How Facebook Got Consistency with MySQL in the Cloud

Sam Dunster
Production Engineer
Consistency
Replication
Replication for High Availability
Asynchronous Replication
Master

Storage engine

Replica
Master

- Storage engine
- Binary logs

Replica
Master

- Storage engine
- Binary logs

Replica

- Storage engine
- Binary logs
UPDATE table_a SET foo = "bar";

Master

- Storage engine
- Binary logs

Replica

- Storage engine
- Binary logs
UPDATE table_a SET foo = "bar";
UPDATE table_a SET foo = "bar";
UPDATE table_a SET foo = "bar";

Master
- Storage engine
- Binary logs

Replica
- Relay logs
- Storage engine
- Binary logs
UPDATE table_a SET foo = "bar";
UPDATE table_a SET foo = "bar";
UPDATE table_a SET foo = "bar";

Master
- Storage engine
- Binary logs

Replica
- Relay logs
- Storage engine
- Binary logs

read/write
read (delayed)
Binary logs
### Position-based

```sql
mysql > SHOW MASTER STATUS;
+------------------+----------+
| File             | Position |
+------------------+----------+
| mysql-bin.000003 | 73       |
```
GTID
Global transaction ID

3E11FA47-71CA-11E1-9E33-C80AA9429562:23

source_id  transaction_id
GTID Set

Show me which transactions you have executed

2174B383-5441-11E8-B90A-C80AA9429562:1-3,
24DA167-0C0C-11E8-8442-00059A3C7B00:1-19
GTID-based Auto positioning

mysql> CHANGE MASTER TO
   > MASTER_HOST = host,
   > MASTER_PORT = port,
   > MASTER_USER = user,
   > MASTER_PASSWORD = password,
   > MASTER_AUTO_POSITION = 1;
GTID

Global transaction ID
UPDATE table_a SET foo = "bar";

Master
- Storage engine
- Binary logs

Replica
- Relay logs
- Storage engine
- Binary logs

read/write
read (delayed)
UPDATE table_a SET foo = "bar";
Promotion
Live Master Promotion
Live Master Promotion
Dead Master Promotion
Live Master Promotion

Master
Commit

Replica
Live Master Promotion

Master Commit

Replica
Live Master Promotion

Master
Commit

Replica
MASTER_POS_WAIT()
Live Master Promotion
Dead Master Promotion

Master Commit

Replica
Dead Master Promotion

Diagram showing the process of promoting a dead master to a replica.
Dead Master Promotion

Master
Commit

Replica

SQL THREAD WAIT
Synchronous Replication
Synchronous Replication

- Master
- Replica
Synchronous Replication

Master
Prepare

Replica
Synchronous Replication

- Master
  - Prepare
- Replica
  - Commit
Synchronous Replication

Master
Commit

Replica
Commit

Commit

Commit
Live Master Promotion

Master

Replica
Live Master Promotion
Dead Master Promotion
Dead Master Promotion
Synchronous Constraints
Semi-Synchronous Replication
Semi-Synchronous Replication

Master

Replica
Semi-Synchronous Replication
Semi-Synchronous Replication

Master Binlogs ←→ Replica RelayLog
Semi-Synchronous Replication

Master Binlogs → Replica RelayLog

Replica RelayLog ← Master Binlogs
Semi-Synchronous Constraints

A few micro seconds

Master

A few micro seconds

Semi Sync Replica

Semi Sync Replica
Semi-Synchronous Replication
Semi-Synchronous mysqlbinlog
Semi-Synchronous Constraints

A few micro seconds

Logtailer

A few micro seconds

Logtailer

Master

Milliseconds OK!

Milliseconds OK!

Replica
Dead Master Failover

Diagram:
- Master
- Replica
- Logtailer

Process:
1. When the master is dead, the replica takes over.
2. Logtailer then monitors the replica for any updates.
Dead Master Failover

[Diagram showing a failover process with 'Master' and 'Replica', indicating file size and read master log pos]
Dead Master Failover

Master

Replica

--start-position=10000

mysqbinlog

10000

11000
Lossless semi-sync
Lossless Semi-Sync

=> Commit;
Binlog Prepare => No-op
InnoDB Prepare => Written to InnoDB for recovery
Binlog Commit => Written to binlog
InnoDB Commit => Visible from other clients
<= OK;
Lossless Semi-Sync

=> Commit;
Binlog Prepare => No-op
InnoDB Prepare => Written to InnoDB for recovery
Binlog Commit => Written to binlog
InnoDB Commit => Visible from other clients
<= OK;

=> Commit;
Binlog Prepare
InnoDB Prepare
Binlog Commit
InnoDB Commit
Wait for Semi-Sync Ack
<= OK;
Lossless Semi-Sync

=> Commit;
Binlog Prepare => No-op
InnoDB Prepare => Written to InnoDB for recovery
Binlog Commit  => Written to binlog
InnoDB Commit  => Visible from other clients
<= OK;

=> Commit;
Binlog Prepare
InnoDB Prepare
Binlog Commit
InnoDB Commit
Wait for Semi-Sync Ack
<= OK;

Crash!
Lossless Semi-Sync

=> Commit;
Binlog Prepare => No-op
InnoDB Prepare => Written to InnoDB for recovery
Binlog Commit  => Written to binlog
InnoDB Commit  => Visible from other clients
<= OK;

=> Commit;
Binlog Prepare
InnoDB Prepare
Binlog Commit
InnoDB Commit
Wait for Semi-Sync Ack
<= OK;

 <=> Crash!

=> Commit;
InnoDB Prepare
Binlog Commit
Wait for Semi-Sync Ack
<= OK;

(MYSQL 5.7)
Rollout

https://www.youtube.com/watch?v=sl-PACHY2zE
mysqlbinlog + semi-sync patches
Binlog Server
Local Disk

Master

Semi-sync Replication

Semi-sync Binlog Writer/Acker

mysqlbinlog

log.index

Binlog

00001.bin

00002.bin

00003.bin
Local Disk

Replica

Master

Semi-sync Replication

Semi-sync Binlog Writer/Acker

Binlog Server Log-tailer

Binlog Reader

Promotion Catchup

log.index

00001.bin

00002.bin

00003.bin

Binlog

Binlog

Binlog
mysqbinlog --start-position=10000

binary-logs-3301.127526

Master

Logtailer

Replica
CHANGE MASTER TO

Master

Logtailer

Replica
Binlog Server ++
Lagged replicas?
Error 1236
Replicaset 12345

Master

Replication

Greatly Lagged Replica

slave status:
Error Msg:
Master has purged the required binary logs
Replicaset 12345

- Binlog Reader/Sender
- Binlog Locator
- Binlog Server

Greatly Lagged Replica

change master to Binlog Server;
binlog_server> show slave status
*************************** 1. row ***************************
Slave_IO_State: Waiting for master to send event
Master_Host: HOSTNAME
Master_Port: PORT
Connect_Retry: 0
Master_Log_File: binary-logs-xxxxxx.007964
Read_Master_Log_Pos: 97115
   BinLog_File: binary-logs-xxxxxx.007964
   Binlog_Pos: 97115
Last_IO_Errno: 0
Master_Server_Id: 3695980966
Executed_Gtid_Set: ea4a5e01-b3e4-4273-a25e-88d06db8d1a5:1-902842,
b29a87bd-d60b-4455-9ab8-90d7b720f169:1-81669
Mysql_Replicaset: REPLICA_SET_NAME
Replicaset_Tier_Version: VERSION_NUM
Semisync_Slave: Yes
There’s plenty more to Binlog Server
Search for “Binlog Server at Facebook”
MariaDB MaxScale

https://mariadb.com/resources/blog/the-binlog-server/

https://github.com/mariadb-corporation/MaxScale
Distributed systems are really hard
DBAs don’t scale as well as MySQL does
Lossless Semi-Sync

=> Commit;
Binlog Prepare => No-op
InnoDB Prepare => Written to InnoDB for recovery
Binlog Commit => Written to binlog
InnoDB Commit => Visible from other clients
<= OK;

=> Commit;
Binlog Prepare
InnoDB Prepare
Binlog Commit
InnoDB Commit  => Crash!
Wait for Semi-Sync Ack
<= OK;

=> Commit;
InnoDB Prepare
Binlog Prepare
Binlog Commit
InnoDB Commit  => OK;
Wait for Semi-Sync Ack
InnoDB Commit
<= OK;
Lossless Semi-Sync

=> Commit;  
Binlog Prepare => No-op  
InnoDB Prepare => Written to InnoDB for recovery  
Binlog Commit  => Written to binlog  
InnoDB Commit  => Visible from other clients  
<= OK;

=> Commit;  
Binlog Prepare  
InnoDB Prepare  
Binlog Commit  
InnoDB Commit  
Wait for Semi-Sync Ack  
<= OK;

=> Commit;  
InnoDB Prepare  
Binlog Prepare  
Binlog Commit  
InnoDB Commit  
Wait for Semi-Sync Ack  
<= OK;

Crash!
<table>
<thead>
<tr>
<th>status</th>
<th>semi-sync thread</th>
<th>async thread</th>
</tr>
</thead>
<tbody>
<tr>
<td>transaction 1</td>
<td>acked</td>
<td></td>
</tr>
<tr>
<td>status</td>
<td>semi-sync thread</td>
<td>async thread</td>
</tr>
<tr>
<td>---------</td>
<td>------------------</td>
<td>--------------</td>
</tr>
<tr>
<td>transaction 1</td>
<td>acked</td>
<td></td>
</tr>
<tr>
<td>transaction 2</td>
<td>prepare</td>
<td></td>
</tr>
<tr>
<td>status</td>
<td>semi-sync thread</td>
<td>async thread</td>
</tr>
<tr>
<td>------------</td>
<td>------------------</td>
<td>--------------</td>
</tr>
<tr>
<td>transaction 1</td>
<td>acked</td>
<td></td>
</tr>
<tr>
<td>transaction 2</td>
<td>waiting for ack</td>
<td></td>
</tr>
<tr>
<td>status</td>
<td>semi-sync thread</td>
<td>async thread</td>
</tr>
<tr>
<td>------------</td>
<td>------------------</td>
<td>--------------</td>
</tr>
<tr>
<td>transaction 1</td>
<td>acked</td>
<td></td>
</tr>
<tr>
<td>transaction 2</td>
<td>acked</td>
<td></td>
</tr>
<tr>
<td>Status</td>
<td>Semi-sync Thread</td>
<td>Async Thread</td>
</tr>
<tr>
<td>-----------</td>
<td>------------------</td>
<td>--------------</td>
</tr>
<tr>
<td>transaction 1</td>
<td>acked</td>
<td></td>
</tr>
<tr>
<td>transaction 2</td>
<td>acked</td>
<td></td>
</tr>
<tr>
<td>transaction 3</td>
<td>prepare</td>
<td></td>
</tr>
<tr>
<td>transaction</td>
<td>status</td>
<td>semi-sync thread</td>
</tr>
<tr>
<td>---------------</td>
<td>------------------</td>
<td>------------------</td>
</tr>
<tr>
<td>transaction 1</td>
<td>acked</td>
<td></td>
</tr>
<tr>
<td>transaction 2</td>
<td>acked</td>
<td></td>
</tr>
<tr>
<td>transaction 3</td>
<td>waiting for ack</td>
<td></td>
</tr>
<tr>
<td>transaction 1</td>
<td>status</td>
<td>semi-sync thread</td>
</tr>
<tr>
<td>--------------</td>
<td>--------</td>
<td>------------------</td>
</tr>
<tr>
<td></td>
<td>ached</td>
<td></td>
</tr>
<tr>
<td>transaction 2</td>
<td>ached</td>
<td></td>
</tr>
<tr>
<td>transaction 3</td>
<td>ached</td>
<td></td>
</tr>
</tbody>
</table>
Flappy/partially-isolated master
MySQL Automation

Client <-> Master <-> Replica

Logtailer

Logtailer
Logtailer failures
MySQL Automation

Client <-> Master <-> Replica

Logtailer

Logtailer
MySQL Automation

Client <-> Master (blocked) <-> Replica

lbu010
lbu030
These situations were very rare
Everything open source*
https://github.com/facebook/mysql-5.6/

* except Facebook's Binlog Server

MariaDB MaxScale
https://mariadb.com/resources/blog/the-binlog-server/
https://github.com/mariadb-corporation/MaxScale
Sam Dunster

facebook

Come chat at the Facebook booth right after this!