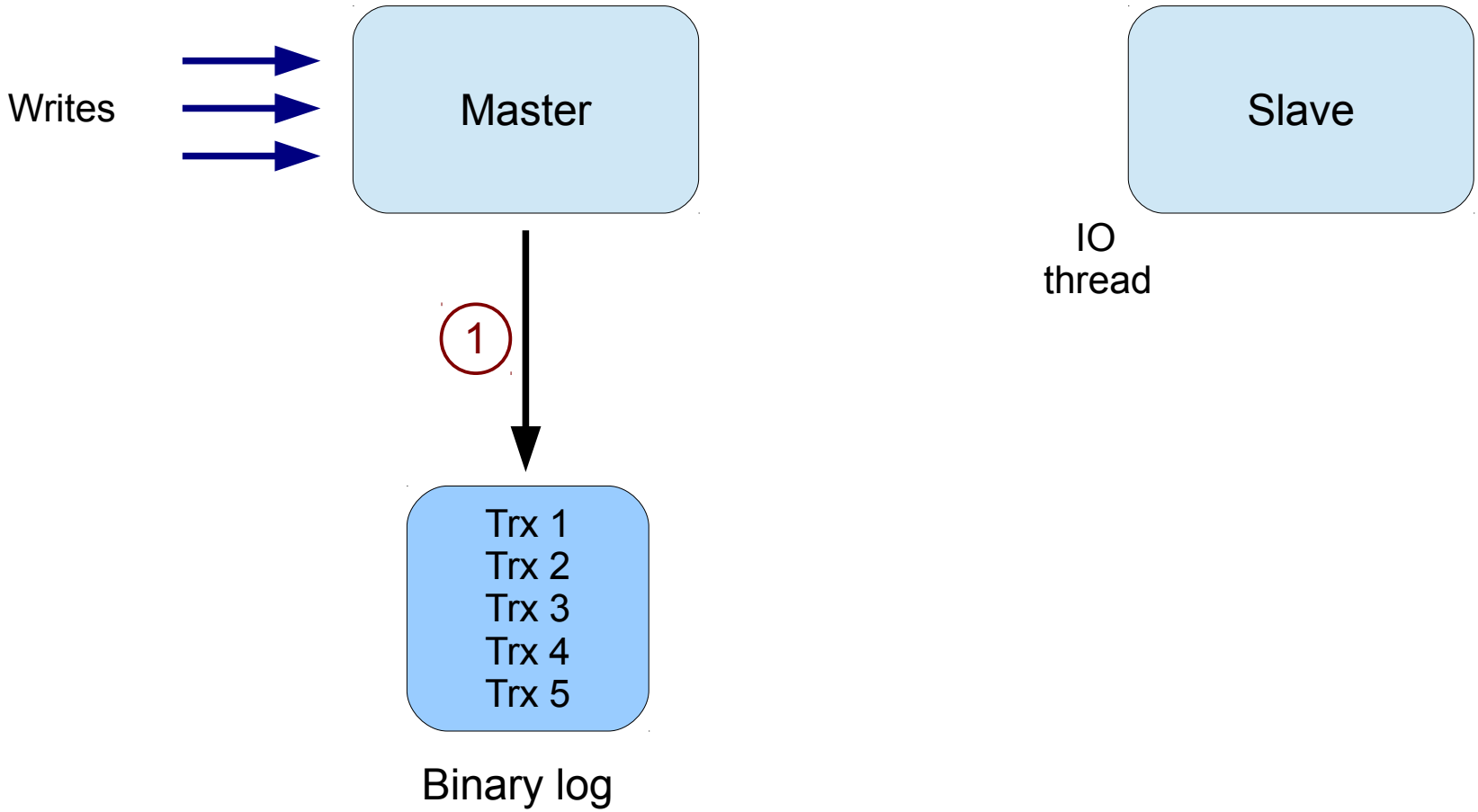
High-level description



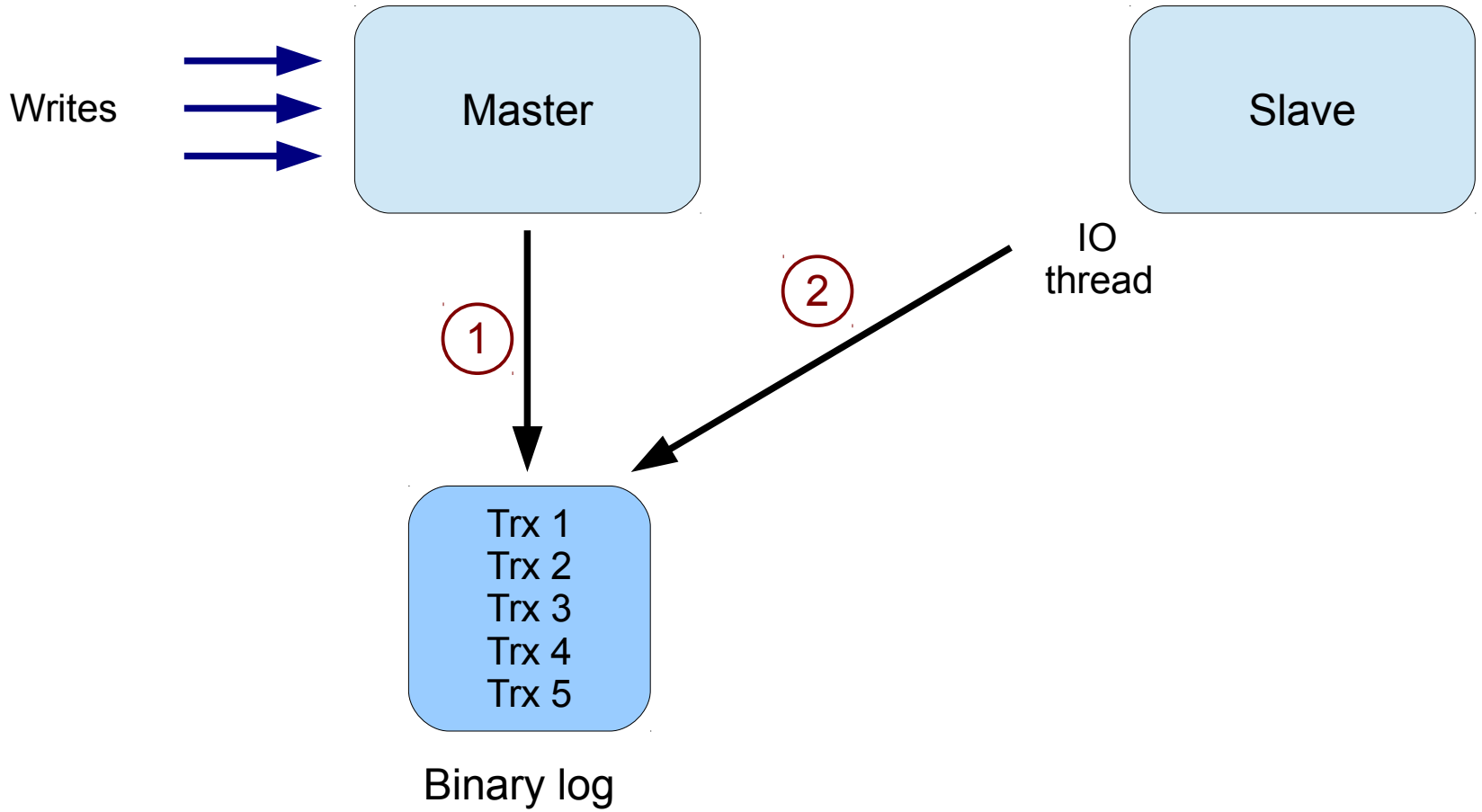
High-level description



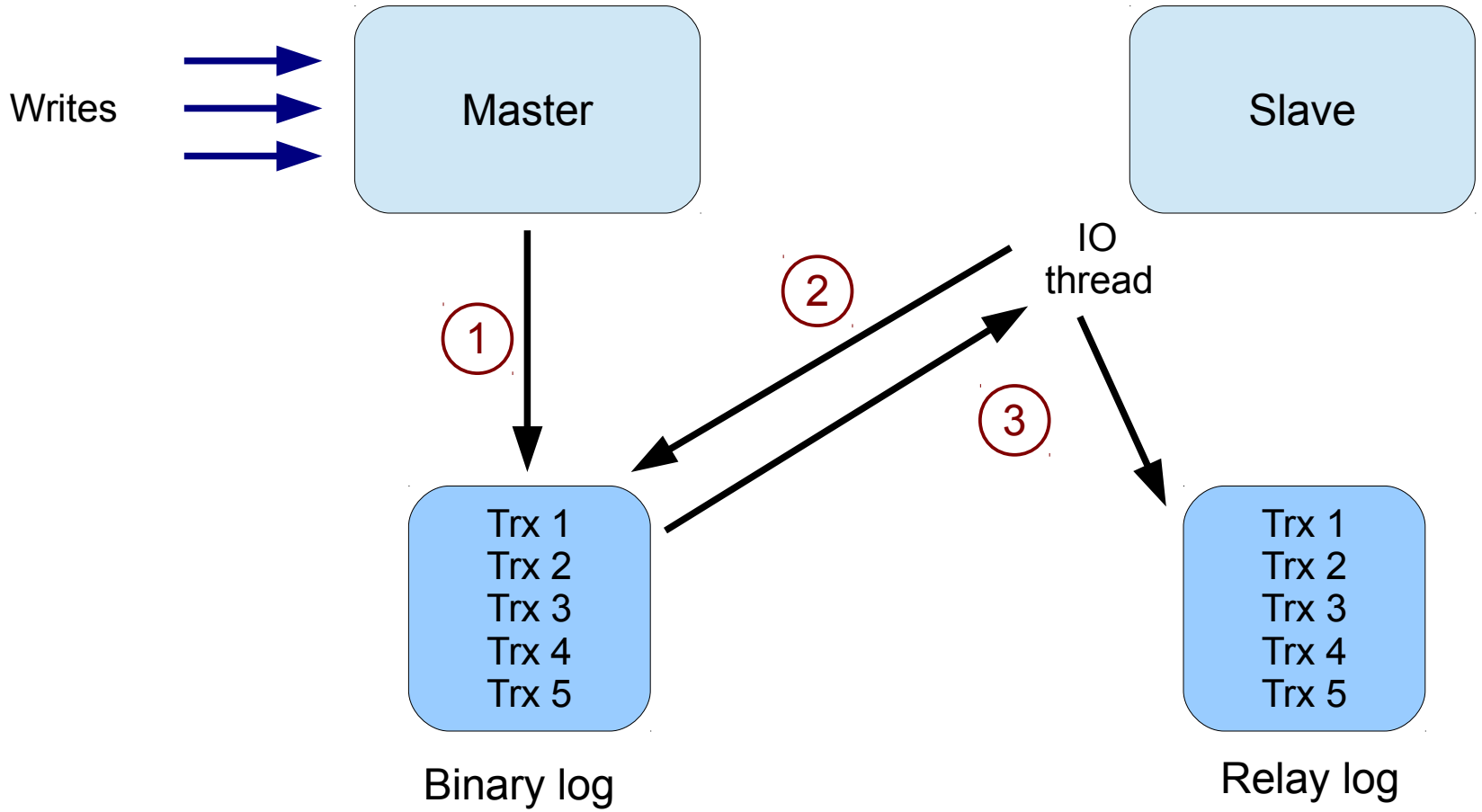
High-level description



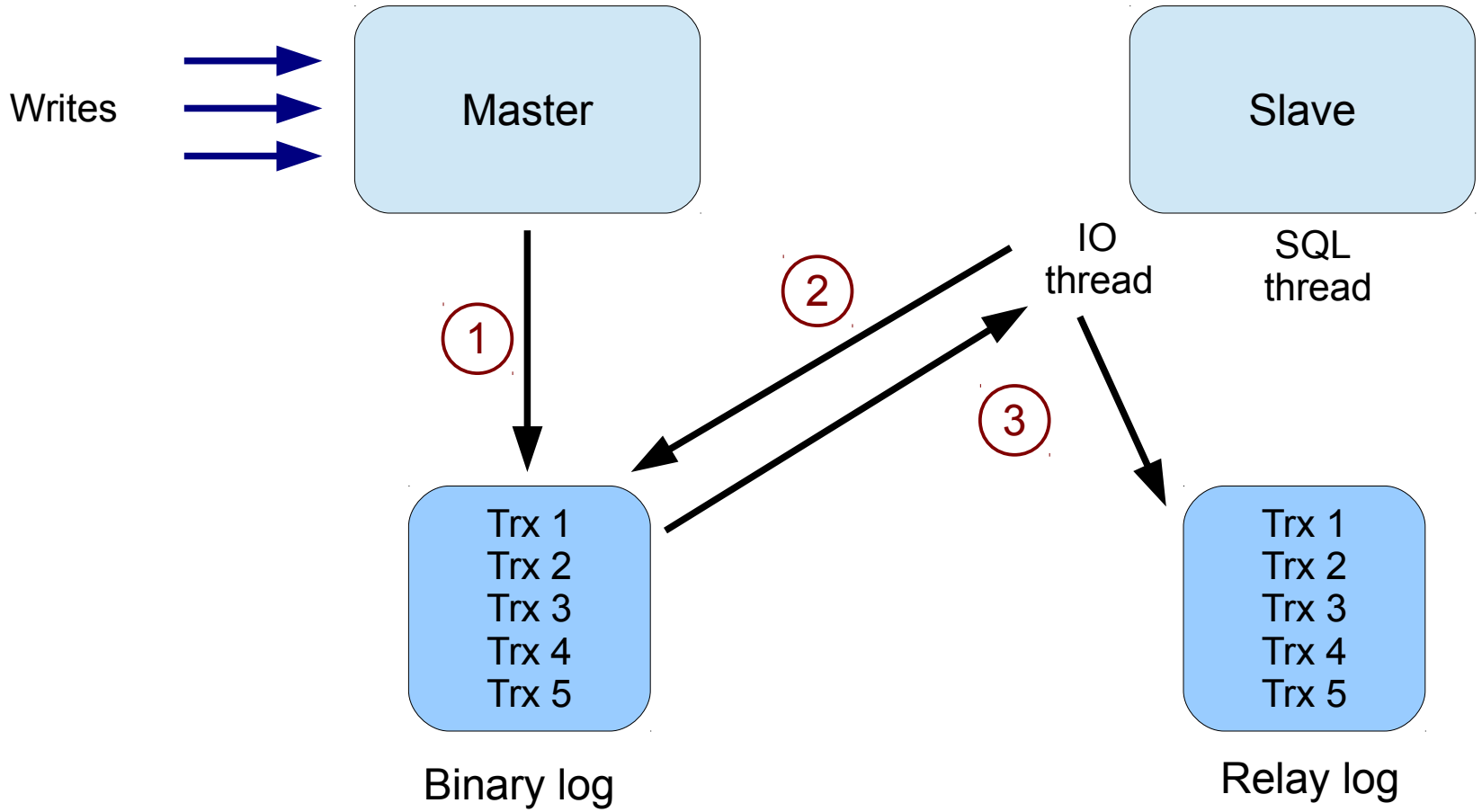
High-level description



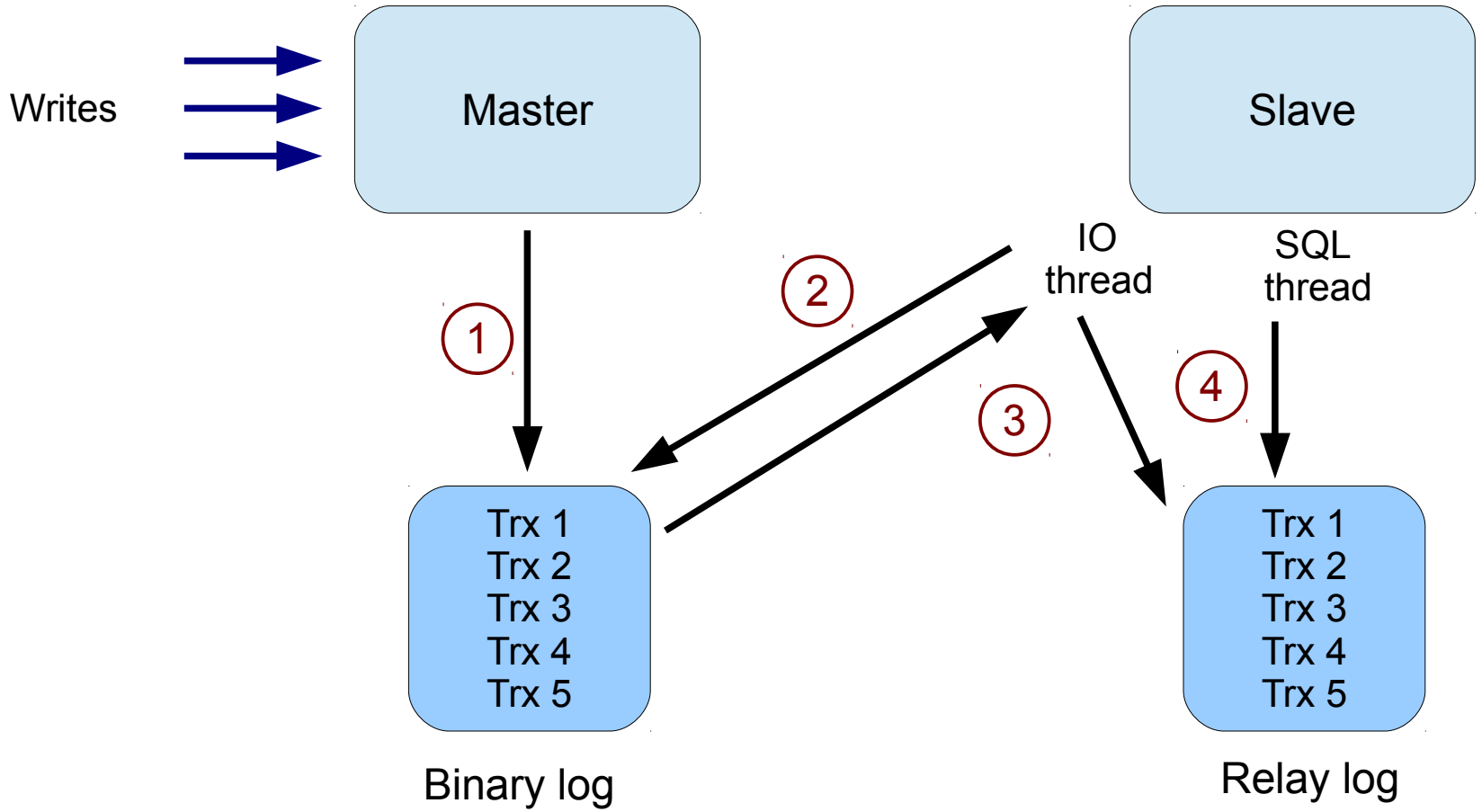
High-level description



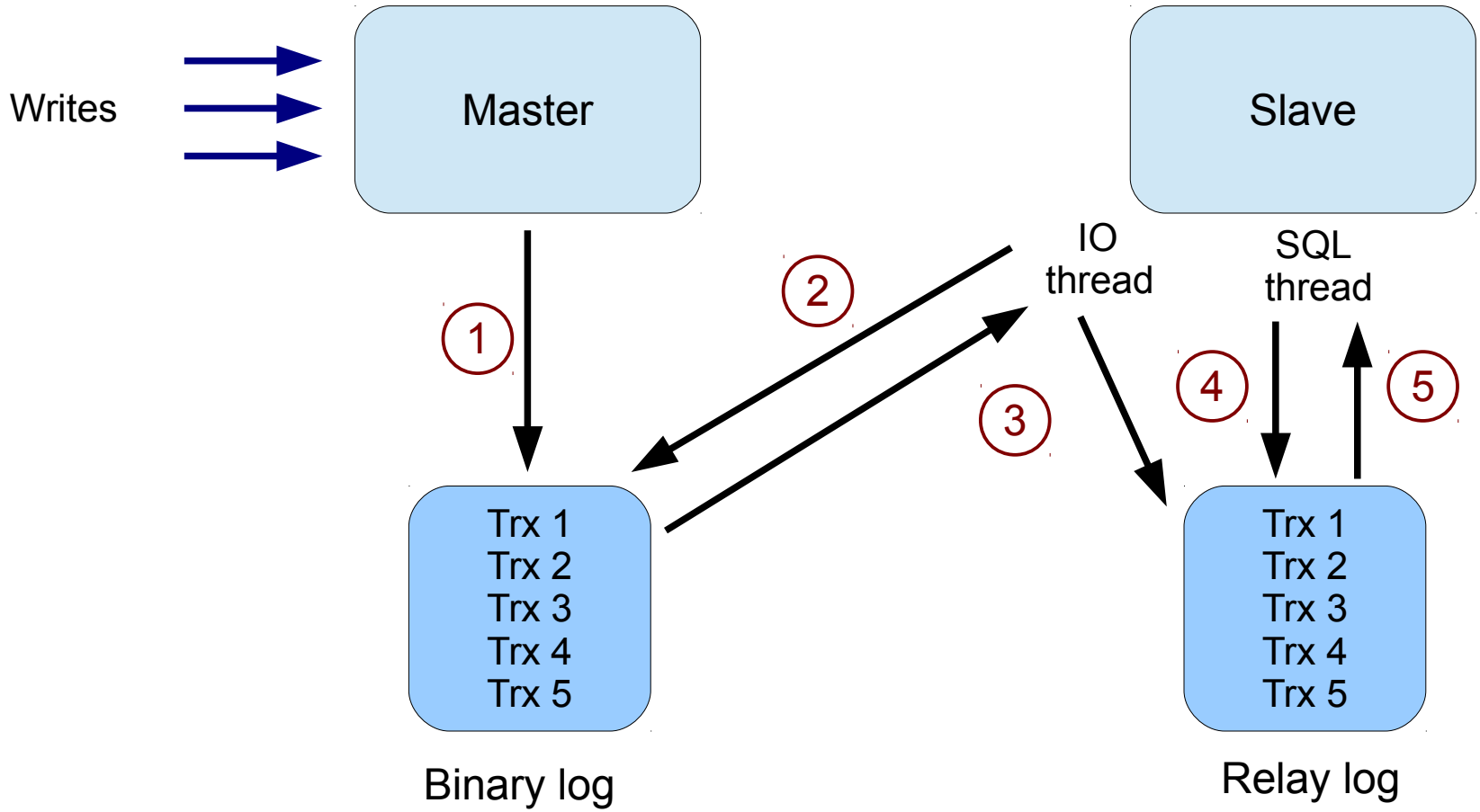
High-level description



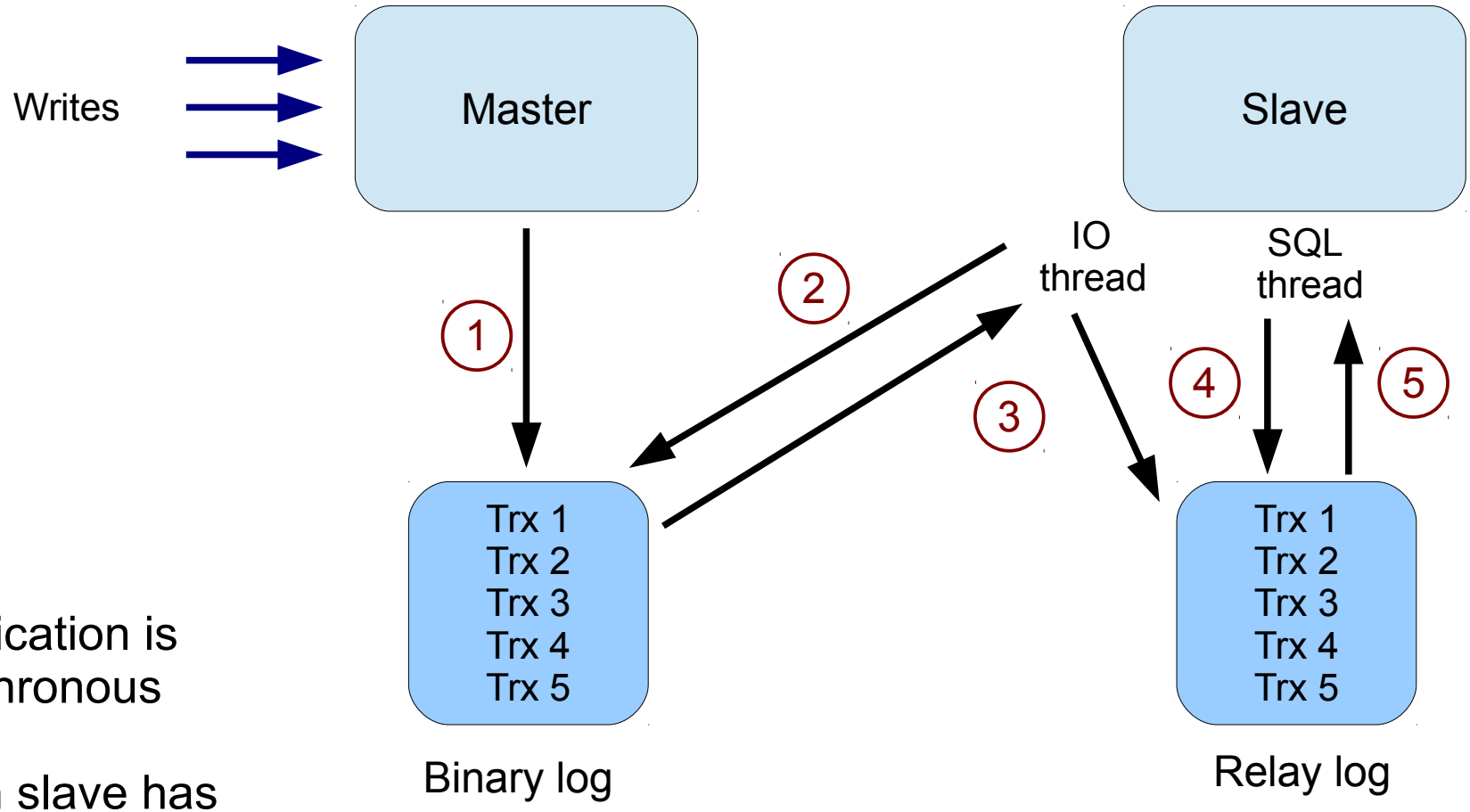
High-level description



High-level description

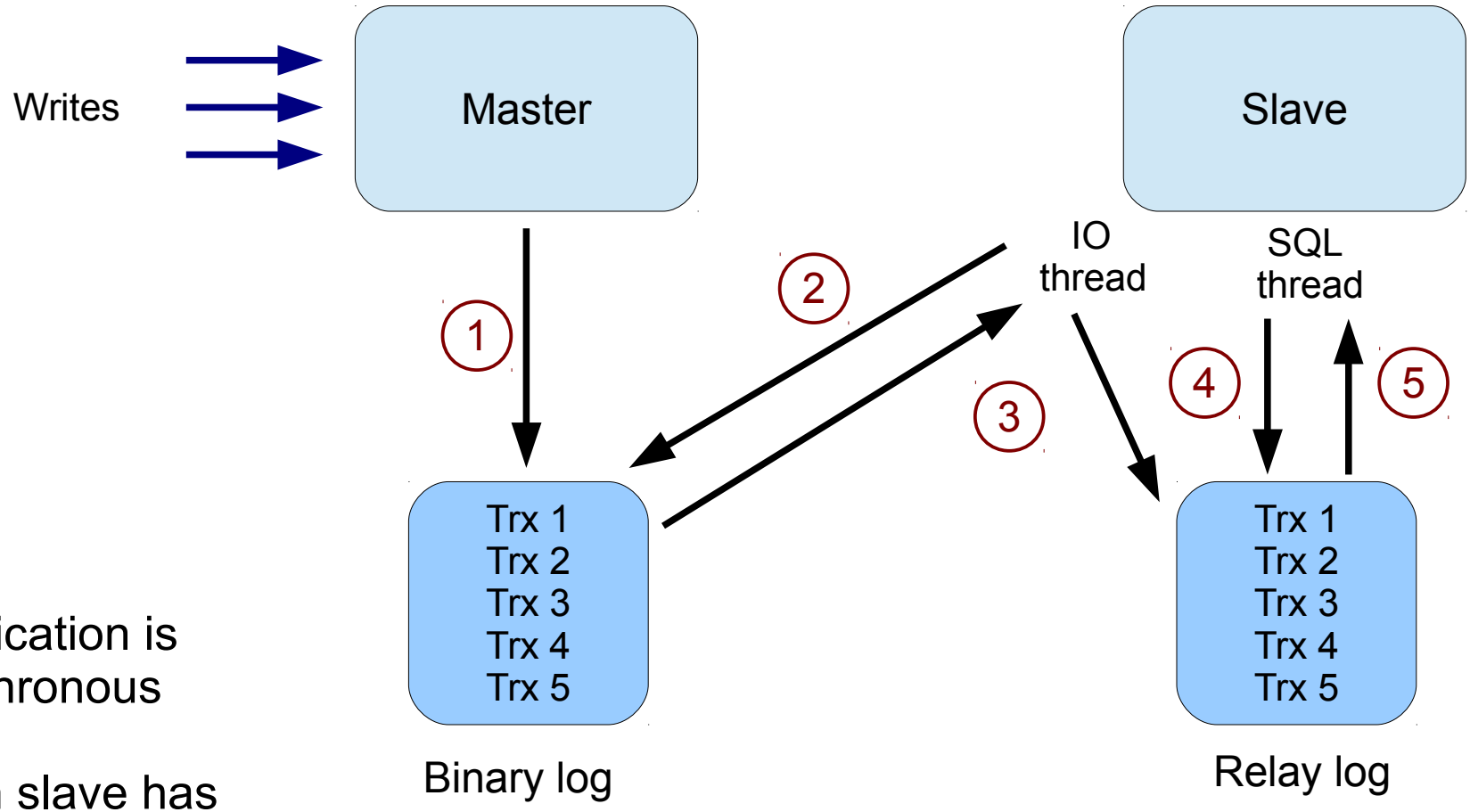


High-level description



- Replication is asynchronous
- Each slave has its own relay log

High-level description



- Replication is asynchronous
- Each slave has its own relay log

Binlog checksums

What is the problem?

- Master and slaves can have different data
- Even when replication runs without error. Why?
  - Non deterministic queries
  - Writes executed on slaves outside of replication
  - Wrong usage of replication filters
  - Using a non transactional storage engine
  - Binlog or relay log corruption
  - ...

How to check?

- Before 5.6, nothing was provided by the server
- Your best bet was to use pt-table-checksum
  - http://www.percona.com/doc/percona-toolkit/2.2/pt-table-checksum.html

```
[stephane@centos2 ~]$ pt-table-checksum --empty-replicate-table -d sakila h=127.0.0.1,P=22585
            TS ERRORS  DIFFS     ROWS  CHUNKS SKIPPED    TIME TABLE
04-03T15:39:37      0      0      200       1       0   0.281 sakila.actor
04-03T15:39:38      0      0      603       1       0   0.299 sakila.address
04-03T15:39:38      0      0       16       1       0   0.047 sakila.category
04-03T15:39:38      0      0      600       1       0   0.049 sakila.city
04-03T15:39:38      0      0      109       1       0   0.049 sakila.country
04-03T15:39:38      0      0      599       1       0   0.302 sakila.customer
04-03T15:39:38      0      0     1000       1       0   0.292 sakila.film
04-03T15:39:39      0      0     5462       1       0   0.307 sakila.film_actor
04-03T15:39:39      0      0     1000       1       0   0.300 sakila.film_category
04-03T15:39:39      0      0     1000       1       0   0.048 sakila.film_text
04-03T15:39:39      0      0     4581       1       0   0.301 sakila.inventory
04-03T15:39:39      0      0        6       1       0   0.083 sakila.language
04-03T15:39:40      0      1    16049       1       0   0.381 sakila.payment
04-03T15:39:40      0      0    16044       1       0   0.355 sakila.rental
04-03T15:39:40      0      0        2       1       0   0.034 sakila.staff
04-03T15:39:40      0      0        2       1       0   0.056 sakila.store
```

How to check?

- Before 5.6, nothing was provided by the server
- Your best bet was to use pt-table-checksum
  - http://www.percona.com/doc/percona-toolkit/2.2/pt-table-checksum.html

```
[stephane@centos2 ~]$ pt-table-checksum --empty-replicate-table -d sakila h=127.0.0.1,P=22585
            TS ERRORS  DIFFS     ROWS  CHUNKS SKIPPED    TIME TABLE
04-03T15:39:37      0      0      200       1       0   0.281 sakila.actor
04-03T15:39:38      0      0      603       1       0   0.299 sakila.address
04-03T15:39:38      0      0       16       1       0   0.047 sakila.category
04-03T15:39:38      0      0      600       1       0   0.049 sakila.city
04-03T15:39:38      0      0      109       1       0   0.049 sakila.country
04-03T15:39:38      0      0      599       1       0   0.302 sakila.customer
04-03T15:39:38      0      0     1000       1       0   0.292 sakila.film
04-03T15:39:39      0      0     5462       1       0   0.307 sakila.film_actor
04-03T15:39:39      0      0     1000       1       0   0.300 sakila.film_category
04-03T15:39:39      0      0     1000       1       0   0.048 sakila.film_text
04-03T15:39:39      0      0     4581       1       0   0.301 sakila.inventory
04-03T15:39:39      0      0        6       1       0   0.083 sakila.language
04-03T15:39:40      0      1    16049       1       0   0.381 sakila.payment
04-03T15:39:40      0      0    16044       1       0   0.355 sakila.rental
04-03T15:39:40      0      0        2       1       0   0.034 sakila.staff
04-03T15:39:40      0      0        2       1       0   0.056 sakila.store
```

How to check?

- Before 5.6, nothing was provided by the server
- Your best bet was to use pt-table-checksum
  - http://www.percona.com/doc/percona-toolkit/2.2/pt-table-checksum.html

```
[stephane@centos2 ~]$ pt-table-checksum --empty-replicate-table -d sakila h=127.0.0.1,P=22585
            TS ERRORS  DIFFS     ROWS  CHUNKS SKIPPED    TIME TABLE
04-03T15:39:37      0      0      200       1       0   0.281 sakila.actor
04-03T15:39:38      0      0      603       1       0   0.299 sakila.address
04-03T15:39:38      0      0       16       1       0   0.047 sakila.category
04-03T15:39:38      0      0      600       1       0   0.049 sakila.city
04-03T15:39:38      0      0      109       1       0   0.049 sakila.country
04-03T15:39:38      0      0      599       1       0   0.302 sakila.customer
04-03T15:39:38      0      0     1000       1       0   0.292 sakila.film
04-03T15:39:39      0      0     5462       1       0   0.307 sakila.film_actor
04-03T15:39:39      0      0     1000       1       0   0.300 sakila.film_category
04-03T15:39:39      0      0     1000       1       0   0.048 sakila.film_text
04-03T15:39:39      0      0     4581       1       0   0.301 sakila.inventory
04-03T15:39:39      0      0        6       1       0   0.083 sakila.language
04-03T15:39:40      0      1    16049       1       0   0.3[illegible]1 sakila.payment
04-03T15:39:40      0      0    16044       1       0   0.355 sakila.rental
04-03T15:39:40      0      0        2       1       0   0.034 sakila.staff
04-03T15:39:40      0      0        2       1       0   0.056 sakila.store
```

Checksums on slaves

- Step #1: enable checksums on master
  - `binlog_checksum = CRC32;`
- Step #2: enable checks on slaves
  - `slave_sql_verify_checksum = ON;`
- Both are enabled by default

In case of relay log corruption

In case of relay log corruption

```
mysql> show slave status\G

               [...]
 Slave_IO_Running: Yes
Slave_SQL_Running: No
           [...]
       Last_Errno: 1594
       Last_Error: Relay log read failure: Could not parse relay log event entry.
The possible reasons are: the master's binary log is corrupted (you can check this
by running 'mysqlbinlog' on the binary log), the slave's relay log is corrupted (you
can check this by running 'mysqlbinlog' on the relay log), a network problem, or a
bug in the master's or slave's MySQL code. If you want to check the master's binary
log or slave's relay log, you will be able to know their names by issuing 'SHOW
SLAVE STATUS' on this slave.
```

- For a more meaningful error message, use mysqlbinlog

```
# mysqlbinlog --verify-binlog-checksum mysql_sandbox18676-relay-bin.000002
[...]
ERROR: Error in Log_event::read_log_event(): 'Event crc check failed! Most likely
there is event corruption.', data_len: 103, event_type: 2
ERROR: Could not read entry at offset 283: Error in log format or read error.
[...]
```

In case of relay log corruption

```
mysql> show slave status\G

               [...]
 Slave_IO_Running: Yes
Slave_SQL_Running: No
           [...]
       Last_Errno: 1594
       Last_Error: Relay log read failure: Could not parse relay log event entry.
The possible reasons are: the master's binary log is corrupted (you can check this
by running 'mysqlbinlog' on the binary log), the slave's relay log is corrupted (you
can check this by running 'mysqlbinlog' on the relay log), a network problem, or a
bug in the master's or slave's MySQL code. If you want to check the master's binary
log or slave's relay log, you will be able to know their names by issuing 'SHOW
SLAVE STATUS' on this slave.
```

- For a more meaningful error message, use mysqlbinlog

```
# mysqlbinlog --verify-binlog-checksum mysql_sandbox18676-relay-bin.000002
[...]
ERROR: Error in Log_event::read_log_event(): 'Event crc check failed! Most likely
there is event corruption.', data_len: 103, event_type: 2
ERROR: Could not read entry at offset 283: Error in log format or read error.
[...]
```

In case of relay log corruption

```
mysql> show slave status\G

               [...]
 Slave_IO_Running: Yes
Slave_SQL_Running: No
           [...]
       Last_Errno: 1594
       Last_Error: Relay log read failure: Could not parse relay log event entry.
The possible reasons are: the master's binary log is corrupted (you can check this
by running 'mysqlbinlog' on the binary log), the slave's relay log is corrupted (you
can check this by running 'mysqlbinlog' on the relay log), a network problem, or a
bug in the master's or slave's MySQL code. If you want to check the master's binary
log or slave's relay log, you will be able to know their names by issuing 'SHOW
SLAVE STATUS' on this slave.
```

- For a more meaningful error message, use mysqlbinlog

```
# mysqlbinlog --verify-binlog-checksum mysql_sandbox18676-relay-bin.000002
[...]
ERROR: Error in Log_event::read_log_event(): 'Event crc check failed! Most likely
there is event corruption.', data_len: 103, event_type: 2
ERROR: Could not read entry at offset 283: Error in log format or read error.
[...]
```

Checksums on the master

- `master_verify_checksum = ON;`

- But... this setting won't prevent you from writing to a corrupted binlog file!

- You will have to test manually

```
mysql> SHOW BINLOG EVENTS IN 'mysql-bin.000006';

ERROR 1220 (HY000): Error when executing command SHOW
BINLOG EVENTS: Wrong offset or I/O error
```

Limitations

- Back to the 'What can go wrong' list
  - Non deterministic queries
  - Writes executed on slaves outside of replication
  - Wrong usage of replication filters
  - Using a non transactional storage engine
  - Binlog or relay log corruption
  - ...

Checksums only help here!

- CPU overhead

- Relay log corruption was already often detected

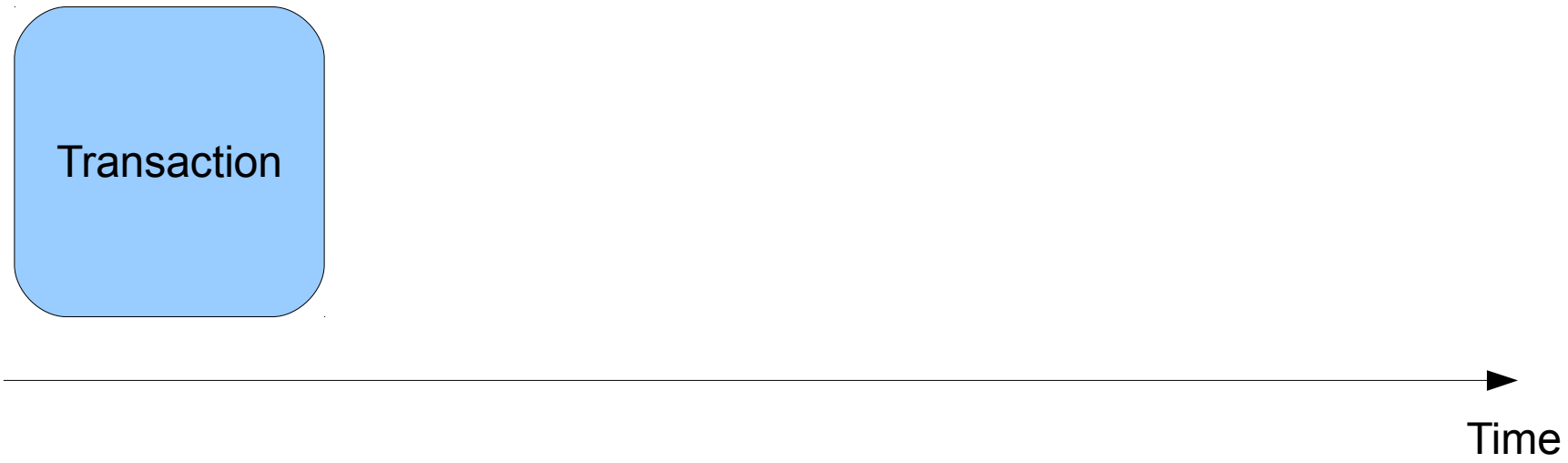
Crash safe slaves

What is the problem?

- Replication information is stored in files by default (relay-log.info and master.info)
- And updated after transaction is committed

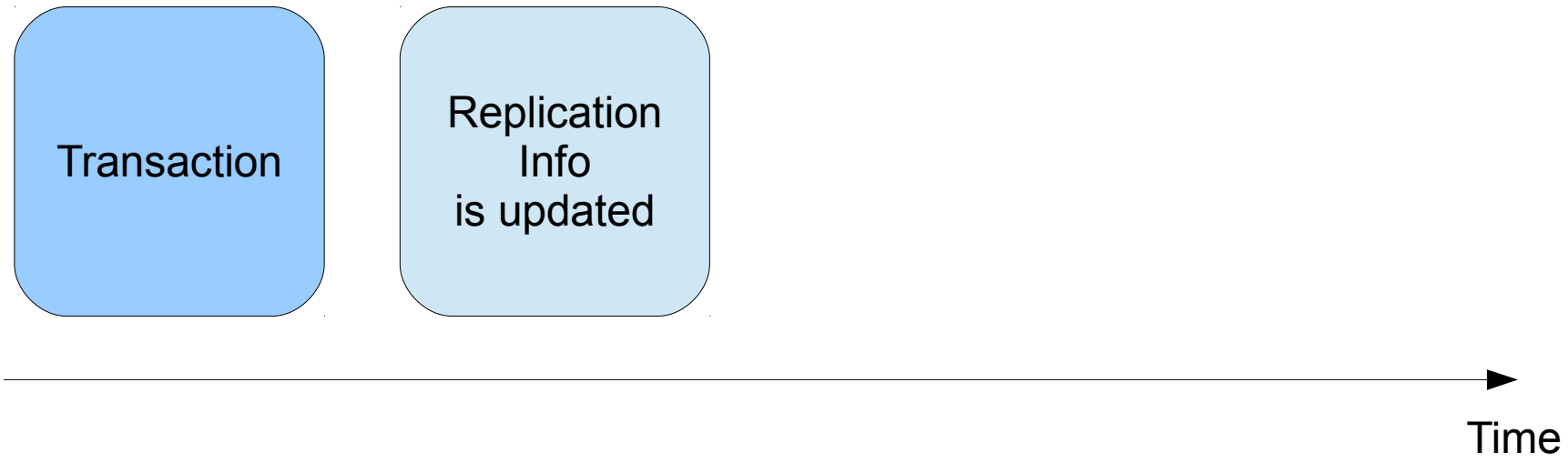
What is the problem?

- Replication information is stored in files by default (relay-log.info and master.info)
- And updated after transaction is committed



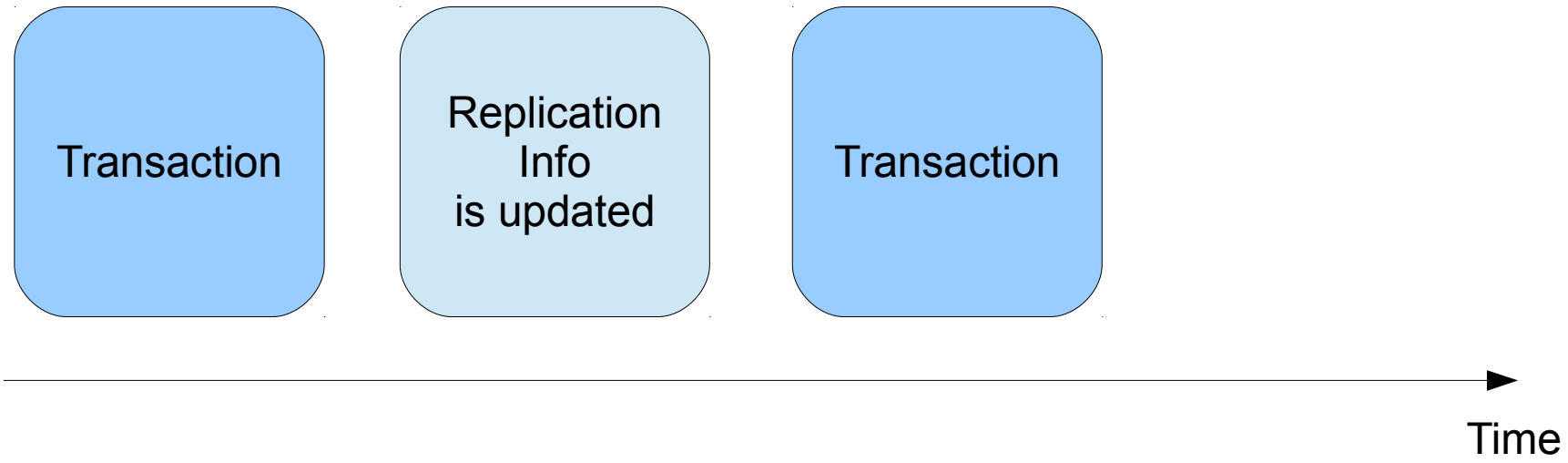
What is the problem?

- Replication information is stored in files by default (relay-log.info and master.info)
- And updated after transaction is committed



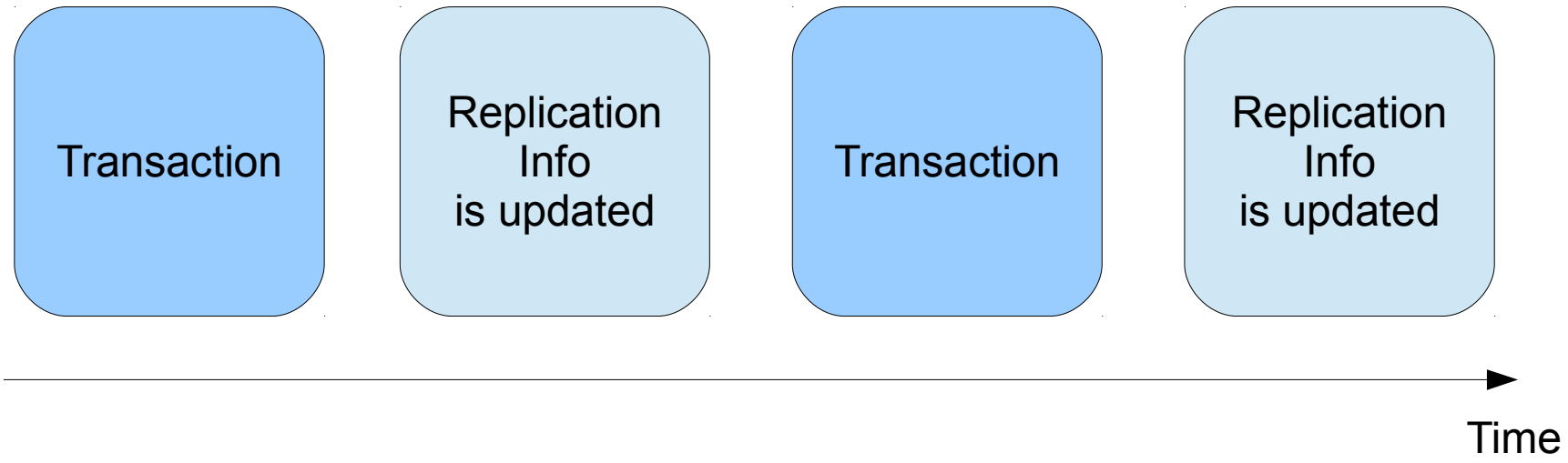
What is the problem?

- Replication information is stored in files by default (relay-log.info and master.info)
- And updated after transaction is committed



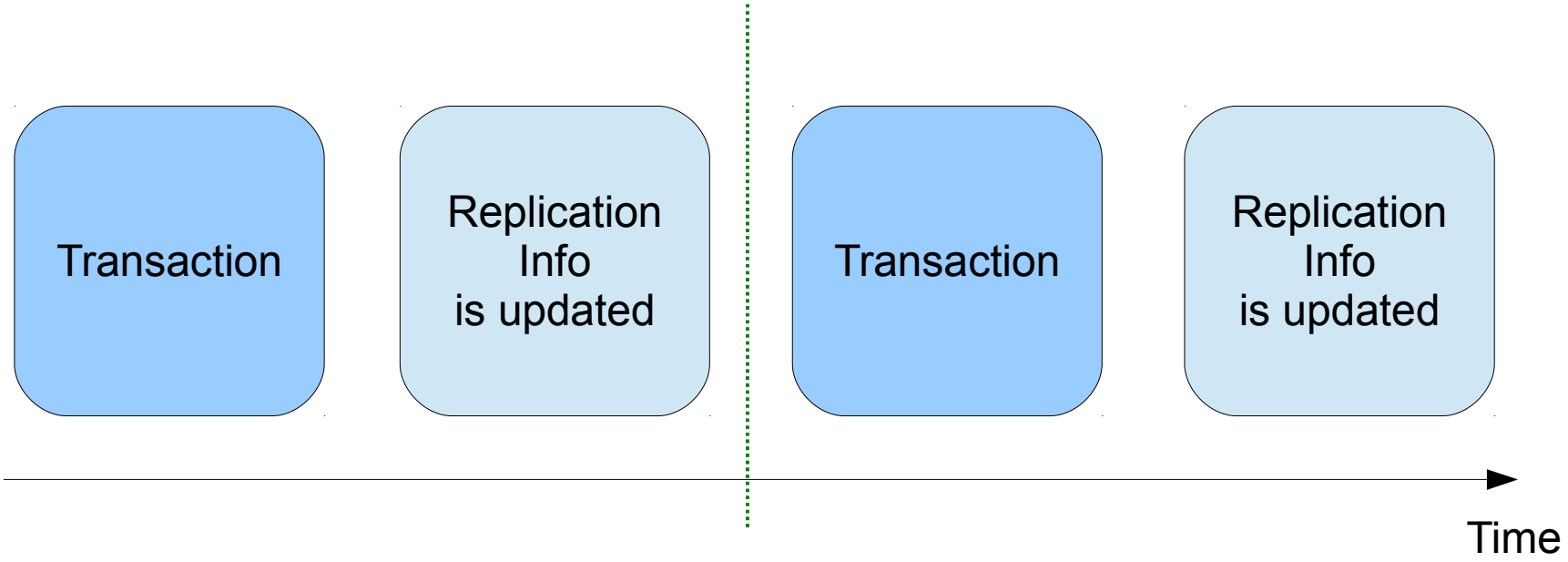
What is the problem?

- Replication information is stored in files by default (relay-log.info and master.info)
- And updated after transaction is committed



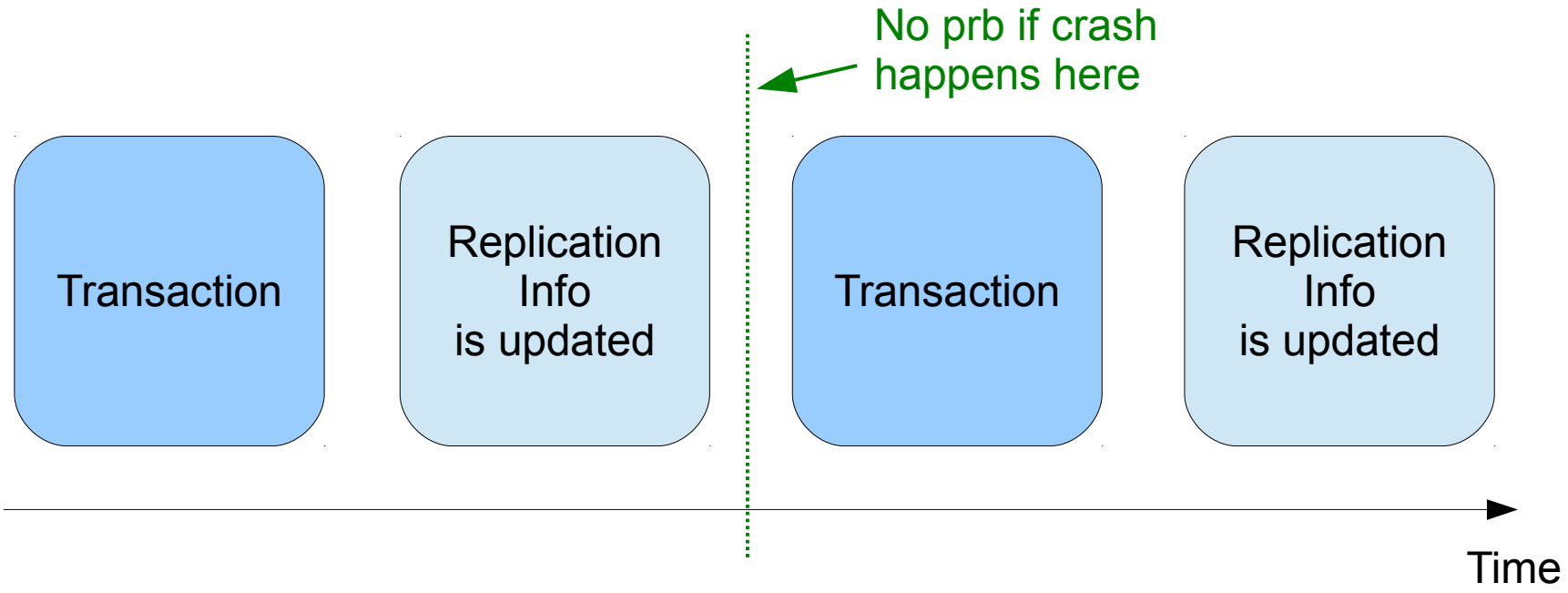
What is the problem?

- Replication information is stored in files by default (relay-log.info and master.info)
- And updated after transaction is committed



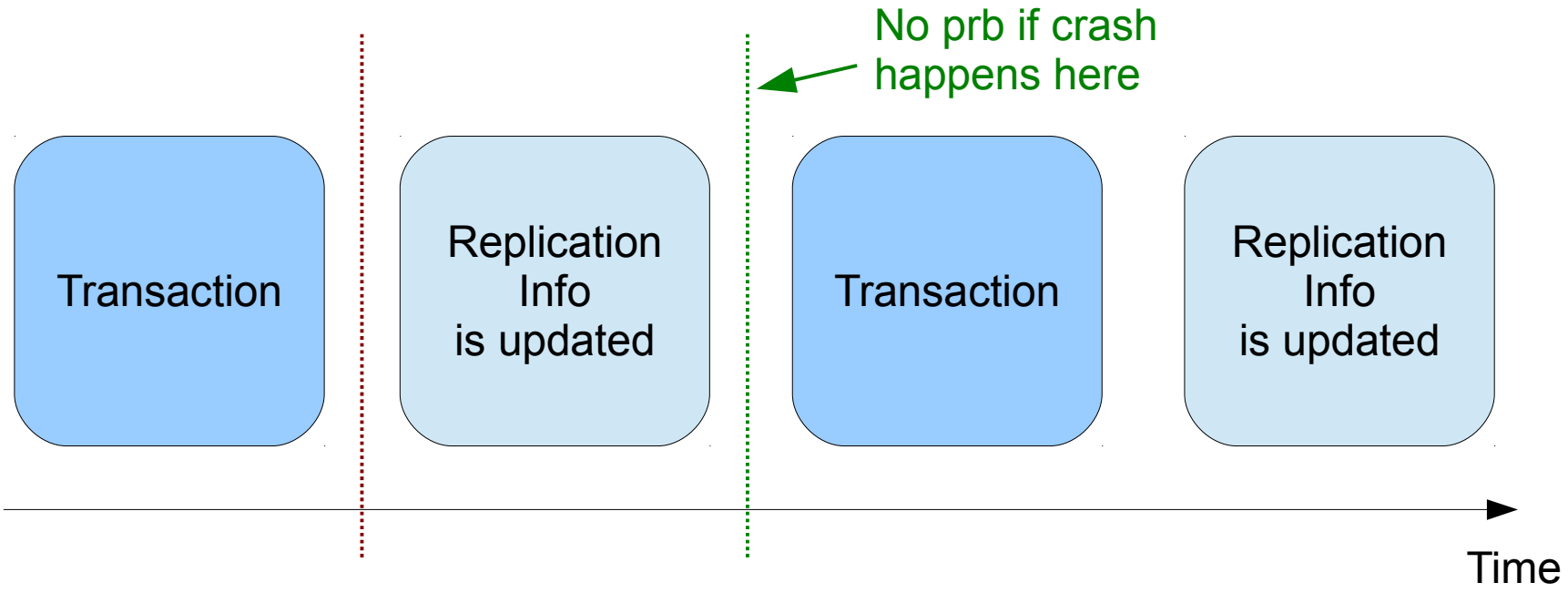
What is the problem?

- Replication information is stored in files by default (relay-log.info and master.info)
- And updated after transaction is committed



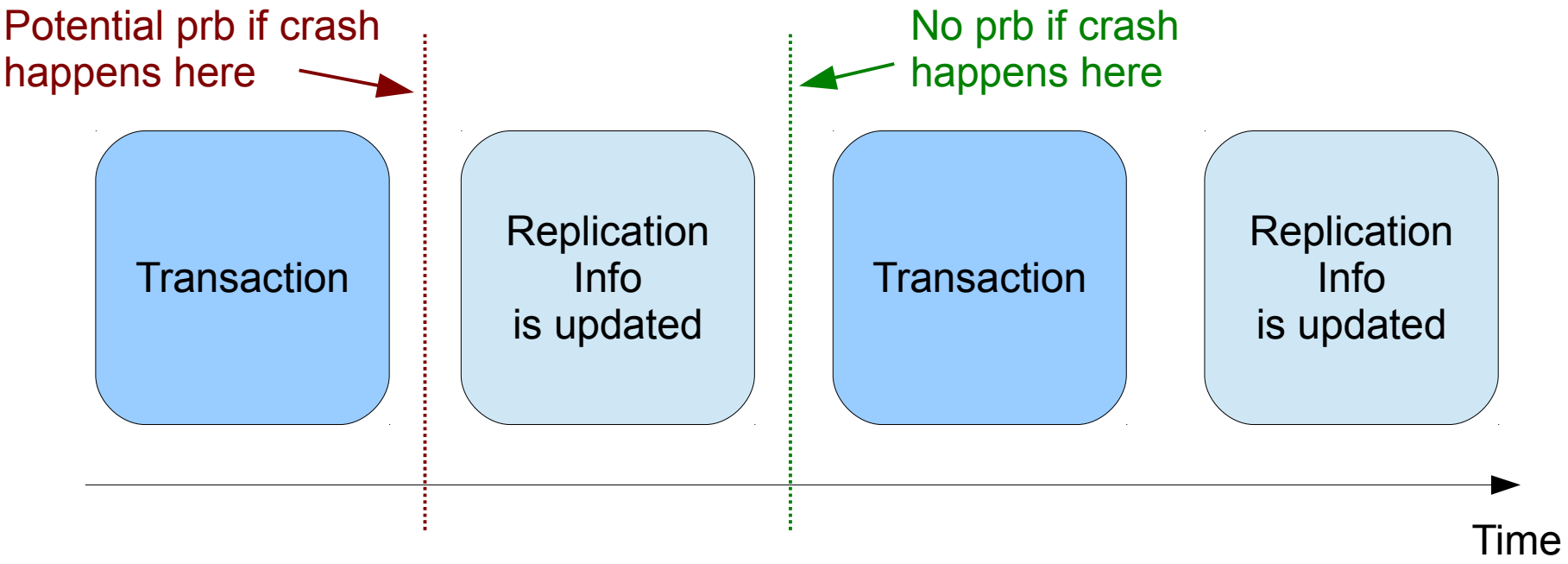
What is the problem?

- Replication information is stored in files by default (relay-log.info and master.info)
- And updated after transaction is committed



What is the problem?

- Replication information is stored in files by default (relay-log.info and master.info)
- And updated after transaction is committed

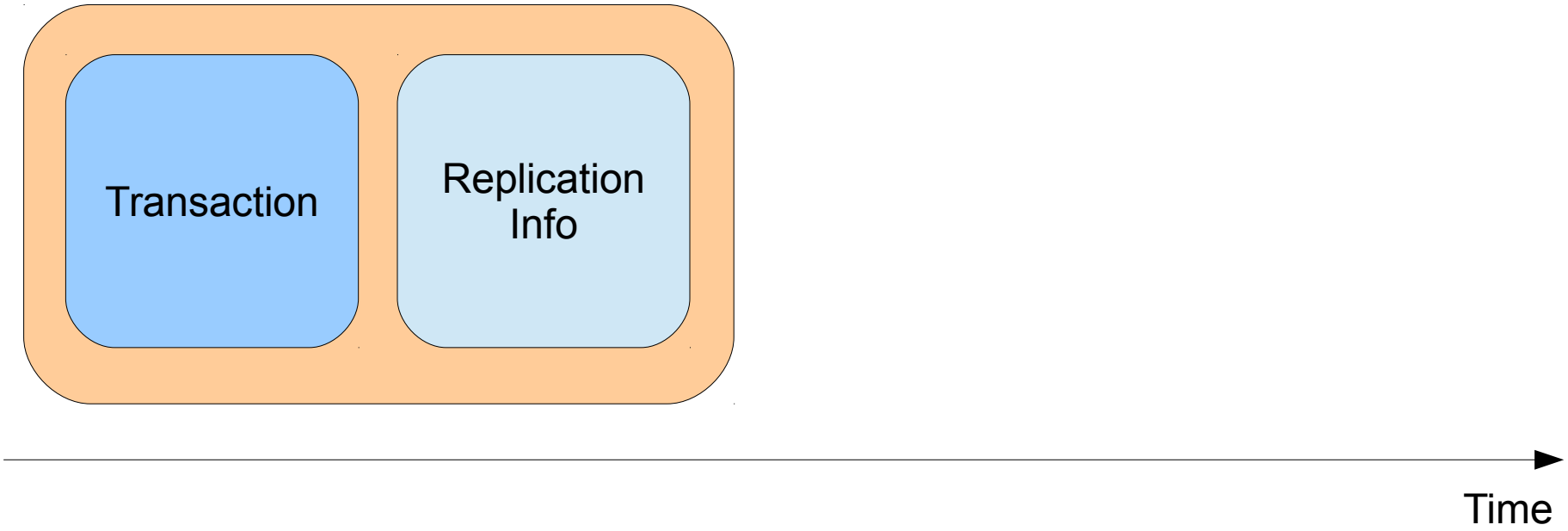


Making slaves crash safe

- Updating replication information is now part of the transaction

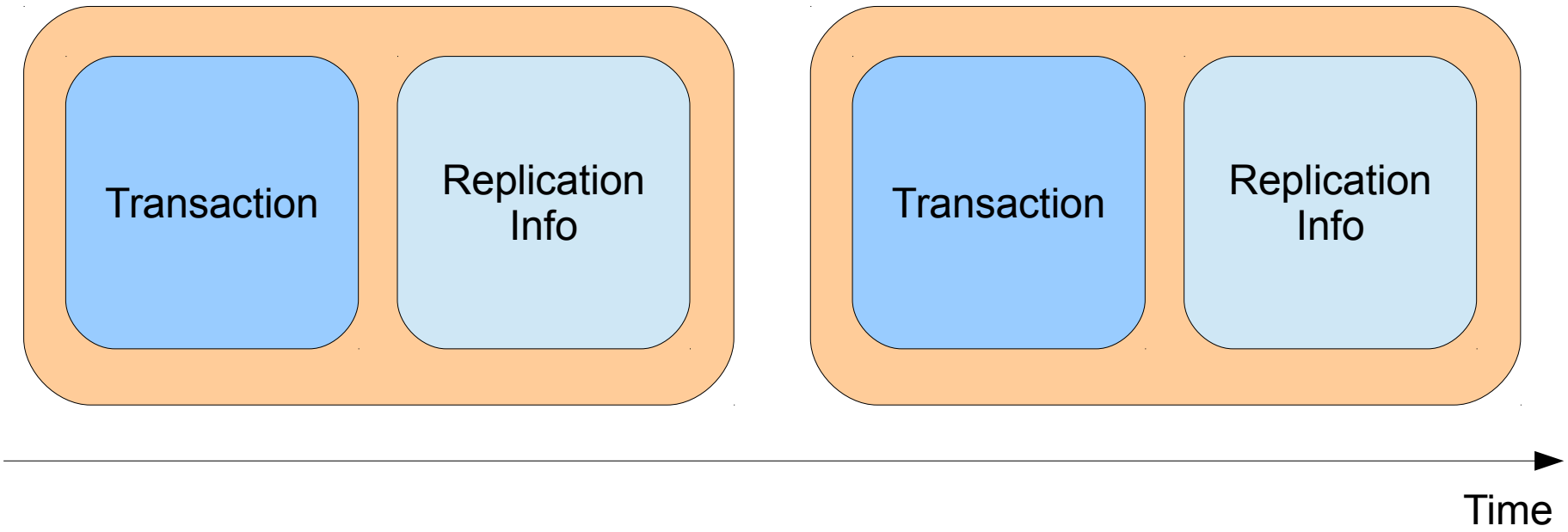
Making slaves crash safe

- Updating replication information is now part of the transaction



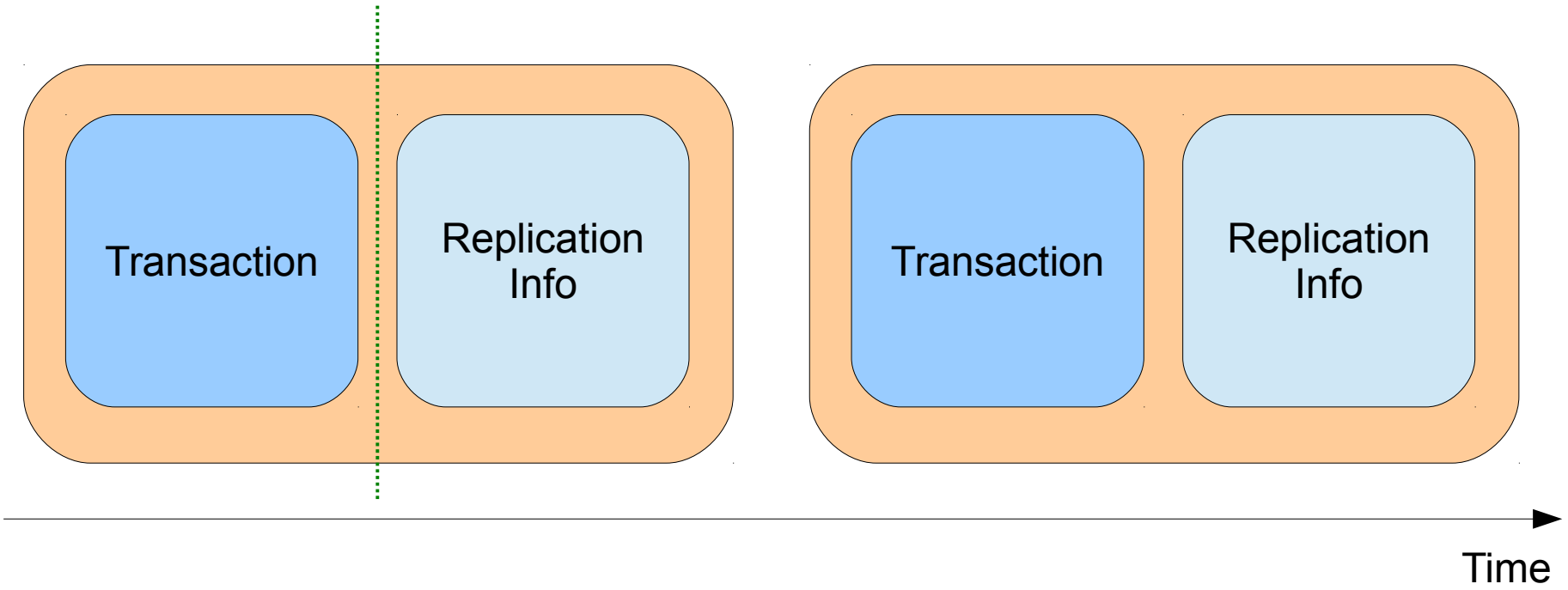
Making slaves crash safe

- Updating replication information is now part of the transaction



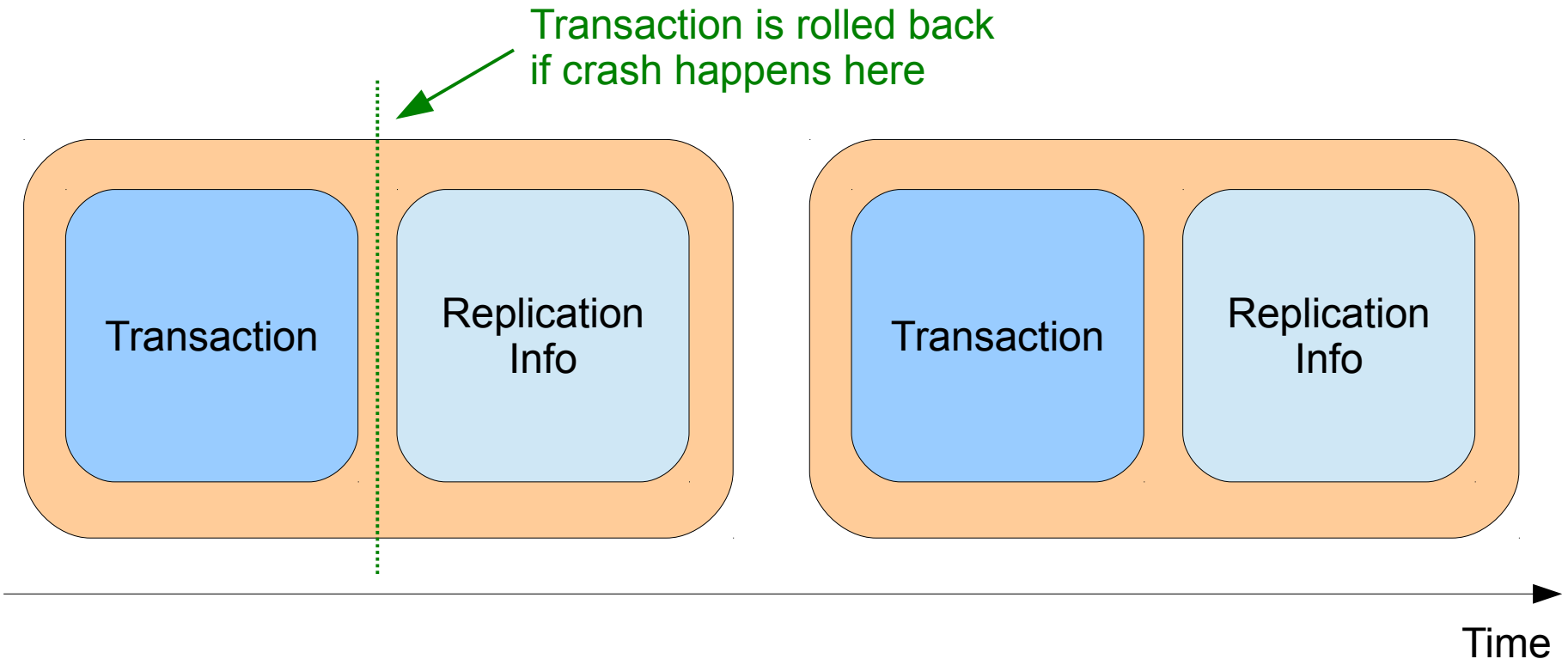
Making slaves crash safe

- Updating replication information is now part of the transaction



Making slaves crash safe

- Updating replication information is now part of the transaction



Settings to change

- You will need
  - `relay_log_info_repository = TABLE (*)`
  - `master_info_repository = TABLE (*)`
  - `relay_log_recovery = ON (+)`
- (*) Needs to run `STOP SLAVE` first
- (+) Needs to restart MySQL

relay_log_recovery

- By default, slave_master_info table is not updated at each transaction
  - Except if you set sync_master_info = 1
  - But it can have a performance impact
- With relay_log_recovery, at server startup
  - All unprocessed relay logs are discarded
  - Replication coordinates are recovered from the slave_relay_log_info table
  - It removes the need to have sync_master_info = 1

For users of Percona Server

- A similar feature exists in Percona Server
  - Turn on `innodb_recovery_update_relay_log` (5.5)
  - Turn on `innodb_overwrite_relay_log_info` (5.1)

- Oracle's crash safe slaves obsoletes this feature in Percona Server 5.6

Multi-threaded slaves

What is the problem?

- On the slave, replication uses 1 thread to run queries (the SQL thread)
- But on the master writes can be done in parallel
- A moderately loaded master can easily make your slaves lag
- Common workarounds: SSDs, relaxed durability, prefetcher tools

Adding parallelism

- Data must be partitioned across N databases
- Set `slave_parallel_workers = N`
- Use crash-safe replication setting to be able to correctly recover from crash

- The regular SQL_thread acts as a coordinator
- The other threads can execute trx in parallel if they come from different databases

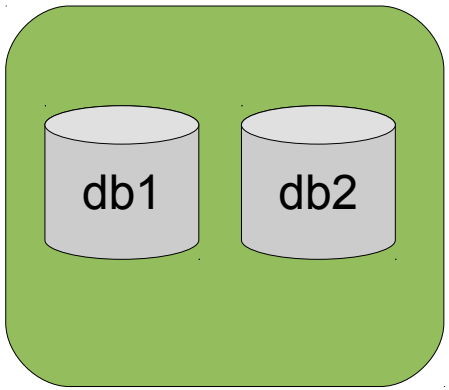
Visual explanation

Visual explanation

- Standard replication

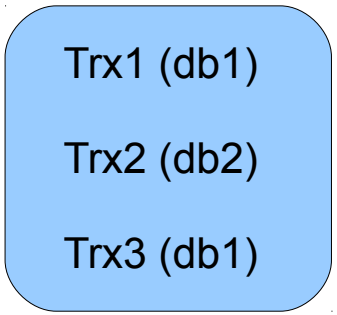
Visual explanation

- Standard replication

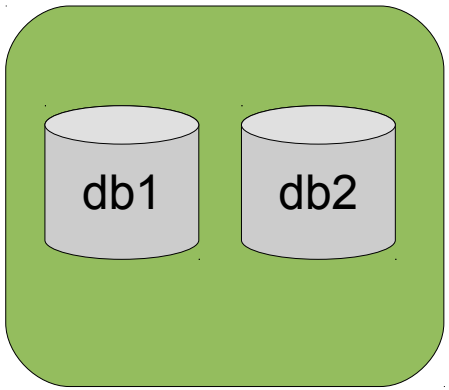


Visual explanation

- Standard replication



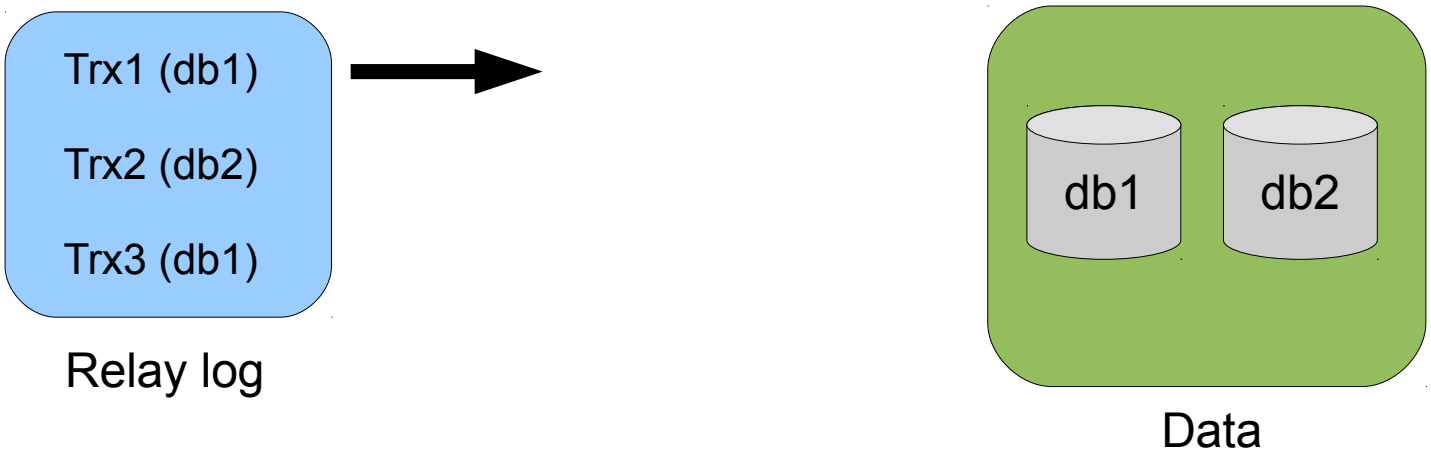
Relay log



Data

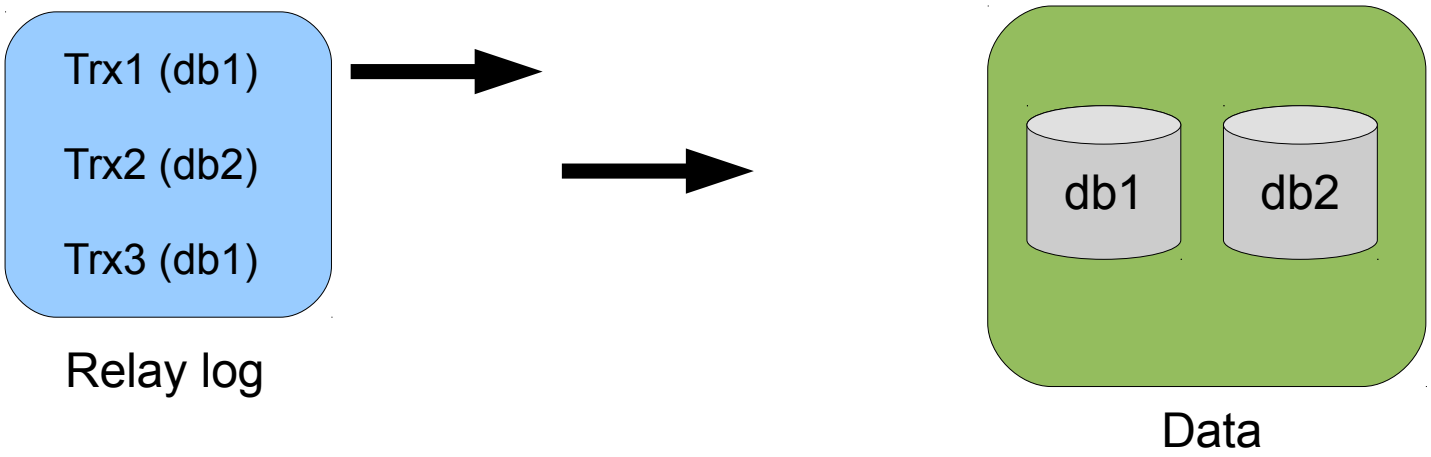
Visual explanation

- Standard replication



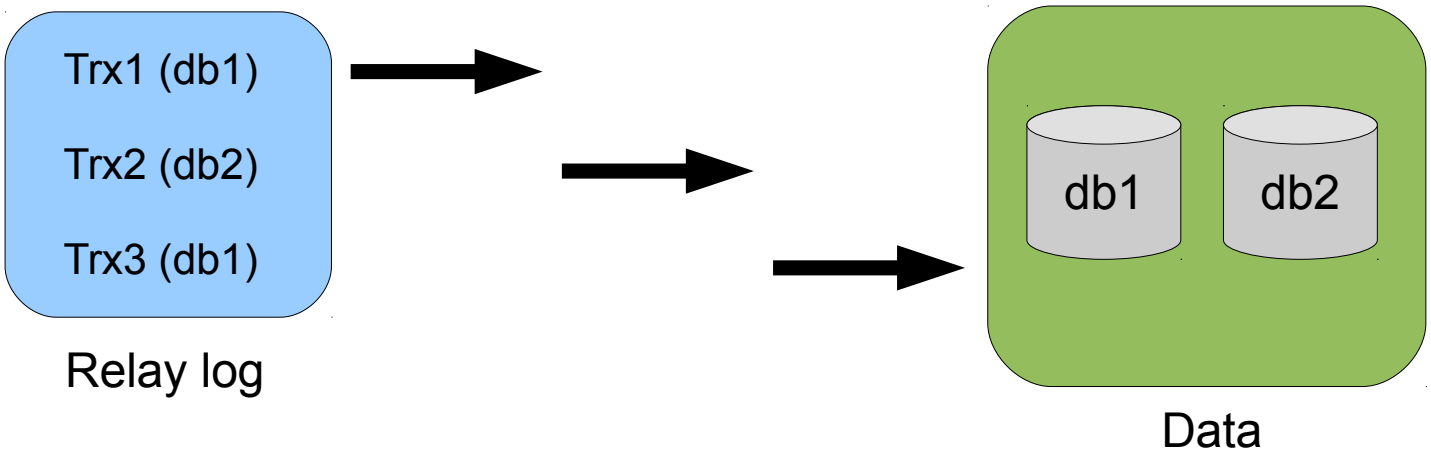
Visual explanation

- Standard replication



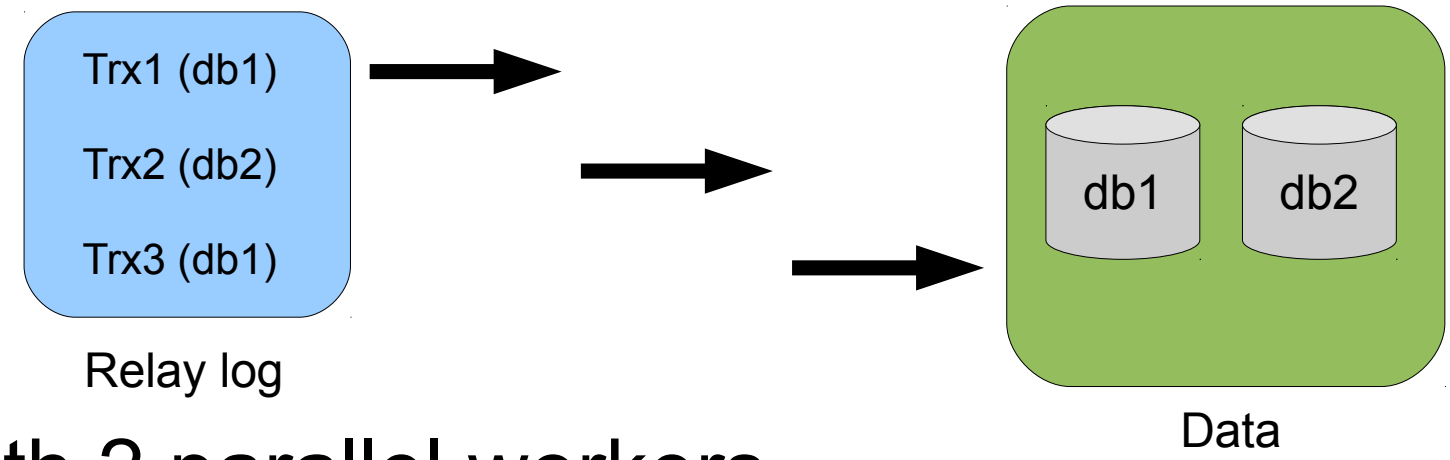
Visual explanation

- Standard replication



Visual explanation

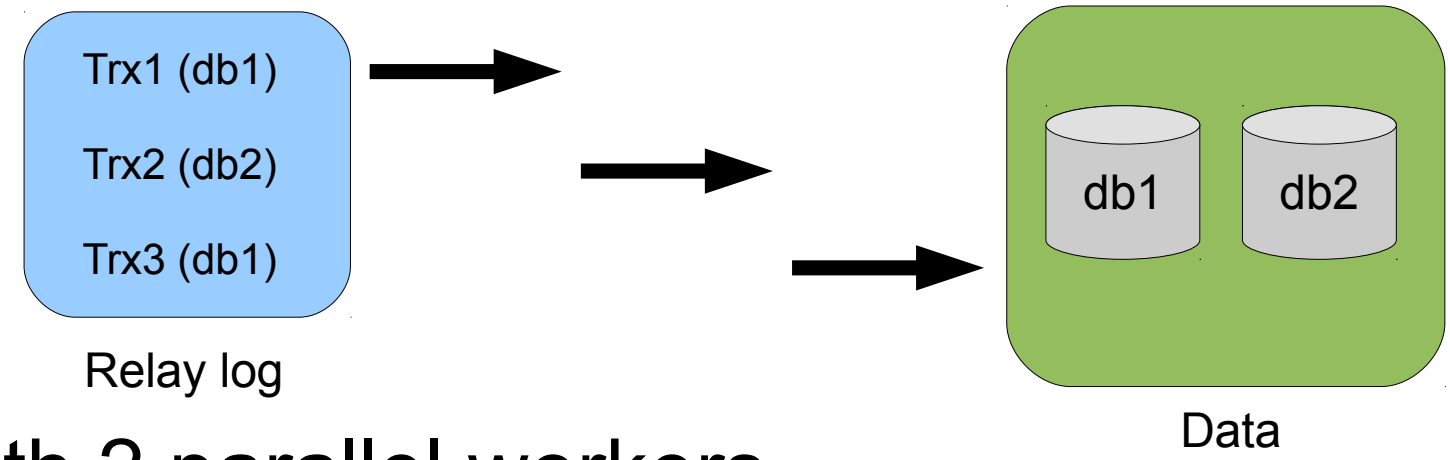
- Standard replication



- With 2 parallel workers

Visual explanation

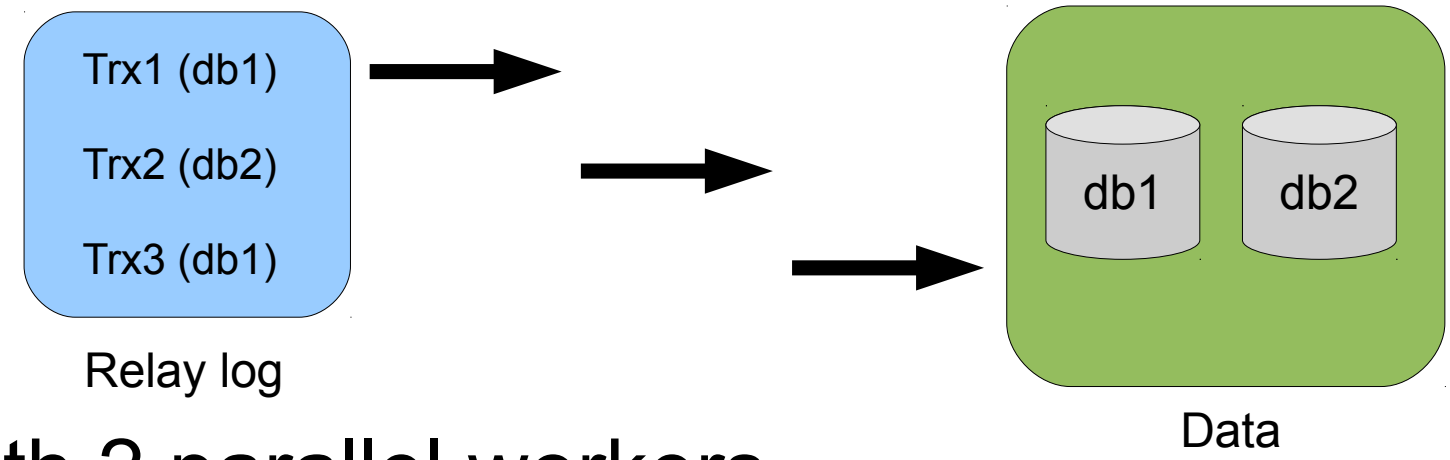
- Standard replication



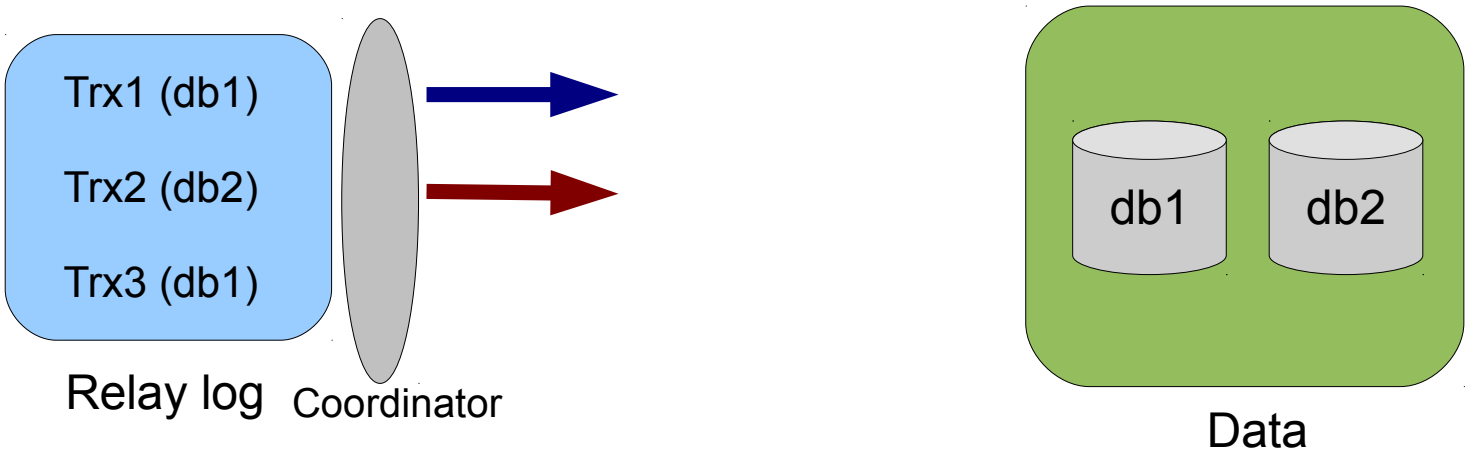
- With 2 parallel workers

Visual explanation

- Standard replication

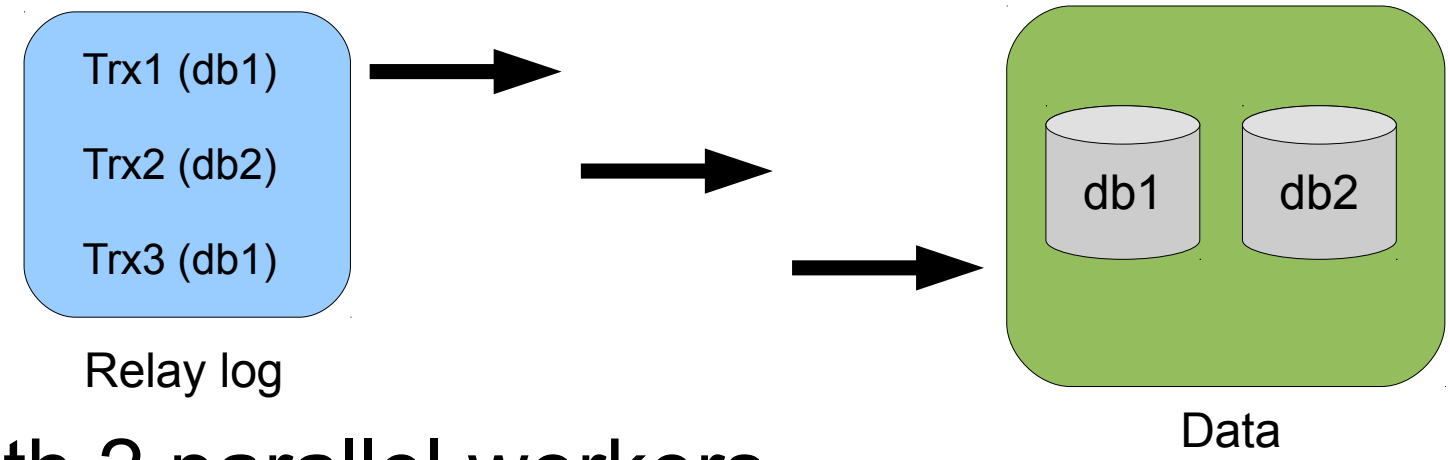


- With 2 parallel workers

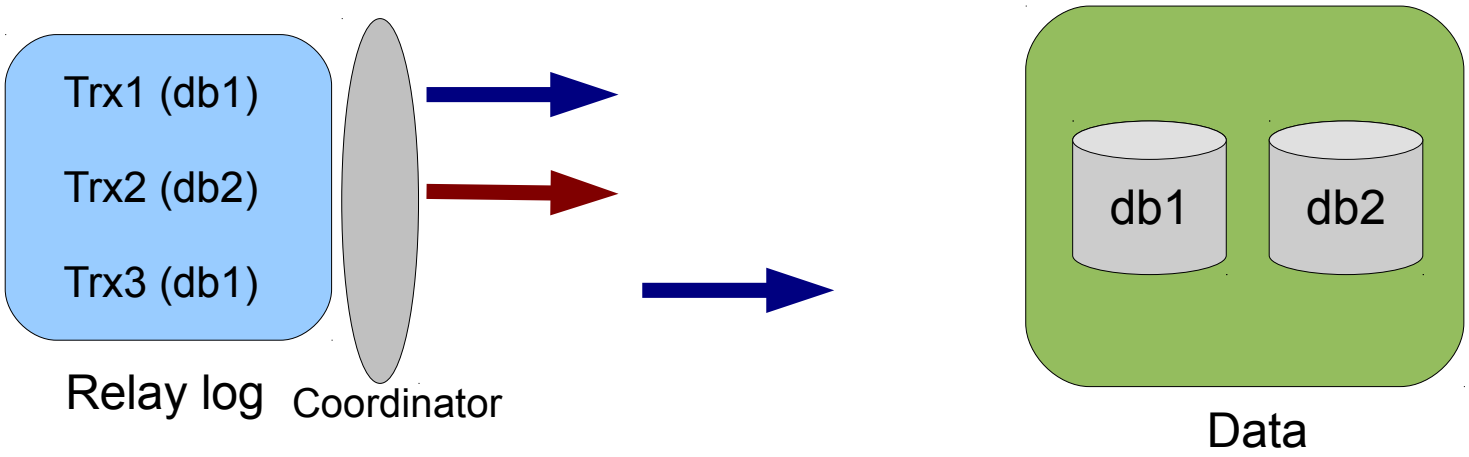


Visual explanation

- Standard replication



- With 2 parallel workers



Administrative table

```
 mysql> select * from mysql.slave_worker_info\G
****************** 2. row ***************
                          Id: 2
               Relay_log_name:
./mysql_sandbox19279-relay-bin.000034
                Relay_log_pos: 1038
              Master_log_name: mysql-bin.000005
               Master_log_pos: 1107
 Checkpoint_relay_log_name:
./mysql_sandbox19279-relay-bin.000034
  Checkpoint_relay_log_pos: 790
Checkpoint_master_log_name: mysql-bin.000005
 Checkpoint_master_log_pos: 859
             Checkpoint_seqno: 0
     Checkpoint_group_size: 64
   Checkpoint_group_bitmap:
```

Positions

- Transactions are not guaranteed to be executed in the order of the binlog
  - Not a problem as transactions are independent
  - Slave keeps track of such execution gaps
- The position reported by `SHOW SLAVE STATUS` is not accurate
  - It is a low-water mark corresponding to the latest checkpoint (see `slave_checkpoint_period` setting)
  - Executed transactions since latest checkpoint are recorded for recovery

Limitations

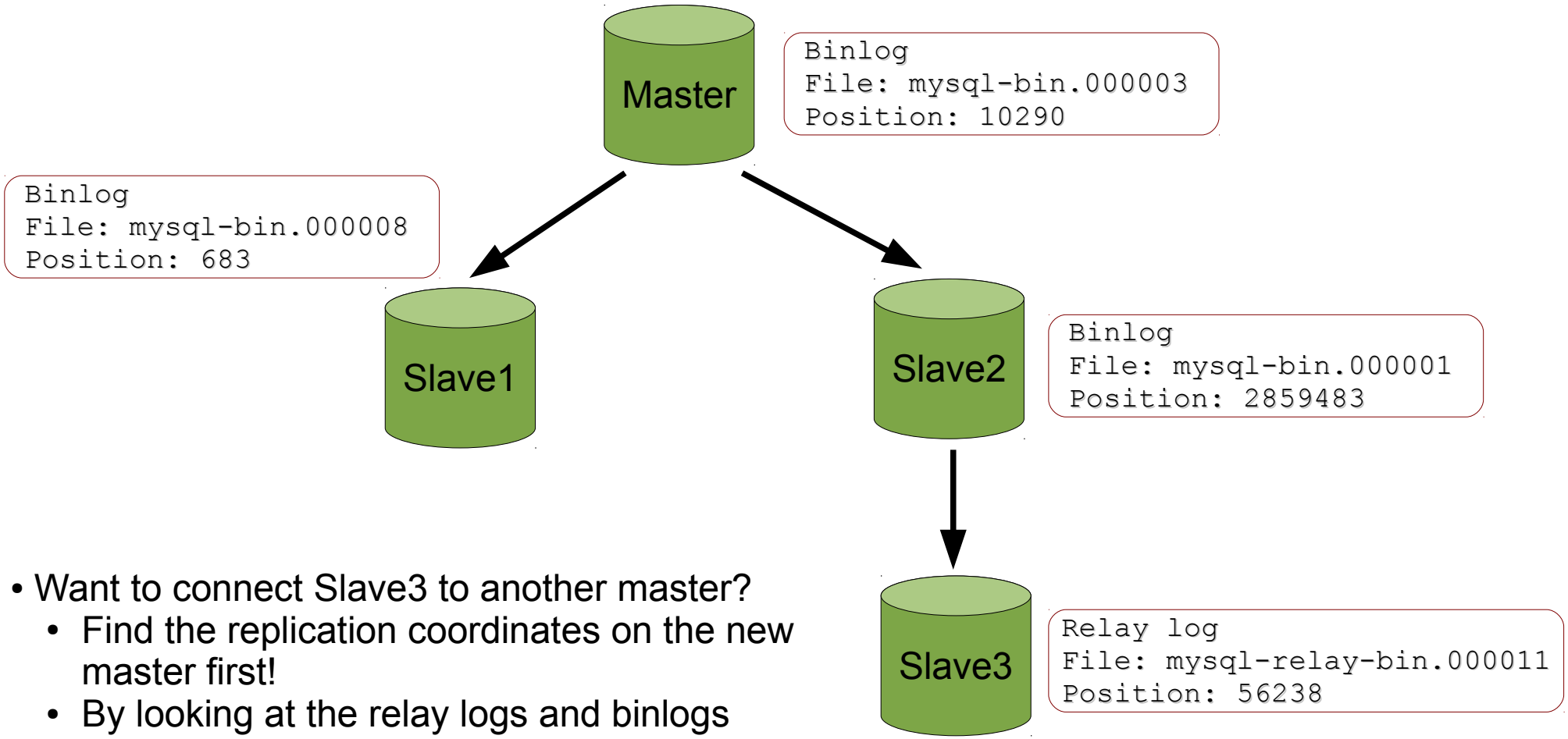
- Only useful if data is partitioned across N dbs
- And if there are only cross-db transactions
- If N is too high, you're likely to hit bottlenecks (CPU, disks, internal implementation)
- `SHOW SLAVE STATUS` output must be taken with care

Global Transactions IDs

What is the problem?

- The same event is very likely to have different binary log file and binary log position on 2 separate servers
- That makes it hard and error-prone to reconfigure replication

Position-based replication

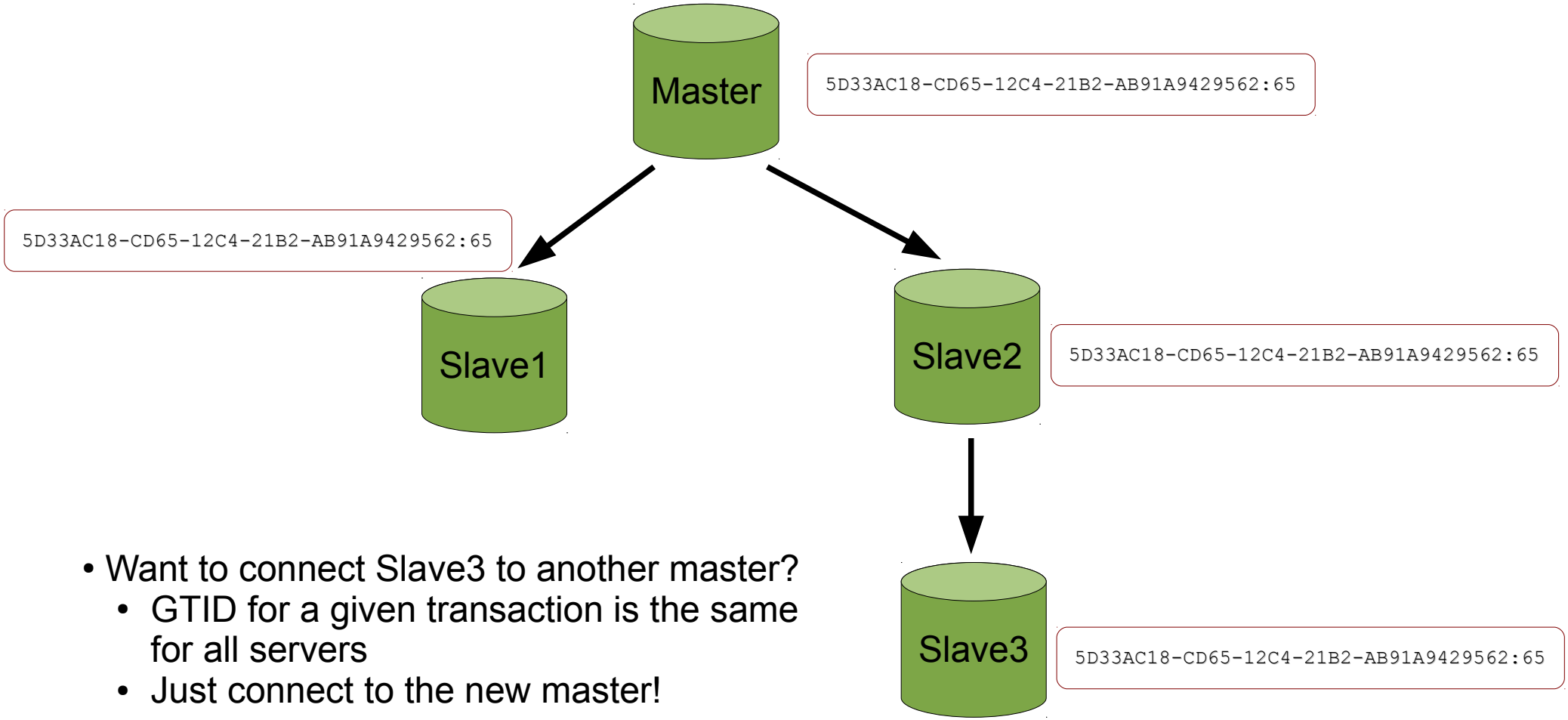


- Want to connect Slave3 to another master?
  - Find the replication coordinates on the new master first!
  - By looking at the relay logs and binlogs
  - Tedious and error-prone

What is a GTID?

- It is the unique identifier of a transaction across all servers of a replication setup
- A GTID has 2 parts
  - source_id:transaction_id
- A sequence of GTIDs is simply
  - source_id:trx_start-trx_stop
  - Eg `3E11FA47-71CA-11E1-9E33-C80AA9429562:1-5`

GTID-based replication



- Want to connect Slave3 to another master?
  - GTID for a given transaction is the same for all servers
  - Just connect to the new master!

GTID-based replication setup (1)

- Enable read-only mode and make sure the slaves are in sync
  - `SET GLOBAL read_only=1;`
  - Compare output of `SHOW SLAVE STATUS` with output of `SHOW MASTER STATUS`
  - Positions should be the same

GTID-based replication setup (2)

- Change configuration for all servers

```
log-bin
log-slave-updates
gtid_mode=ON
enforce-gtid-consistency
```

- Restart them
- Instruct all servers to use GTIDs
  - `CHANGE MASTER TO MASTER_AUTO_POSITION = 1;`
- Disable read-only mode

Limitations

- Downtime required when switching to GTIDs
- Binary logging should be enabled on all slaves
- Not possible to use temporary tables inside transactions
- Monitoring tables for crash-safe slaves and multi-threaded replication not ready for GTIDs

Other features

Also included in MySQL 5.6

- Binary log group commit
- Optimized row-based replication
- MySQL Utilities
  - Python CLI tools, not limited to replication
- Time-delayed replication
- Remote binlog backups

Q&A

Thanks for attending!

Feel free to contact me:
stephane.combaudon@percona.com

Percona Live London 2013

November 11-12, 2013

Millennium Gloucester Conference Centre

http://www.percona.com/live/london-2013/