Demystifying MySQL Replication Crash Safety

by Jean-François Gagné
System and MySQL Expert at HubSpot
Presented at MinervaDB Athena 2020
jfg DOT mysql AT gmail.com
Twitter: @jfg956

Summary

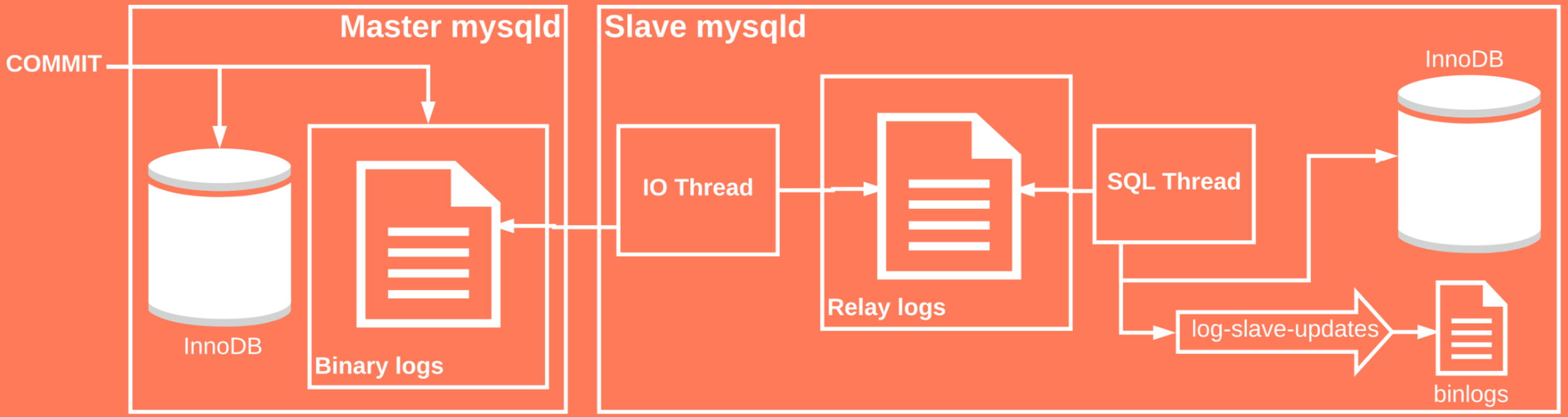
(Demystifying MySQL Replication Crash Safety – MinervaDB Athena 2020)

- High level view of – and then Zoom in – Replication
- Crash Safety / Unsafety, and MySQL 5.6 solution (and its problems)
- Complexifying things with GTIDs and Multi-Threaded Slave
- Impacts of reducing durability (***sync_binlog != 1*** *and* ***innodb_flush_log_at_trx_commit != 1***)
- Overview of related subjects: MariaDB, Pseudo-GTIDs, and Complexity
- Closing with links, bugs and questions

Overview of MySQL Replication

(Demystifying MySQL Replication Crash Safety – MinervaDB Athena 2020)

- The master records transactions in a journal (binary logs), and slaves
  - Downloads the journal and saves it locally in the relay logs (IO thread)
  - Executes the relay logs on its local database (SQL thread)
  - Could also write binlogs to be themselves a master (`log-slave-updates`)



Overview of Replication Crash Safety

(Demystifying MySQL Replication Crash Safety – MinervaDB Athena 2020)

What is Replication Crash Safety ?

- After a crash and recovery on a slave, replication can resume
  (OK if state rewinds after recovery, as long as it is eventually consistent)
- After a crash and recovery on a master, replication can resume
- All that without sacrificing data consistency (no data corruption / drift)

There are two types of replication crash safety: slave and master !

Intermediate masters qualify both as master and slave

This discussion is limited to transactional Storage Engines:
InnoDB, TokuDB and MyRocks (obviously not MyISAM)

(Proving replication crash unsafety is easy, proving safety is hard)

State of the Dolphin and Sea Lion

(Demystifying MySQL Replication Crash Safety – MinervaDB Athena 2020)

State of the Dolphin for Slave Replication Crash Safety

- MySQL 5.5 is not crash-safe
- MySQL 5.6 can be made crash-safe (not the default)
- MySQL 5.7 is mostly the same as 5.6 (with complexity added by Logical Lock)
- MySQL 8.0 is still not crash-safe by default

Quick state of the Sea Lion for Slave Replication Crash Safety

- MariaDB 5.5 is not crash-safe
- MariaDB 10.x can be made crash-safe (not default)

All can be made master crash-safe,
but only MySQL 8.0 is master crash-safe by default

Zoom in Replication Details [1 of 3]

(Demystifying MySQL Replication Crash Safety – MinervaDB Athena 2020)

The state of the IO Thread is stored in *master info*
The state of the SQL Thread is stored in *relay log info*

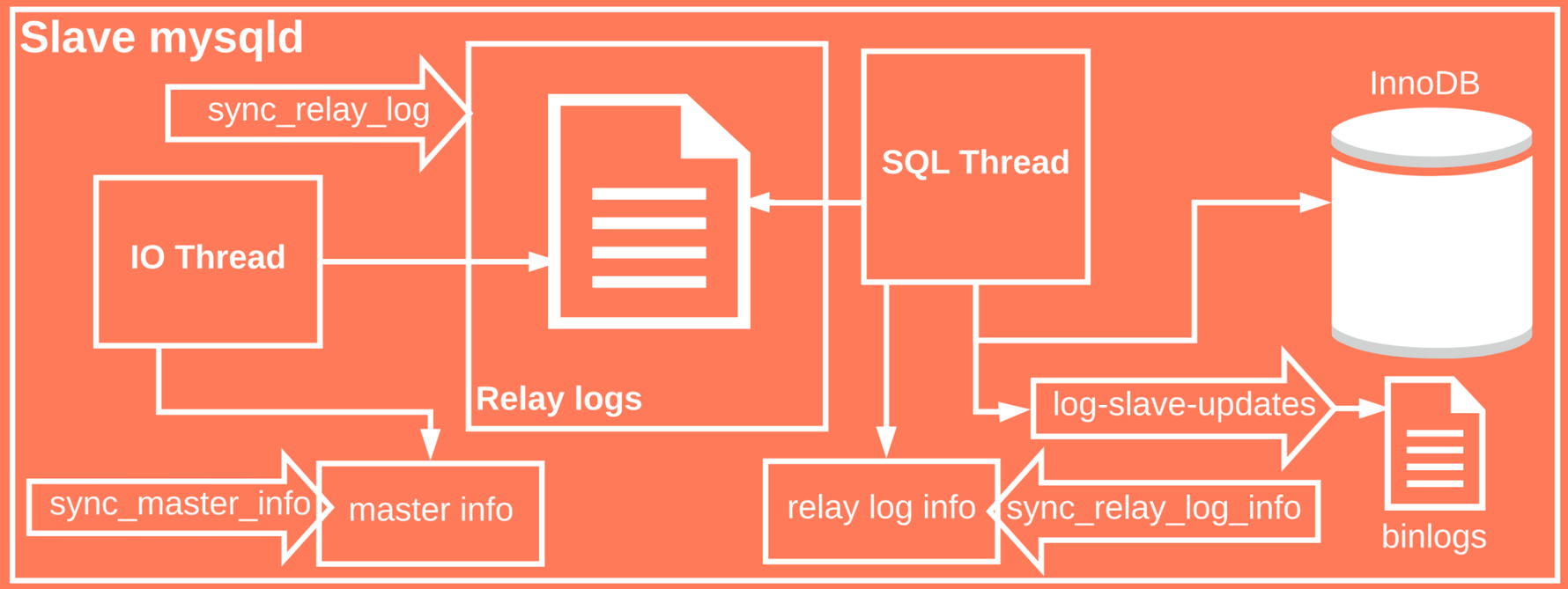
```
slave1 [localhost] {msandbox} ((none)) > show slave status\G
*************************** 1. row ***************************
               Slave_IO_State: Waiting for master to send event
[...]
                  Master_Log_File: mysql-bin.000001  <-------+-- master info
              Read_Master_Log_Pos: 25489               <-------+
                   Relay_Log_File: mysql-relay.000002  <--+
                    Relay_Log_Pos: 10788               <--+
            Relay_Master_Log_File: mysql-bin.000001    <--+-- relay log info
[...]                                                     |
              Exec_Master_Log_Pos: 10575               <--+
[...]
1 row in set (0.00 sec)
```

Zoom in Replication Details [2 of 3]

(Demystifying MySQL Replication Crash Safety – MinervaDB Athena 2020)

In MySQL 5.5, *master info* and *relay log info* are stored in files

- No atomicity of “progress” and “tracking” of the IO and SQL Threads
- Flushing the relay logs is expensive (not flushed by default in 5.5)
- No consistency of actual vs registered state → not crash-safe



Zoom in Replication Details [3 of 3]

(Demystifying MySQL Replication Crash Safety – MinervaDB Athena 2020)

Even more parameters: **`sync_binlog`** (and **`trx_commit`**)
(we will come back to these later for impact of reducing durability)
(**`trx_commit`** is short for **`innodb_flush_log_at_trx_commit`**)

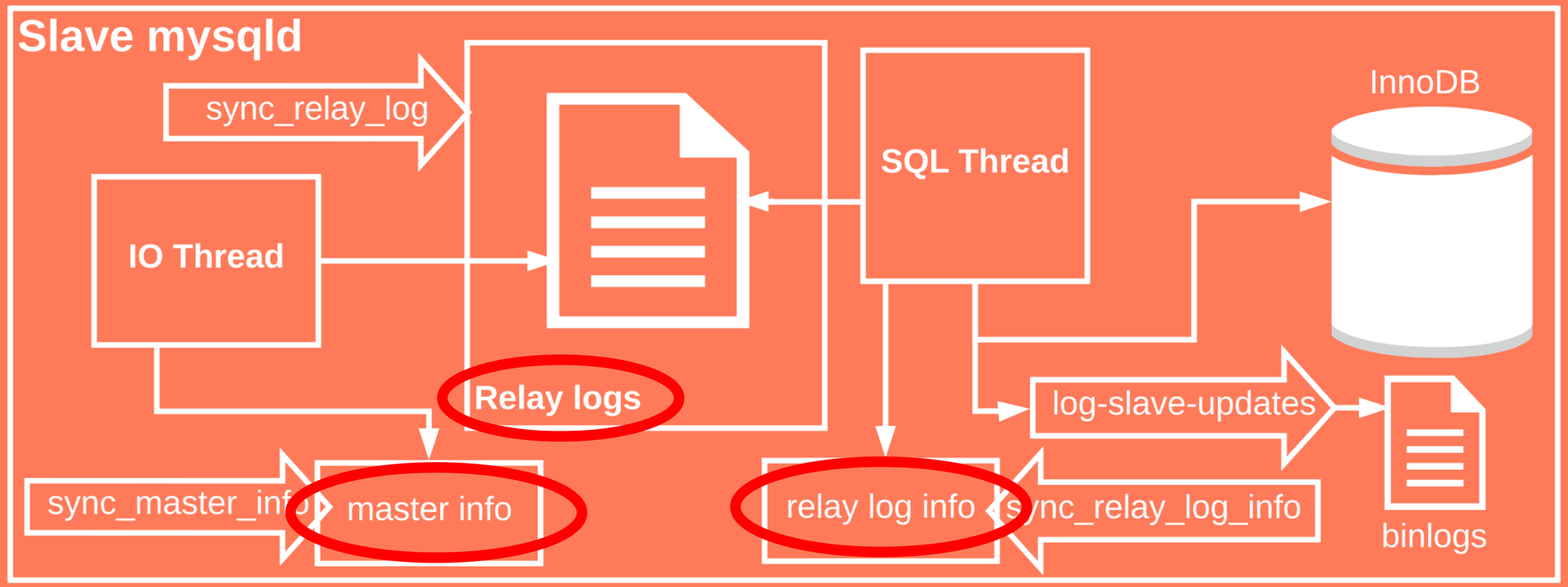
- **`sync_binlog = N`**: binlogs are flushed every N transactions (commit group)
  - default 0 in MySQL 5.5 and 5.6 → not durable 😐
  - default 1 in MySQL 5.7 and 8.0 → fully durable, and safer 🙂 (but slower)
  - default 0 in MariaDB 5.5 and 10.0 😐; and also 0 in 10.{1,2,3,4,5} 😕
- **`trx_commit = 1`**: InnoDB Redo logs are flushed after each trx (default 🙂)
  - **`= 0`**: logs written and flushed every second (not durable)
  - **`= 2`**: logs written after each trx and flushed every second
    (durable and safe for mysqld crashes, not for OS crashes)

MySQL 5.6 Crash Safety [1 of 4]

(Demystifying MySQL Replication Crash Safety – MinervaDB Athena 2020)

Reminder: MySQL 5.5 is not replication crash-safe because

- The position of the IO and SQL Threads cannot be trusted after a crash
- The content of the relay logs cannot be trusted after a crash

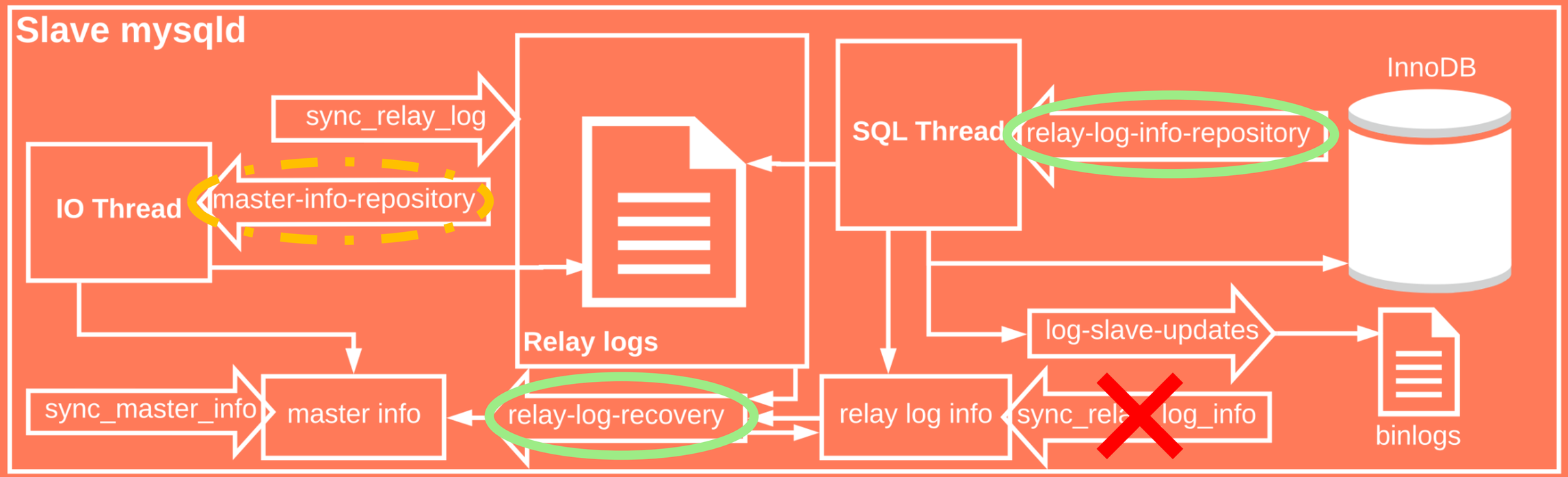


MySQL 5.6 Crash Safety [2 of 4]

(Demystifying MySQL Replication Crash Safety – MinervaDB Athena 2020)

MySQL 5.6 solution (crash safety for single thread and non-GTID slaves)

- Atomicity for SQL Thread: **`relay-log-info-repository = TABLE`** (default = **`FILE`**)
- Providing a way to “fix” the relay logs: **`relay-log-recovery = 1`** (default = 0)
- Useless: storing master info in a table (**`master-info-repository = TABLE | FILE`**)



MySQL 5.6 Crash Safety [3 of 4]

(Demystifying MySQL Replication Crash Safety – MinervaDB Athena 2020)

More details about *Relay Log Recovery*

- Relay log recovery is only used on mysqld startup (dynamic would be useless)
- If **`relay-log-recovery = 0`**, nothing special is done (and a new relay log is created)
- If **`relay-log-recovery = 1`**
  - The IO Thread position is rewond to the SQL Thread position (trustworthy)
  - The SQL Thread position is set to the newly created relay log
  - So the SQL Thread will only consume the new relays logs (trustworthy)
  - Said otherwise, the previous relay logs (untrustworthy) are skipped !
  - And this will happen even if MySQL (or the IO Thread) did not crash
    (OK for 1st implementation but waste of mostly good relay logs in many cases)
    (Remember that is is done on every MySQL startup, not only after crashes)

MySQL 5.6 Crash Safety [4 of 4]

(Demystifying MySQL Replication Crash Safety – MinervaDB Athena 2020)

In MySQL 5.7

- No change of defaults (for slave replication crash safety) 😐
- Relay log recovery still simplistic 😐

In MySQL 8.0

- Still simplistic relay log recovery 😕
- New defaults
  - `relay-log-info-repository = TABLE` 🙂
  - `master-info-repository = TABLE` (not sure this is very useful)
- Default for `relay-log-recovery` unchanged 😕 😕 😕 😕 😕

Bug#74323: Avoid overloading the master NIC on relay-log-recovery of a lagging slave
Bug#74321: Execute relay-log-recovery only when needed
Bug#93081: Please implement a better relay log recovery
Bug#101875: Make MySQL 8.0 replication crash-safe by default

Adding Complexity with GTIDs [1 of 4]

(Demystifying MySQL Replication Crash Safety – MinervaDB Athena 2020)

MySQL 5.6 also introduces Global Transaction IDs (*GTIDs*)

- This tags every transaction with a unique ID when writing to the binlogs
- The GTID state of the master and of slaves are tracked in the binlogs
- IO and SQL Thread states are now partially in the binlogs (and relay logs)

- Optionally, slaves can use GTID to replicate (instead of file and position)
- This allows easier repointing of slaves to a new master (including fail over)
- This relies on precise tracking of GTID states on master and slaves
- And this tracking it is affected when **`sync_binlog != 1`**

- GTID replication is not crash-safe on slaves with **`sync_binlog != 1`**
  → After an OS crash, a slave needs to be restored from backup

Bug#70659: Make crash-safe slave with gtid + less durable settings
Bug#101874: The manual is missing **`sync_binlog = 1`** for GTID replication crash safety

Adding Complexity with GTIDs [2 of 4]

(Demystifying MySQL Replication Crash Safety – MinervaDB Athena 2020)

To make a GTID slave replication crash-safe in MySQL 5.6

- `relay-log-info-repository = TABLE` (default = `FILE`)
- `relay-log-recovery = 1` (default = `0`)
- `sync_binlog = 1` (default = `0`)
- `innodb_flush_log_at_trx_commit = 1` (default = `1` 🙂)

- In 5.7, the default is `sync_binlog = 1` 🙂 (two other unchanged 😐)
- In 8.0, the missing default is `relay-log-recovery = 1` 😕😕😕😕😕
- MySQL 5.7 adds a table (`mysql.gtid_executed`) for the GTID state
  - Allows GTIDS slaves without binlogs or without `log-slave-updates` (*lsu*)
  - With lsu, this table is not updated after each trx, only at binlog rotation
  - Missed opportunity for OS crash safety with `sync_binlog != 1` 😕😕😕😕😕

Bug#92109: Make GTID replication crash-safe with less durable setting (bis)

Adding Complexity with GTIDs [3 of 4]

(Demystifying MySQL Replication Crash Safety – MinervaDB Athena 2020)

When running a GTID slave with **`sync_binlog != 1`**

- The GTID state is corrupted after an OS crash: you should restore a backup
- But file+position is trustworthy (in a table), so we might avoid the restore

How to save a GTID slave after an OS crash (this is voodoo)

- Make sure the slave starts with replication disabled (**`skip-slave-start`**)
- Note the GTID state (**`gtid_executed`**) and wipe the binlogs (**`RESET MASTER`**)
- Use with file+pos. (**`CHANGE MASTER TO MASTER_AUTO_POSITION = 0; START SLAVE`**)
- After running a few transactions, stop replication and restore the GTID state

(this last step is the voodoo part: you have to figure it out on your own)
(hint: it uses the GTID state noted before **`RESET MASTER`**)
(and as this wipes the binlogs, mind the impact on failover)

https://jfg-mysql.blogspot.com/2020/12/fixing-low-durability-gtid-replica-with-voodoo.html.html

Adding Complexity with GTIDs [4 of 4]

(Demystifying MySQL Replication Crash Safety – MinervaDB Athena 2020)

MySQL 8.0.17 stores the GTID position in InnoDB (WL#9211)

- Facebook is known to have this in their MySQL 5.6 fork
- But it is unclear if this solves the replication crash safety problem (Bug#70659 and Bug#92106 are still opened)
- And GTIDs in InnoDB not good for other Storage Engines (MyRocks, …)
- IMHO, better to update the table (`mysql.gtid_executed`) after each trx

While working on Bug#101874, I could easily break replication for 5.7.32 and 8.0.16 but not for 8.0.17 and 8.0.22. Bug#70659 and #92106 fixed ? Unsure what to think !

WL#9211: InnoDB: Clone Replication Coordinates
Bug#70659: Make crash-safe slave with gtid + less durable settings
Bug#92109: Make GTID replication crash-safe with less durable setting (bis)
Bug#101874: The manual is missing `sync_binlog = 1` for GTID replication crash safety

Master Replication Crash Safety [1 of 4]

(Demystifying MySQL Replication Crash Safety – MinervaDB Athena 2020)

Relaxing durability (`sync_binlog != 1`) loses the GTID state on OS crash

- What about the consequence on the master, with and without GTID ?

- With `sync_binlog != 1`, an OS crash will lose binary logs
- With `sync_binlog != 1`, usually `trx_commit != 1` (normally 2, but can be 0)
  - `trx_commit = 2` preserves data on mysqld crashes, 0 does not (2 is better)

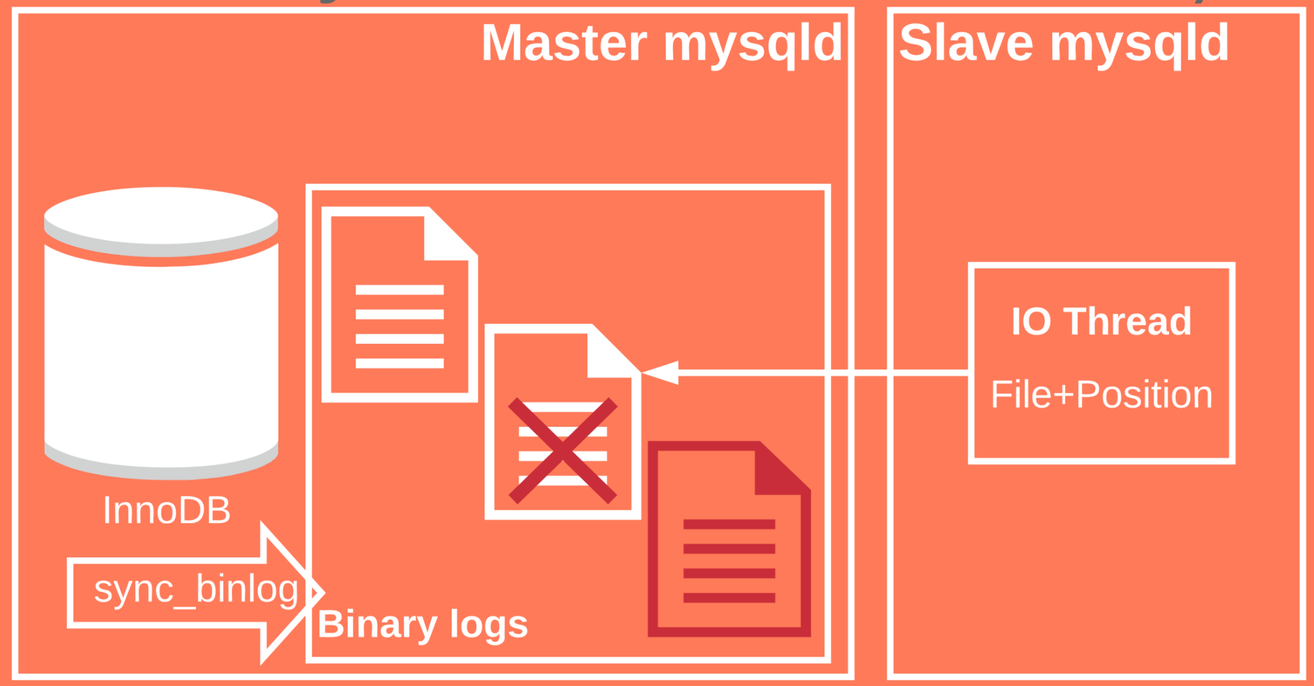
- After an OS crash with low durability, InnoDB will be out-of-sync with the binlogs
- And we cannot trust the binary logs on such master
  - Usually, InnoDB has more trx than binlogs (log flushed every seconds)
  - But the binary logs could have more trx than InnoDB in some cases

The failure mode will be different depending on the situation

https://jfg-mysql.blogspot.com/2018/10/consequences-sync-binlog-neq-1-part-1.html

Master Replication Crash Safety [2 of 4]

(Demystifying MySQL Replication Crash Safety – MinervaDB Athena 2020)

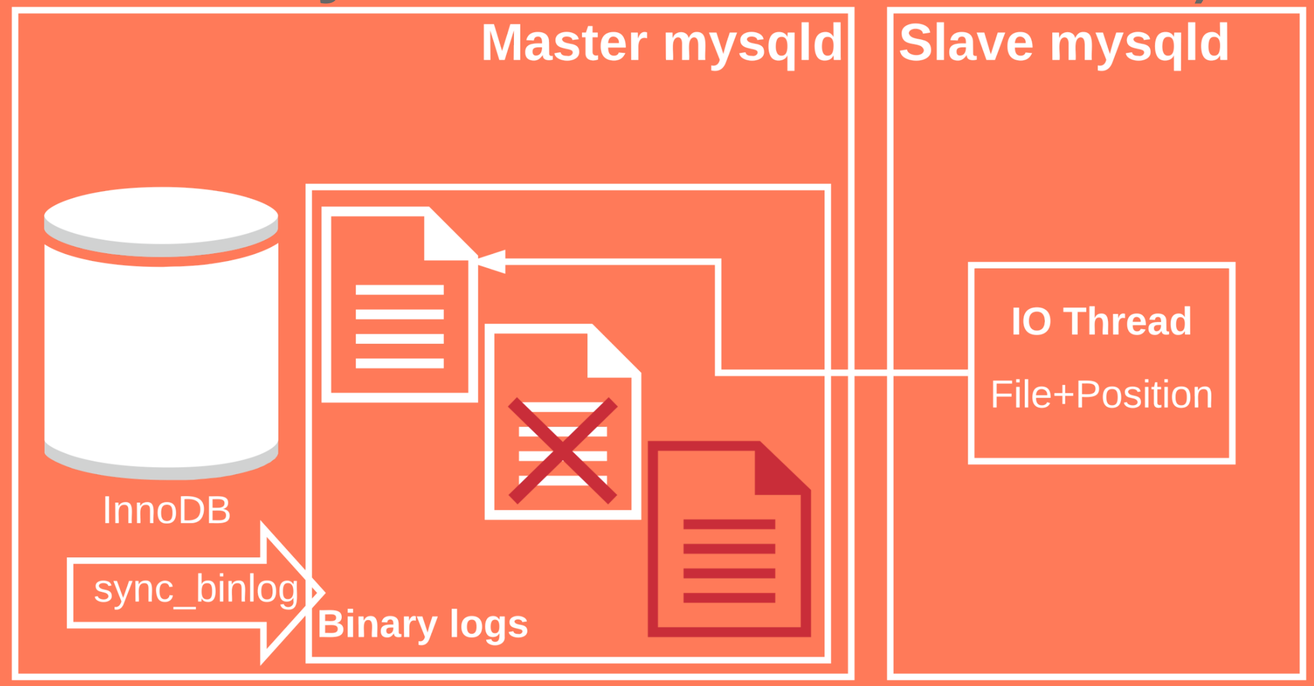


With file+position replication

- IO Thread in vanished binlogs
- So slaves executed phantom trx (ghost in binlogs, maybe not in InnoDB)
- When the master is restarted:
  - It records trx in a new binlog file
  - Most slaves are broken because they point in missing binlogs, and they might be out-of-sync with each-others

Master Replication Crash Safety [2 of 4]

(Demystifying MySQL Replication Crash Safety – MinervaDB Athena 2020)



With file+position replication

- IO Thread in vanished binlogs
- So slaves executed phantom trx (ghost in binlogs, maybe not in InnoDB)
- When the master is restarted:
  - It records trx in a new binlog file
  - Most slaves are broken because they point in missing binlogs, and they might be out-of-sync with each-others
  - Some lagging slave might skip vanished binlogs

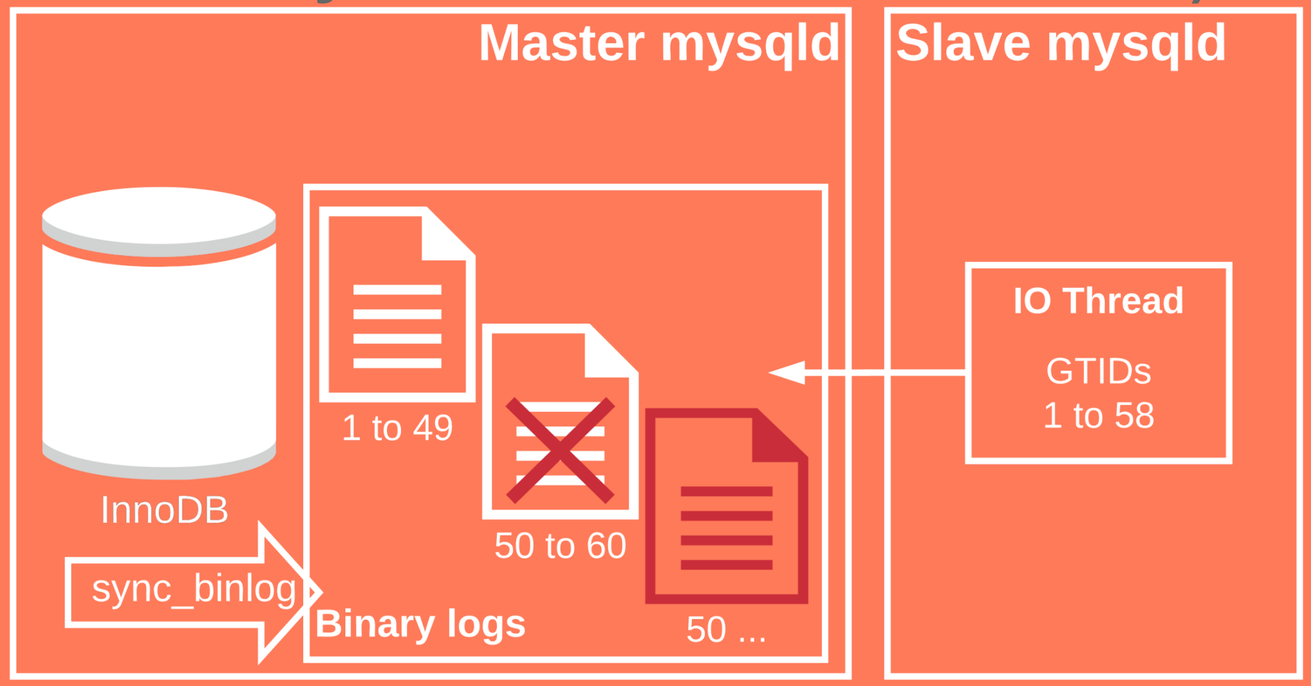
- We have broken slaves with more data than the master (data drift)
- And different data drift on “lucky” lagging slaves that might not break
- So data consistency is completely compromised !

Master Replication Crash Safety [3 of 4]

(Demystifying MySQL Replication Crash Safety – MinervaDB Athena 2020)

With GTID replication

- Slave also executed ghost transactions from binlogs
- But these are in their GTID state
- A recovered master reuses the GTIDs of the ghost transactions



- Slaves will *magically* reconnect to the master (`MASTER_AUTO_POSITION = 1`)
  1. If the master has not reused all ghost GTIDs, the slave breaks
  2. If it has, the slave skips the new transactions → more data drift
     (in illustration, the slave will skip the new 50 to 58 because it has the old one)
- So data consistency is also completely compromised !

Master Replication Crash Safety [4 of 4]

(Demystifying MySQL Replication Crash Safety – MinervaDB Athena 2020)

Summary of running with `sync_binlog != 1` (and `trx_commit != 1`)

- The binary logs of the master or slaves cannot be trusted after an OS crash
- On a master, having mysqld restart normally after such a crash leads to data drift
- After an OS crash, make sure no slaves reconnect to a recovered master (`OFFLINE_MODE = ON` in config file, and fail over to a slave)
- On slaves, having mysqld restarts after such a crash leads to truncated binlogs
- After an OS crash, purge all binary logs on the recovered slave (they are corrupted) (and if using GTIDs, the voodoo trick allows you to avoid restoring a backup)

Intermediate Masters are both master and slaves: do both of above

Bug#70659: Make crash-safe slave with gtid + less durable settings
Bug#92109: Make GTID replication crash-safe with less durable setting (bis)
Bug#98448: Please make running MySQL with `sync_binlog != 1` safer
Bug#101874: The manual is missing `sync_binlog = 1` for GTID replication crash safety

Adding complexity with MTS [1 of 4]

(Demystifying MySQL Replication Crash Safety – MinervaDB Athena 2020)

Multi-Threaded Slave (*MTS*) in 5.6 is doing out-of-order transaction commit

Example: transactions **A**, **B**, **C**, **D**, **E** on the master

- **`SHOW SLAVE STATUS`** pointing to B means trx up to A is committed
- C and E might be committed, B might be running and D might be pending scheduling (maybe B and D are in the same schema, with C and E in other schemas)

Same in 5.7 by default, with the **`LOGICAL_CLOCK`** parallel replication type having a **`slave_preserve_commit_order`** (*SPCO*) parameter

- SPCO **`OFF`** by default in 5.7 and 8.0 😐, **`ON`** requiring **`log-slave-updates`** 😕
- Since MySQL 8.0.19, SPCO can be **`ON`** without **`log-slave-updates`** 🙂

With out-of-order commit, a file+position is not enough for the slave state

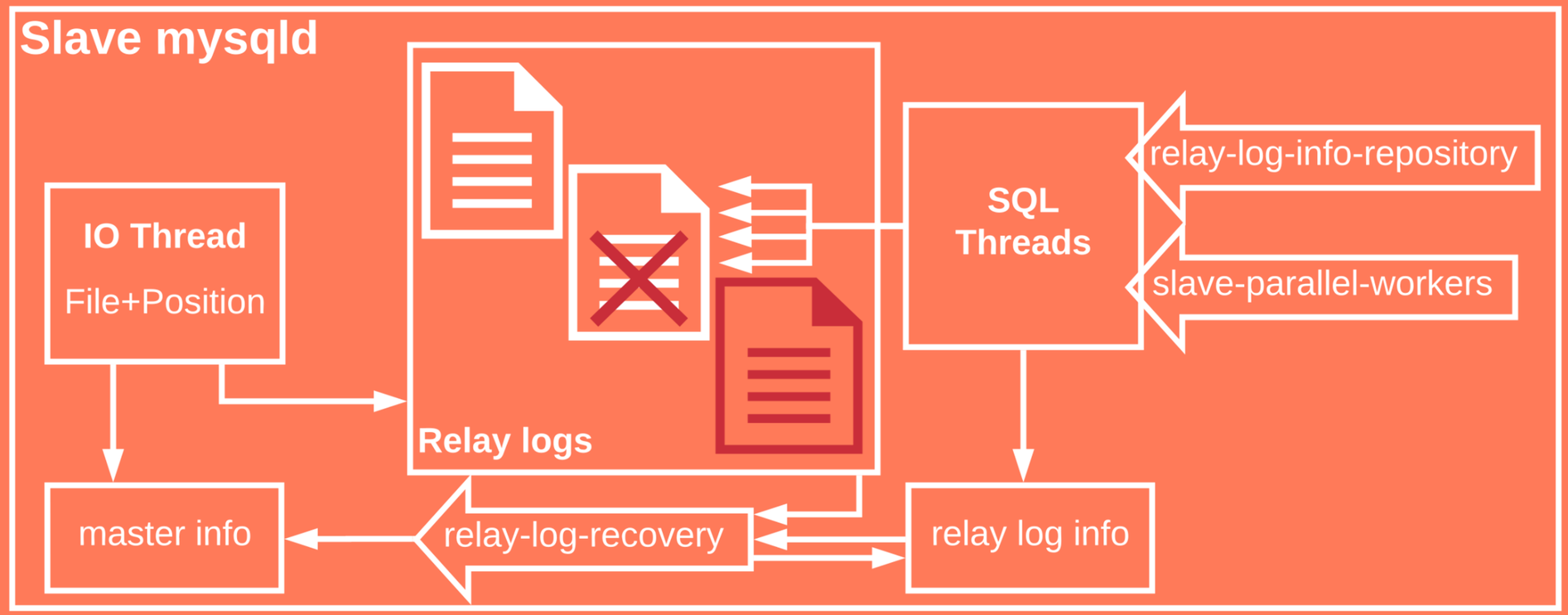
- GTID allows tracking complex a position (generating temporary holes on slaves)
- For file+position, there is the **`mysql.slave_worker_info`** table
  (More details in WL#5599: MTS: Recovery Process for MTS mode)

Adding complexity with MTS [2 of 4]

(Demystifying MySQL Replication Crash Safety – MinervaDB Athena 2020)

Without GTID, resuming replication after a crash needs filling the trx gap

- Did not work before 5.6.31 and 5.7.13, even for a simple mysqld crash (Bug#77496)
- Automated on recovery since then (`START SLAVE UNTIL SQL_AFTER_MTS_GAPS`)
- But this needs relay logs, which might have vanished after an OS crash

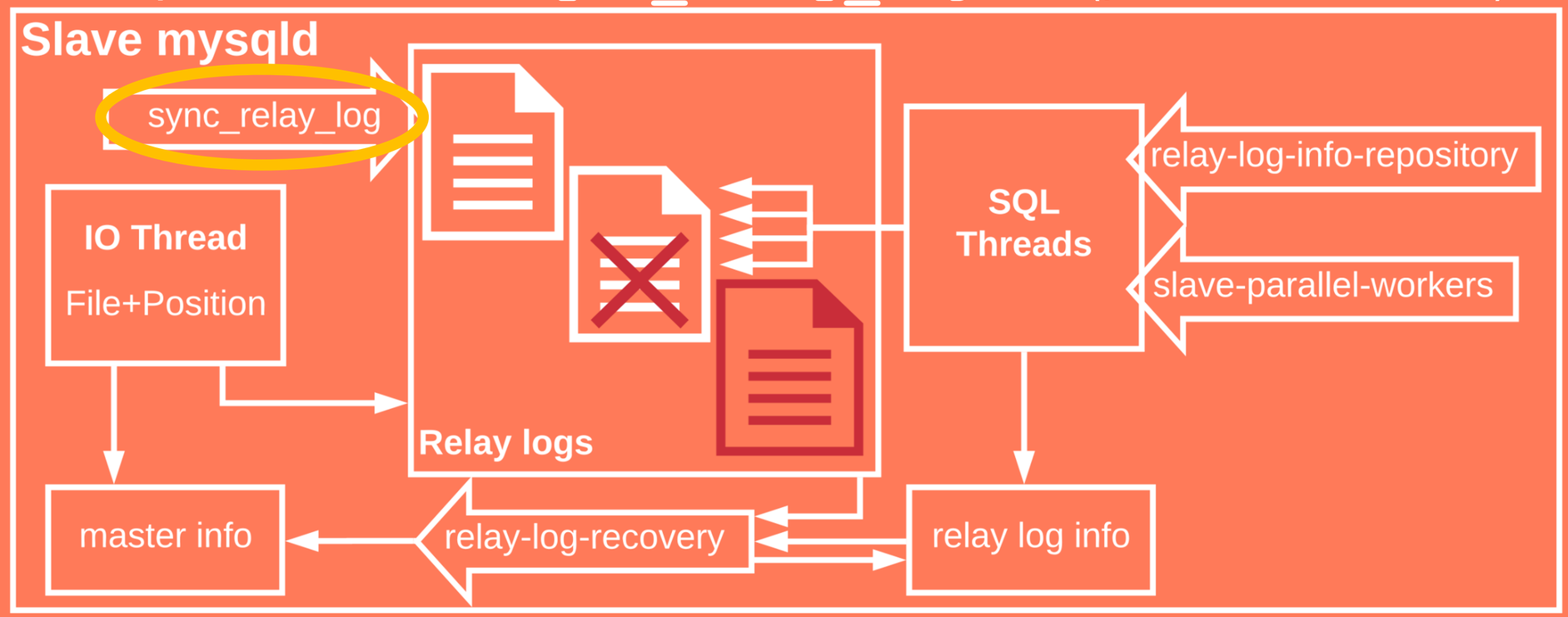


Adding complexity with MTS [3 of 4]

(Demystifying MySQL Replication Crash Safety – MinervaDB Athena 2020)

MTS without GTID not safe for OS crash and out-of-order commit (Bug#81840)

- Hard to accept workaround: **`sync_relay_log = 1`** (performance killer)



- Good state in **`mysql.slave_worker_info`**, so recovery possible with manual effort
- The good solution would be a better relay log recovery (Bug#93081)

Adding complexity with MTS [4 of 4]

(Demystifying MySQL Replication Crash Safety – MinervaDB Athena 2020)

With GTID, MySQL 5.6, 5.7 & 8.0 are replication crash-safe with MTS

- But this needs **`MASTER_AUTO_POSITION = 1`**
- And this needs **`sync_binlog = 1`** and **`trx_commit = 1`** (because GTID)
  (if 5.7+, low durability works without binlogs or without **`lsu`** because GTID table)

Example: **A**, **B**, **C**, **D**, **E** on the master with GTID **10**, **11**, **12**, **13**, **14**

- GTID executed on the slave is 1-10:12:14 before a crash
- Replication resumes by fetching 11:13:15… (after relay log recovery)

But with **`sync_binlog = 1`**, replication is slow (opposite goal of MTS)
We loop again to a trustworthy GTID position: Bug#92109 (and WL#9211 ?)
And voodoo only works with **`slave_preserve_commit_order`**
(Better relay log recovery would allow voodoo with out-of-order commit: Bug#93081)

Adding complexity with MTS [4 of 4]

(Demystifying MySQL Replication Crash Safety – MinervaDB Athena 2020)

With GTID, MySQL 5.6, 5.7 & 8.0 ~~are~~ **should be** repl. crash-safe with MTS

- But this needs `MASTER_AUTO_POSITION = 1`
- And this needs `sync_binlog = 1` and `trx_commit = 1`
  (if 5.7+, low durability works without binlogs or without `lsu` because GTID table)

Bug#92882: MTS not replication crash-safe with GTID and all the right parameters
(Only applies to OS crashes, fixed in MySQL 5.7.28 and 8.0.18)

Example: **A**, **B**, **C**, **D**, **E** on the master with GTID **10**, **11**, **12**, **13**, **14**

- GTID executed on the slave is 1-10:12:14 before the crash
- Relay log recovery tries to “fill the gaps” and this fails (relay logs are gone)
  (This might be a regression from the fix of Bug#77496)
- And reconnecting to the master with GTID would solve the problem
  (Easy workaround: `stop slave; reset slave; start slave`)

Related subjects – Low Durability

(Demystifying MySQL Replication Crash Safety – MinervaDB Athena 2020)

MySQL high availability needed **`sync_binlog = 1`** and **`trx_commit = 1`**

- This was the legacy MySQL high availability with shared disk
- Modern MySQL high availability is failing-over to a slave
  and this does not need **`sync_binlog = 1`** and **`trx_commit = 1`**

Also, flush / sync became more expensive recently (because Cloud)

- Magnetic disks were 10 ms
- Raid caches improved to 0.04 ms, and high-end NVMe to 0.15 ms
- But storage in the cloud raised this back to 0.6 ms (network round-trip)

More people run MySQL in the Cloud → **`sync_binlog`** and **`trx_commit != 1`**

MySQL is not 100% Cloud-Ready until these bugs are fixed (WL#9211 ?)

Bug#92109: Make GTID replication crash-safe with less durable setting (bis)
Bug#98448: Please make running MySQL with **`sync_binlog != 1`** safer

Related subjects – MariaDB

(Demystifying MySQL Replication Crash Safety – MinervaDB Athena 2020)

MariaDB still stores its *master info* and *relay log info* in files 😕

- But its GTID state is in a table (**`mysql.gtid_slave_pos`**) 🙂
- So even with **`sync_binlog != 1`**, it is repl. crash-safe with GTID
- But MariaDB relay log recovery with GTID is worse than MySQL
  https://jfg-mysql.blogspot.com/2015/10/bad-commands-with-mariadb-gtids-2.html

MariaDB has an interesting feature

- With more than one storage engine, one table is not optimal
- Having one table per storage engine would be better

Improving replication with multiple storage engines (MariaDB 10.3)
https://kristiannielsen.livejournal.com/19223.html

Related subjects – Pseudo GTIDs

(Demystifying MySQL Replication Crash Safety – MinervaDB Athena 2020)

Pseudo-GTIDs

- A way to get GTID-like features without GTIDs
- They work with any version of MySQL/MariaDB (even 5.5)
- But they assume in-order-commit: does not work with MTS (unless using **`slave_preserve_commit_order = ON`**)

They can provide slave replication crash safety

- With **`log-slave-updates`** and **`sync_binlog = 1`**
- Even on MySQL 5.5 or MariaDB 5.5

https://github.com/github/orchestrator/blob/master/docs/pseudo-gtid.md

Related subjects – Complexity

(Demystifying MySQL Replication Crash Safety – MinervaDB Athena 2020)

Tables for replication in the **`mysql`** schema:

- **`slave_master_info`** and **`slave_relay_log_info`** (updated every trx)
- **`slave_worker_info`** (for MTS)
- **`gtid_slave_pos`** (complex update logic, at binlog rotation with lsu)

Replication position being stored…

- in files and/or in tables, and/or in the binlary logs and/or in the InnoDB logs

And even more (ever growing GTID set, remembering the content of this talk, …)

I would like Oracle to build complexity out of all this

I think Binlog Servers are part of a simpler solution

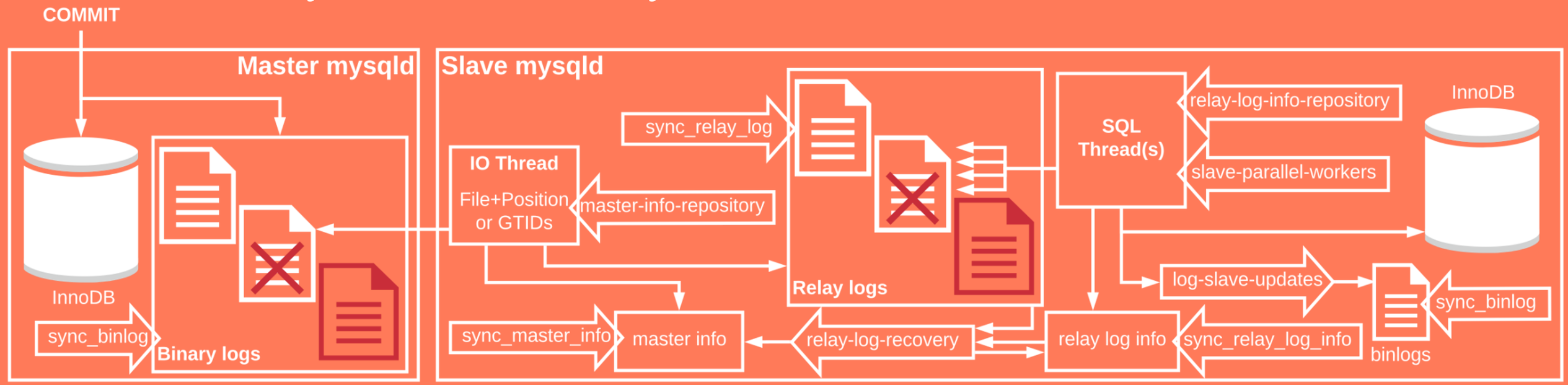
https://jfg-mysql.blogspot.com/2015/10/binlog-servers-for-backups-and-point-in-time-recovery.html
https://www.percona.com/blog/2019/03/15/mysql-ripple-first-impression-of-mysql-binlog-server/

Conclusion

(Demystifying MySQL Replication Crash Safety – MinervaDB Athena 2020)

Replication Crash Safety is complicated / complex !

- It has many edge cases
- Things might still change as bugs are fixed (or introduced)
- And hopefully improvements will be made
- So sorry, there is no easy short version



Conclusion [2 of 4]

(Demystifying MySQL Replication Crash Safety – MinervaDB Athena 2020)

For achieving master replication crash-safety

- `sync_binlog = 1`
- `innodb_flush_log_at_trx_commit = 1`

More complicated for slave replication crash-safety…

Conclusion [3 of 4]

(Demystifying MySQL Replication Crash Safety – MinervaDB Athena 2020)

Some parameters never impact slave replication crash safety

- `master-info-repository`
- `sync_master_info`
- `sync_relay_log_info`

Some are always needed for slave replication crash safety

- `relay-log-info-repository = TABLE`
- `relay-log-recovery = 1`

And some special situations apply…

Conclusion [4 of 4]

(Demystifying MySQL Replication Crash Safety – MinervaDB Athena 2020)

Three special situations for slave replication crash safety

If GTID with **`log-slave-updates`**, then **`sync_binlog = 1`** and **`trx_commit = 1`**
If GTID with out-of-order commit, then above plus **`MASTER_AUTO_POSITION = 1`**
If not GTID with out-of-order commit, then **`sync_relay_log = 1`** (will be slow)

Out-of-order commit only happens with MTS: **`slave_parallel_workers`** (***SPW***) **> 1**
with **`slave_parallel_type`** (***SPT***) = **`DATABASE`** (or in MySQL 5.6)
or with **`slave_preserve_commit_order`** (***SPCO***) = **`OFF`** (**`LOGICAL_CLOCK`** in 5.7)

A crash of GTID low durability (**`sync_binlog`** != **1** and **`trx_commit`** != 1) slave that uses **`log-slave-updates`** can be fixed with voodoo if in-order commit

In-order commit always happens with single-threaded replication
or with MTS (**`SPW`** > 1) and **`SPT`** = **`LOGICAL_CLOCK`** and **`SPCO`** = **`ON`**

Links [1 of 4]

(Demystifying MySQL Replication Crash Safety – MinervaDB Athena 2020)

Crash-Safe MySQL Replication - A Visual Guide

https://hackmongo.com/post/crash-safe-mysql-replication-a-visual-guide/

(diagrams in this talk are inspired by this post)

Links [2 of 4]

(Demystifying MySQL Replication Crash Safety – MinervaDB Athena 2020)

Content from the speaker related to this talk:

- On the consequences of `sync_binlog != 1` (part #1)
  https://jfg-mysql.blogspot.com/2018/10/consequences-sync-binlog-neq-1-part-1.html
- The consequences of `sync_binlog != 1` (FOSDEM 2020 talk)
  https://fosdem.org/2020/schedule/event/sync_binlog/
- Fixing low durability GTID slaves / replicas with Voodoo
  https://jfg-mysql.blogspot.com/2020/12/fixing-low-durability-gtid-replica-with-voodoo.html.html
- Replication crash safety with MTS in MySQL 5.6 and 5.7: reality or illusion?
  https://jfg-mysql.blogspot.com/2016/01/replication-crash-safety-with-mts.html
- A discussion about `sync-master-info` and other replication parameters
  https://jfg-mysql.blogspot.com/2016/08/discussion-about-sync-master-info-and-replication-parameters.html
- Do not run those commands with MariaDB GTIDs - part # 2
  https://jfg-mysql.blogspot.com/2015/10/bad-commands-with-mariadb-gtids-2.html
- The Full MySQL and MariaDB Parallel Replication Tutorial
  https://www.slideshare.net/JeanFranoisGagn/the-full-mysql-and-mariadb-parallel-replication-tutorial
- Binlog Servers for Simplifying Point in Time Recovery
  https://jfg-mysql.blogspot.com/2015/10/binlog-servers-for-backups-and-point-in-time-recovery.html

Links [3 of 4]

(Demystifying MySQL Replication Crash Safety – MinervaDB Athena 2020)

Related bugs (some fixed, some still open) for the full history on this

- Bug#70659: Make crash-safe slave work with gtid + less durable settings
- Bug#70669: Slave can't continue replication after master's crash recovery
- Bug#74321: Execute relay-log-recovery only when needed
- Bug#74323: Avoid overloading the master NIC on **`relay-log-recovery`** of a lagging slave
- Bug#77496: Replication position lost after crash on MTS configured slave (*really fixed ?*)
- Bug#81840: Automatic Replication Recovery Does Not Handle Lost Relay Log Events
- Bug#82667: Please clarify resiliency introduced by **`master_info_repository`** to **`TABLE`**
- Bug#90935: Modify **`relay_log_recovery`** Documentation
- Bug#92064: Information for the recovery of the IO thread is NOT in **`mysql.slave_master_info`**
- Bug#92093: Replication crash safety needs **`relay_log_recovery`** even with GTID
- Bug#92109: Please make replication crash-safe with GITD and less durable setting (bis)
- Bug#92882: MTS not replication crash-safe with GTID and all the right parameters
- Bug#93081: Please implement a better relay log recovery
- Bug#98448: Please make running MySQL with **`sync_binlog != 1`** safer
- Bug#101874: The manual is missing **`sync_binlog = 1`** for GTID replication crash safety
- Bug#101875: Make MySQL 8.0 replication crash-safe by default
- Bug#101876: The manual is overly conservative in parameters for replication crash safety

Links [4 of 4]

(Demystifying MySQL Replication Crash Safety – MinervaDB Athena 2020)

And more links:

- WL#9211: InnoDB: Clone Replication Coordinates
  https://dev.mysql.com/worklog/task/?id=9211
- How Binary Logs (and Filesystems) Affect MySQL Performance
  https://www.percona.com/blog/2018/05/04/how-binary-logs-and-filesystems-affect-mysql-performance/
- Improving replication with multiple storage engines (in MariaDB 10.3)
  https://kristiannielsen.livejournal.com/19223.html
- Pseudo-GTID
  https://github.com/github/orchestrator/blob/master/docs/pseudo-gtid.md
  https://speakerdeck.com/shlominoach/pseudo-gtid-and-easy-mysql-replication-topology-management
- MySQL Ripple: The First Impression of a MySQL Binlog Server
  https://www.percona.com/blog/2019/03/15/mysql-ripple-first-impression-of-mysql-binlog-server/

Thanks !

by Jean-François Gagné
System and MySQL Expert at HubSpot
Presented at MinervaDB Athena 2020
jfg DOT mysql AT gmail.com
Twitter: @jfg956