How to Avoid Pitfalls in Schema Upgrade

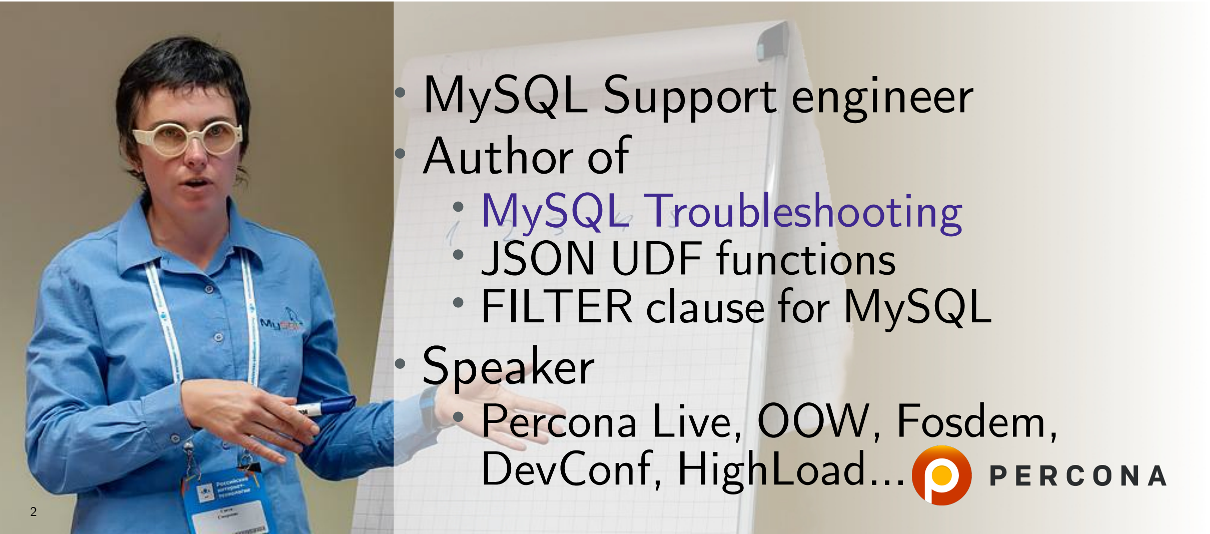
with Galera

February 2, 2020

Sveta Smirnova

PERCONA

Sveta Smirnova



- MySQL Support engineer
- Author of
  - MySQL Troubleshooting
  - JSON UDF functions
  - FILTER clause for MySQL
- Speaker
  - Percona Live, OOW, Fosdem, DevConf, HighLoad...

Table of Contents

- TOI
- RSU
- pt-online-schema-change (pt-osc)

Introduction

- Galera Replication Library
  - Provides synchronous replication for MySQL

Introduction

- Galera Replication Library
  - Provides synchronous replication for MySQL
- Galera Clusters

MariaDB Galera Cluster

Percona XtraDB Cluster

Galera Cluster for MySQL

PERCONA

How Galera works

- Data modification happens on a node
- Optimistic locking control

How Galera works

- Data modification happens on a node
- Optimistic locking control
- At the `COMMIT` time
  - Broadcasts write set for the cluster
  - Waits confirmation of the successful update
    - From all other nodes

Yes Commits transaction locally

No Rollbacks transaction

Data Updates

- Committed on all nodes or nowhere
- Safe

Challenges of DDL

- Replicated independently from storage engine

Challenges of DDL

- Replicated independently from storage engine
- Changes may affect query results
  - Adding/removal of `UNIQUE` keys
  - Adding/removal columns
  - Changing column definition

Challenges of DDL

- Replicated independently from storage engine
- Changes may affect query results
- Modification can happen on any node
  - The schema must be upgraded before DML
  - There is no way to rollback schema upgrade
  - MDLs are set only on one node
    - Not across the cluster
    - Not possible to rely on them for all nodes
    - Additional control required

TOI

Total Order Isolation (TOI)

- DDL changes are replicated in the same order regarding other transactions
- All nodes are in the absolutely same state at any point of time

TOI: Illustration

- 3-nodes cluster
  - Node A
  - Node B
  - Node C

TOI: Illustration

- Initial state

Node A	Node B	Node C
`INSERT(103)`	`UPDATE(101)`	`SELECT(100)`
`UPDATE(104)`	`INSERT(102)`	`INSERT(112)`
`ALTER(105)`	`DELETE(108)`	`SELECT(113)`
	`UPDATE(109)`	`UPDATE(114)`

PERCONA

TOI: Illustration

- Queries status

Node A

- ▶ INSERT(103)
- ▶ UPDATE(104)
- 🕒 ALTER(105)

Node B

- ▶ UPDATE(101)
- ▶ INSERT(102)
- 🕒 DELETE(108)
- 🕒 UPDATE(109)

Node C

- ▶ SELECT(100)
- 🕒 INSERT(112)
- ↻ SELECT(113)
- 🕒 UPDATE(114)

TOI: Illustration

- ALTER in progress

Node A	Node B	Node C
▶ `ALTER(105)`	🕒 `DELETE(108)`	🕒 `INSERT(112)`
	🕒 `UPDATE(109)`	↻ `SELECT(113)`
		🕒 `UPDATE(114)`

TOI: Illustration

- ALTER finished

Node A

Node B

- `DELETE(108)`
- `UPDATE(109)`

Node C

- `INSERT(112)`
- `SELECT(113)`
- `UPDATE(114)`

PROCESSLIST: DML before ALTER

```
DML node> select DB, COMMAND, TIME, STATE, INFO from information_schema.processlist WHERE DB=’sbtest’;
+--------+---------+------+---------------------------------------------------+-----------------------+
| DB     | COMMAND | TIME | STATE                                             | INFO                  |
+--------+---------+------+---------------------------------------------------+-----------------------+
| sbtest | Query   | 1    | wsrep: initiating pre-commit for write set (2886) | COMMIT                |
| sbtest | Query   | 1    | wsrep: initiating pre-commit for write set (2888) | COMMIT                |
| sbtest | Query   | 1    | wsrep: initiating pre-commit for write set (2884) | COMMIT                |
| sbtest | Query   | 1    | updating                                          | DELETE FROM sbtest1.. |
| sbtest | Query   | 1    | wsrep: initiating pre-commit for write set (2887) | COMMIT                |
| sbtest | Query   | 0    | wsrep: initiating pre-commit for write set (2889) | COMMIT                |
| sbtest | Query   | 1    | wsrep: initiating pre-commit for write set (2885) | COMMIT                |
| sbtest | Query   | 1    | wsrep: pre-commit/certification passed (2883)     | COMMIT                |
+--------+---------+------+---------------------------------------------------+-----------------------+
8 rows in set (0.00 sec)
```

PROCESSLIST: SELECT before ALTER

```
SELECT node> select DB, COMMAND, TIME, STATE, INFO from information_schema.processlist
          -> WHERE DB=’sbtest’;
+--------+---------+------+-------------------+--------------------------------------+
| DB     | COMMAND | TIME | STATE             | INFO                                 |
+--------+---------+------+-------------------+--------------------------------------+
| sbtest | Query   | 0    | statistics        | SELECT pad FROM sbtest2 WHERE id=5009 |
| sbtest | Query   | 0    | starting          | SELECT pad FROM sbtest3 WHERE id=4951 |
| sbtest | Query   | 0    | statistics        | SELECT pad FROM sbtest4 WHERE id=4954 |
| sbtest | Query   | 0    | System lock       | SELECT pad FROM sbtest2 WHERE id=5351 |
| sbtest | Query   | 0    | cleaning up       | SELECT pad FROM sbtest2 WHERE id=4954 |
| sbtest | Sleep   | 0    |                   | NULL                                 |
| sbtest | Query   | 0    | Sending to client | SELECT pad FROM sbtest1 WHERE id=4272 |
| sbtest | Query   | 0    | closing tables    | SELECT pad FROM sbtest4 WHERE id=4722 |
+--------+---------+------+-------------------+--------------------------------------+
8 rows in set (0.00 sec)
```

ALTER

```
DDL node> use ddltest;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

DDL node> alter table sbtest1 add key k1(c, k, pad);
Query OK, 0 rows affected (3 min 53.73 sec)
Records: 0 Duplicates: 0 Warnings: 0
```

PROCESSLIST: DML during ALTER

```
DML node> select DB, COMMAND, TIME, STATE, INFO from information_schema.processlist
       -> WHERE DB in ('sbtest','ddltest');
+---------+---------+------+---------------------------------------------------+----------------------+
| DB      | COMMAND | TIME | STATE                                             |                      |
+---------+---------+------+---------------------------------------------------+----------------------+
| sbtest  | Query   | 36   | wsrep: initiating pre-commit for write set (7886) | COMMIT               |
| sbtest  | Query   | 37   | wsrep: initiating pre-commit for write set (7882) | COMMIT               |
| sbtest  | Query   | 27   | wsrep: initiating pre-commit for write set (7887) | COMMIT               |
| sbtest  | Query   | 27   | wsrep: initiating pre-commit for write set (7888) | COMMIT               |
| sbtest  | Query   | 36   | wsrep: initiating pre-commit for write set (7885) | COMMIT               |
| sbtest  | Query   | 37   | wsrep: initiating pre-commit for write set (7883) | COMMIT               |
| sbtest  | Query   | 37   | wsrep: initiating pre-commit for write set (7884) | COMMIT               |
| sbtest  | Query   | 10   | wsrep: initiating pre-commit for write set (7889) | COMMIT               |
| ddltest | Sleep   | 38   | altering table                                    | alter table sbtest1. |
+---------+---------+------+---------------------------------------------------+----------------------+
9 rows in set (0.00 sec)
```

PROCESSLIST: SELECT during ALTER

```
SELECT node> select DB, COMMAND, TIME, STATE, INFO from information_schema.processlist
          -> WHERE DB in (’sbtest’,’ddltest’);
+---------+---------+------+-------------------+-------------------------------------------+
| DB      | COMMAND | TIME | STATE             |                                           |
+---------+---------+------+-------------------+-------------------------------------------+
| sbtest  | Sleep   | 0    |                   | NULL                                      |
| sbtest  | Sleep   | 0    |                   | NULL                                      |
| sbtest  | Query   | 0    | Sending to client | SELECT pad FROM sbtest4 WHERE id=4989     |
| sbtest  | Sleep   | 0    |                   | NULL                                      |
| sbtest  | Query   | 0    | query end         | SELECT pad FROM sbtest2 WHERE id=4961     |
| sbtest  | Sleep   | 0    |                   | NULL                                      |
| sbtest  | Sleep   | 0    |                   | NULL                                      |
| sbtest  | Sleep   | 0    |                   | NULL                                      |
| ddltest | Sleep   | 39   | altering table    | alter table sbtest1 add key k1(c, k, pad) |
+---------+---------+------+-------------------+-------------------------------------------+
9 rows in set (0.14 sec)
```

TOI Advantages

- Data always consistent
- DDL applied to all nodes at the same time
- No failure due to schema inconsistency

TOI Disadvantages

- The whole cluster blocked
  - For the duration of the entire DDL operation
- Schema upgrades replicated as a statement
  - There is no guarantee that the ALTER succeed!

How to Perform Upgrade with TOI

- Schedule maintenance window
- Run DDL
- Cluster won't be accessible until DDL finishes
  - `SELECT`s can continue
  - `wsrep_sync_wait != 1`

When to Use TOI

- Quick DDL operations

When to Use TOI

- Quick DDL operations
- Creating new database objects
  - `CREATE DATABASE`
  - `CREATE TABLE`

When to Use TOI

- Quick DDL operations
- Creating new database objects
- Online operations which modify metadata only
  - `RENAME INDEX`
  - `RENAME TABLE`
  - `DROP INDEX`
  - `ALGORITHM=INSTANT`

Full list

RSU

Rolling Schema Upgrade (RSU)

- Variable `wsrep_OSU_method`
- Puts node into de-sync state
  - For the duration of DDL
- Pauses Galera provider
- Schema can get out of sync!

User Responsibility

- Run DDL on the each node of the cluster
- Block read-write access that depend on DDL
  - Until all nodes are in sync
- Make sure no write is performed to the table
  - Until upgrade finishes on all nodes
- Failure makes cluster unrecoverable!

RSU Workflow

User Action	Node Operation
`SET SESSION wsrep_OSU_method = ’RSU’;`	Nothing
DDL	Is `wsrep_OSU_method` set to RSU? Yes Performs DDL
Any other statement	Nothing

How Node Internally Executes DDL in RSU Mode?

- Does node have transactions in `COMMIT` mode?

How Node Internally Executes DDL in RSU Mode?

▼ Does node have transactions in `COMMIT` mode?

Yes Wait for 5 milliseconds

How Node Internally Executes DDL in RSU Mode?

▼ Does node have transactions in `COMMIT` mode?

Yes Wait for 5 milliseconds

▼ Still transactions in the `COMMIT` mode exist?

How Node Internally Executes DDL in RSU Mode?

▼ Does node have transactions in `COMMIT` mode?

Yes Wait for 5 milliseconds

▼ Still transactions in the `COMMIT` mode exist?

Yes Abort DDL

How Node Internally Executes DDL in RSU Mode?

▼ Does node have transactions in `COMMIT` mode?

No Put node into de-sync state

How Node Internally Executes DDL in RSU Mode?

▼ Does node have transactions in `COMMIT` mode?

No Put node into de-sync state

▼ Pause write-set application

How Node Internally Executes DDL in RSU Mode?

▼ Does node have transactions in `COMMIT` mode?

No Put node into de-sync state

▼ Pause write-set application

▼ Execute DDL

How Node Internally Executes DDL in RSU Mode?

▼ Does node have transactions in `COMMIT` mode?

No Put node into de-sync state

▼ Pause write-set application

▼ Execute DDL

▼ Bring the node back to the cluster

How Node Internally Executes DDL in RSU Mode?

▼ Does node have transactions in `COMMIT` mode?

No Put node into de-sync state

▼ Pause write-set application

▼ Execute DDL

▼ Bring the node back to the cluster

● Synchronize

RSU: Locking

- Not avoidable
- Updates to all objects on the node in RSU mode must finish before the operation
- Failure aborts DDL

RSU Advantages

- Cluster remains functional
- Schedule long-running `ALTER`
  - In the best time possible

RSU Disadvantages

- No checks for data and schema consistency
  - This is your responsibility!

RSU Disadvantages

- No checks for data and schema consistency
- All writes must be stopped on the affected node
    - Otherwise DDL fails with an error

RSU Disadvantages

- No checks for data and schema consistency
- All writes must be stopped on the affected node
- gcache should be big enough to hold changes
  - Made while DDL was running
  - Failure will cause SST when node re-joins cluster
  - All schema changes will be lost

RSU Disadvantages

- No checks for data and schema consistency
- All writes must be stopped on the affected node
- gcache should be big enough to hold changes
- Any error can make cluster dysfunctional

RSU Disadvantages

- No checks for data and schema consistency
- All writes must be stopped on the affected node
- gcache should be big enough to hold changes
- Any error can make cluster dysfunctional
- Affected table must be offline
  - Until the schema upgrade is done on all nodes
  - Unless this is schema-compatible change

How to Use RSU

- Make sure gcache is big enough
    - Must hold all updates while DDL is in progress

How to Use RSU

- Make sure gcache is big enough
  - Must hold all updates while DDL is in progress
- Block all writes to the table/schema

How to Use RSU

- Choose an ”upgrading node”

How to Use RSU

- Choose an ”upgrading node”
- Block all write requests to this node

How to Use RSU

- Choose an ”upgrading node”
- Block all write requests to this node
- `SET SESSION wsrep_OSU_method = 'RSU';`

How to Use RSU

- Choose an ”upgrading node”
- Block all write requests to this node
- `SET SESSION wsrep_OSU_method = ’RSU’;`
- Perform DDL in the same session

How to Use RSU

- Choose an "upgrading node"
- Block all write requests to this node
- `SET SESSION wsrep_OSU_method = 'RSU';`
- Perform DDL in the same session
- `SET SESSION wsrep_OSU_method = 'TOI';`

How to Use RSU

- Choose an ”upgrading node”
- Block all write requests to this node
- `SET SESSION wsrep_OSU_method = ’RSU’;`
- Perform DDL in the same session
- `SET SESSION wsrep_OSU_method = ’TOI’;`
- Re-enable writes

How to Use RSU

- Choose an "upgrading node"
- Block all write requests to this node
- `SET SESSION wsrep_OSU_method = 'RSU';`
- Perform DDL in the same session
- `SET SESSION wsrep_OSU_method = 'TOI';`
- Re-enable writes
- Repeat for other nodes

pt-online-schema-change (pt-osc)

pt-online-schema-change (pt-osc)

- A tool, performing non-blocking upgrades
  - With TOI

pt-online-schema-change (pt-osc)

- A tool, performing non-blocking upgrades
- Creates a copy of table with altered definition

pt-online-schema-change (pt-osc)

- A tool, performing non-blocking upgrades
- Creates a copy of table with altered definition
- Creates triggers which will copy modified rows

pt-online-schema-change (pt-osc)

- A tool, performing non-blocking upgrades
- Creates a copy of table with altered definition
- Creates triggers which will copy modified rows
- Starts copying data in chunks
  - Absolutely under control
  - Can be paused or stopped

–max-flow-ctl

pt-online-schema-change (pt-osc)

- A tool, performing non-blocking upgrades
- Creates a copy of table with altered definition
- Creates triggers which will copy modified rows
- Starts copying data in chunks
  - All rows already in the table are copied in chunks
  - Newly modified rows are copied using triggers

pt-online-schema-change (pt-osc)

- A tool, performing non-blocking upgrades
- Creates a copy of table with altered definition
- Creates triggers which will copy modified rows
- Starts copying data in chunks
- Once copy is complete, drops the table

pt-online-schema-change (pt-osc)

- A tool, performing non-blocking upgrades
- Creates a copy of table with altered definition
- Creates triggers which will copy modified rows
- Starts copying data in chunks
- Once copy is complete, drops the table
- Renames the copy into the original table name

pt-osc Advantages

- DDL is safe and non-blocking

pt-osc Disadvantages

- Works only with InnoDB tables
- Increases IO load even for inplace operations
- Conflicts with already existing triggers
  - Unless you use MariaDB >= 10.2.3
- Foreign keys updates are not effectively safe

How to Use pt-osc

- Study pt-osc options
  - `--max-flow-ctl`
- Set appropriate limits
- Make sure `wsrep_OSU_method` is TOI
- Run pt-osc

Which Method to Use?

- Will DDL be fast?
  - `CREATE DATABASE`
  - `CREATE TABLE`
  - `DROP INDEX`
  - Any `ALTER` on small tables
  - Other

Which Method to Use?

▼ Will DDL be fast?

Yes Use TOI

Which Method to Use?

▼ Will DDL be fast?

Yes Use TOI

No Evaluate if you can use pt-osc

- Operation on the InnoDB table
- Table has no triggers or MariaDB $>=$ 10.2.3
- Table is not referenced by a foreign key
- You can tolerate increased IO

Which Method to Use?

▼ Will DDL be fast?

Yes Use TOI

No Evaluate if you can use pt-osc

Yes Use pt-osc

Which Method to Use?

▼ Will DDL be fast?

Yes Use TOI

No Evaluate if you can use pt-osc

Yes Use pt-osc

No Use RSU

- Stop all write traffic on the node
- Stop all write traffic to the modified table
- Make sure to upgrade on all nodes

Conclusion

- Use TOI whenever possible
- Then use pt-osc
- RSU is a last resort

More information

- Galera Cluster
- MariaDB Galera Cluster
- pt-online-schema-change

Thank you!

www.slideshare.net/SvetaSmirnova

twitter.com/svetsmirnova

github.com/svetasmirnova

PERCONA

DATABASE PERFORMANCE MATTERS