MySQL 8.0: The New Replication Features

Luís Soares
Software Development Director
MySQL Replication
Safe Harbor Statement

The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described for Oracle’s products remains at the sole discretion of Oracle.
Agenda
Program Agenda

1. Introduction
2. Use Cases
3. Enhancements in MySQL 8 (and 5.7)
4. Roadmap
5. Conclusion
Introduction
Today...

- Technology mesh.
- All things distributed.
- Large amounts of data to handle, transform, store.
- Offline periods are horribly expensive, simply unaffordable.
- Go green requires dynamic and adaptative behavior.
- Much more data to store – e.g. social media, “Look at all of my pictures!”; Monitoring – Keeping logs for N years! ; IoT – and much more.
- Moving, transforming and processing data quicker than anyone else means having an edge over competitors.
- It is a zoo. Distributed coordination and monitoring is key.
Database Replication

Replication
“The process of generating and reproducing multiple copies of data at one or more sites.”
MySQL Database Replication: Overview
MySQL Database Replication: Some Notes

Binary Log

• Logical replication log recording master changes (binary log).
• Row or statement based format (may be intermixed).
• Each transaction is split into groups of events.
• Control events: Rotate, Format Description, Gtid, and more.

Layout of the Binary Log.
MySQL Database Replication: Some Notes

Coordination Between Servers

**Since 3.23**
- asynchronous (native)

**Since 5.5**
- semi-synchronous (plugin)

**Since 5.7.17**
- And in MySQL 8 as of 8.0.1
- group replication (plugin)
Use cases
Clustering Made Practical

Replicate
Automate
Integrate
Scale
Enhance
Replicate

Group Replication

• For highly available infrastructures where:
  – the number of servers has to grow or shrink dynamically;
  – with as little pain as possible.
Automate
Group Replication

• Single-primary mode
  – Automatic PRIMARY/SECONDARY role assignment
  – Automatic new PRIMARY election on PRIMARY failures
  – Automatic setup of read/write modes on PRIMARY and SECONDARIES
  – Automatic global consistent view of which server is the PRIMARY
Integrate
Binary Log

- Logical replication log
  - Extract, transform and load.
  - MySQL fits nicely with other technologies.
Scale

Asynchronous Replication

- Replicate between clusters
  - For disaster recovery
- Read Replicas
  - For read-scale out. Deploy asynchronous read replicas connected to the cluster
Enhance InnoDB Cluster

- **InnoDB Cluster Integrated Solution**
  - Group Replication for high availability.
  - Asynchronous Replication for Read Scale-out.
  - One-stop shell to deploy and manage the cluster.
  - Seamlessly and automatically route the workload to the proper database server in the cluster.
  - Hide failures from the application.
Enhancements in MySQL 8 (and 5.7)

3.1 Consistency
3.2 Operations
3.3 Monitoring
3.4 Performance
3.5 Security
3.6 Other
Consistency Levels

• Eventual Consistency (default)
  – Transaction does not wait at all.
  – Executes on the current snapshot of the data on that member.

• Before Consistency (Synchronize on Reads)
  – Transaction waits for all preceding transactions to complete.
  – Executes on the most up to date snapshot of the data in the group.

• After Consistency (Synchronize on Writes)
  – Transaction waits until all members have executed it.
  – Executes on the current snapshot of the data on that member.
Consistency Levels

• Before and After (Yes, you can combine both)
  – Transaction waits for all preceding transactions and for all members to execute it.
  – Executes on the most up to date snapshot of the data in the group and updates everywhere before returning to the application.

• Before On Primary Fail-over
  – Transaction waits for all transactions in the new primary’s replication backlog to be executed.
  – Executes on the snapshot of the data that the old primary was in when it stepped down (or crashed).
Accessing Data: Synchronize Execution **Eventually**

Transaction ✗ Does Not Wait for ✗ to Complete Before it Executes On Member A.

*SET @@session.group_replication_consistency=EVENTUAL*

```
X=5 WHERE X=1
```

```
READ X=?
```

```
X = 1
```

New in 8.0.14
Accessing Data: Synchronize **Before** Execution

Transaction **green** Waits for **red** to Complete Before it Executes On Member A.

**SET @@session.group_replication_consistency=BEFORE**

- **App**
  - X=5 WHERE X=1

- **App**
  - READ X=?

- **App**
  - X = 5

- **dB** A
- **dB** B
- **dB** C

**waits until member A receives and commits.**

*New in 8.0.14*
Accessing Data: Synchronize **After** Execution

- Waits For All Members to Execute.  ▶ Reads Updated Value Without Waiting.

**SET @@session.group_replication_consistency=AFTER**

Diagram:
- **App**
  - X=5 WHERE X=1
- **Database**
  - A → B → C
  - A
  - B
  - C

**Commits after all members have received and executed it.**

**Commit OK.**

**reads X = 5 immediately.**

New in 8.0.14
Accessing Data: Synchronize **Before on Primary Fail-over**

- Waits For All Members to Execute.
- Reads Updated Value Without Waiting.

**SET @@session.group_replication_consistency=BEFORE_ON_PRIMARY_FAILOVER**

---

**Diagram:**

1. **App**
   - **A**: X=5 WHERE X=1
   - **B**: READ X?
   - **C**: READ X?

2. **A (new primary)**
   - **B**: holds execution
   - **C**: until is executed.

3. **App**
   - **READ X? X=5**
Consistency Levels – User Interface

• System Variable Controls the Behavior: `group_replication_consistency`.

• *Global and Session scope*
  – *Can be set per transaction.*

• Values:
  – `EVENTUAL`
  – `BEFORE_ON_PRIMARY_FAILOVER`
  – `BEFORE`
  – `AFTER`
  – `BEFORE_AND_AFTER`
Consistency Levels

- Consistency has an impact on throughput.
- Usually not all transactions are executed under strong consistency requirements...

**Sysbench RW Sustained Throughput**

<table>
<thead>
<tr>
<th>Durable settings</th>
<th>EVENTUAL</th>
<th>BEFORE</th>
<th>AFTER</th>
<th>BEFORE&amp;AFTER</th>
</tr>
</thead>
<tbody>
<tr>
<td>number of client threads</td>
<td>1</td>
<td>8</td>
<td>16</td>
<td>32</td>
</tr>
<tr>
<td>Transactions per second</td>
<td>0</td>
<td>2,000</td>
<td>4,000</td>
<td>6,000</td>
</tr>
</tbody>
</table>

**Sysbench Update Index Sustained Throughput**

<table>
<thead>
<tr>
<th>Durable settings</th>
<th>EVENTUAL</th>
<th>BEFORE</th>
<th>AFTER</th>
<th>BEFORE&amp;AFTER</th>
</tr>
</thead>
<tbody>
<tr>
<td>number of client threads</td>
<td>1</td>
<td>8</td>
<td>16</td>
<td>32</td>
</tr>
<tr>
<td>Transactions per second</td>
<td>0</td>
<td>5,000</td>
<td>10,000</td>
<td>15,000</td>
</tr>
</tbody>
</table>
Enhancements in MySQL 8 (and 5.7)

3.1 Consistency
3.2 Operations
3.3 Monitoring
3.4 Performance
3.5 Security
3.6 Other
Preventing (Involuntary) Writes on Stale Members
Automatically Setting Server to Read-Only

A joins a single primary cluster and it allows update operations prior to joining.

A becomes a secondary and is automatically set to read only.

A leaves the group and remains read only until the DBA reverts.

B joins a single primary cluster and it allows update operations prior to joining.

B becomes a secondary and is automatically set to read only.

B leaves the group and remains read only until the DBA reverts.

Backported to 5.7.20
Preventing (Involuntary) Reads/Writes on Stale Members
Automatically Shoot Member in the Head (ST*NITH)

A is a secondary and was automatically set to read only.

A leaves the group involuntarily (failed to apply changes, network partitioned, etc).

A shuts itself down, thus becoming effectively unavailable for both reads and write operations.

@@group_replication_exit_state_action={ READ_ONLY | ABORT_SERVER }

New in 8.0.12
Backported to 5.7.24
Control Primary Promotion: Priorities

Choose next primary by assigning election weights to the candidates.

- **B** is the primary.
- **B** is not in the group anymore.
- **C** has higher weight than **A**. **C** is elected the new primary.
Control Primary Election: Choose Your Primary.

User tells current primary to give up its role and assign it to another server.

B is the primary.

Now I want A to be the primary, not B.

A is the primary. B stepped down.
Single-Primary to Multi-Primary and Back Online

User can specify, online, on which mode the group operates.

*Single-primary* mode, *B* is the primary.

Now I want the group operating in *multi-primary* mode.

Return to *single-primary* mode.
Dealing With Unreliable Networks: Relaxed Member Eviction

User controls the amount of time to wait until others decide to evict a member from the group.

I think A is not around anymore. Lets kick it out of the group!

Hold on. Lets wait for some time! Maybe it is just too busy to talk to us right now...

OK. Lets give it some more time!

See... I told you. A is back! Good thing we waited.

UNREACHABLE
Automatic Cluster-Rejoin on Partitions

Member tries to rejoin automatically in case it gets evicted.

A is a secondary and was automatically set to read only.

A leaves the group involuntarily because of network issues.

A automatically tried (and succeeded) to rejoin the cluster after it had been evicted. No user intervention.
Enhancements in MySQL 8 (and 5.7)

- Consistency
- Operations
- Monitoring
- Performance
- Security
- Other
Monitor Lag With Microsecond Precision
Through the entire asynchronous topology

How much time does my data take to reach D coming from A?
Monitor Lag With Microsecond Precision
From the immediate master

How much time does my data originated in A takes to flow from B to C?
Monitor Lag with Microsecond Precision
For each stage of the replication applier process

- Per Stage Timestamps
  - User can monitor how much time it takes for a specific transaction to traverse the pipeline.
Global Group Stats Available on Every Server
Version, Role and more

• Query one Replica, Get status of all
  – Every replica reports group-wide information about roles and versions of the members of the group.
  – Also available at any replica are group-wide status.
Group Replication Message Cache Memory Usage

- GCS/XCom’s Paxos message cache is instrumented.
- GCS/XCom’s Paxos message cache memory usage is exposed in performance schema.

```sql
-- This is a session open on ServerA and the user is reading stats on GCS_XCom message cache

ServerA> select * from memory_summary_global_by_event_name where event_name like "GCS_XCom%"

*************************** 1. row ***************************
EVENT_NAME: memory/group_rpl/GCS_XCom::xcom_cache
COUNT_ALLOC: 28890317
COUNT_FREE: 28840318
SUM_NUMBER_OF_BYTES_ALLOC: 24499151783
SUM_NUMBER_OF_BYTES_FREE: 24470424555
LOW_COUNT_USED: 0
CURRENT_COUNT_USED: 49999
HIGH_COUNT_USED: 50000
LOW_NUMBER_OF_BYTES_USED: 0
CURRENT_NUMBER_OF_BYTES_USED: 28727228
HIGH_NUMBER_OF_BYTES_USED: 135676530

1 row in set (0.01 sec)
```
Enhancements in MySQL 8 (and 5.7)

3.1 Consistency
3.2 Operations
3.3 Monitoring
3.4 Performance
3.5 Security
3.6 Other
Highly Efficient Replication Applier

Write set parallelization

• Delivers the **best throughput** of the three dependency trackers, at **any** concurrency level.

• WRITESET dependency tracking allows **applying** a **single threaded** workload in **parallel**.

• Fast Group Replication recovery – time to catch up.
Highly Efficient Replication Applier
Write set parallelization

Applier Throughput: Sysbench Update Index

- COMMIT_ORDER
- WRITESET
- WRITESET_SESSION

Backported to 5.7.22
Fast Group Replication Recovery
Replica quickly online by using WRITESET

Group Replication Recovery Time: Sysbench RW (durable settings)

- Sysbench RW at 33% capacity (workload: 4K TPS on 64 threads)
- Sysbench RW at 66% capacity (workload: 8K TPS on 64 threads)

Group Replication Recovery Time: Sysbench Update Index (durable settings)

- Sysbench RW at 33% capacity (workload: 9K TPS on 64 threads)
- Sysbench RW at 66% capacity (workload: 18K TPS on 64 threads)

MySQL 5.7.20
MySQL 8.0.3
Even Faster Group Replication Recovery: Clone Support

Empty or delayed replica quickly online by using Automatic Cloning and WRITESET

D is empty or has very old data (a lot to catch up)

D automatically takes a snapshot of A (clones A and restores the image into itself). D’s old data is forever gone.

D has recovered and has caught up using a snapshot of A and a small amount of binary logs

Requires that the **clone plugin** is **installed**. The clone plugin is shipped with MySQL 8.0.17.
Even Faster Group Replication Recovery
Empty or delayed replica quickly online by using Automatic Cloning and WRITESET

• Recovery using binary logs only vs recovery using clone and binary logs together.
  – There are cases binary logs are quicker and cases clone together with binary logs take less time.

• No Network involved.
  – Network adds latency
  – Network may not impact throughput (if it is not a bottleneck).

![Database State Transfer Throughput](chart.png)

Database State Transfer Throughput
(TPCC Workload, 100GB Database, local copy)
High Cluster Throughput

More transactions per second while sustaining zero lag on any replica

Asynchronous Replication Sustained Throughput
(Sysbench Update Index, durable settings)

<table>
<thead>
<tr>
<th>Number of Clients on the Master</th>
<th>MySQL 5.7</th>
<th>MySQL 8.0.3</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td></td>
<td></td>
</tr>
<tr>
<td>32</td>
<td></td>
<td></td>
</tr>
<tr>
<td>64</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Asynchronous Replication Sustained Throughput
(Sysbench Update Index, non-durable settings)

<table>
<thead>
<tr>
<th>Number of Clients on the Master</th>
<th>MySQL 5.7</th>
<th>MySQL 8.0.3</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td></td>
<td></td>
</tr>
<tr>
<td>32</td>
<td></td>
<td></td>
</tr>
<tr>
<td>64</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Efficient Replication of JSON Documents

Replicate only changed fields of documents (Partial JSON Updates)

- Numbers are from a specially designed benchmark:
  - tables have 10 JSON fields,
  - each transaction modifies around 10% of the data

![Binary Log Space Per Transaction](chart)

- Entire JSON
- Partial JSON (FULL, MINIMAL)
Efficient Replication of JSON Documents

Replicate only fields of the document that changed (Partial JSON Updates)

Throughput on the Master:
Partial JSON vs Complete JSON

Throughput on the Slave:
Partial JSON vs Complete JSON
Enhancements in MySQL 8 (and 5.7)

3.1 Consistency
3.2 Operations
3.3 Monitoring
3.4 Performance
3.5 Security
3.6 Other
Binary Log Encryption on Disk

New in 8.0.14

App

INSERT ...

relay log

binary log

<empty>

Server A

primary

Comm. Framework

relay log

binary log

Server B

secondary
Binary Log Encryption on Disk

New in 8.0.14

Relay logs are also encrypted.

Binary logs are encrypted.

App

INSERT ...

relay log
<empty>

binary log

Server A

INSERT ...

Comm. Framework

Server B

INSERT ...

relay log

binary log

PRIMARY

SECONDARY
Binary Log Encryption on Disk

- Protects Binary Log Data at rest.
- Controllable Using a System Variable: `binlog_encryption`
- Two tier encryption, one master key and one key per file.

- Rotate the Binary Log Encryption Key:
  
  `ALTER INSTANCE ROTATE BINLOG KEY`

New in 8.0.14

New in 8.0.16
Performance

• Marginal impact on throughput
  – More visible when the commit rate is higher.
Enhancements in MySQL 8 (and 5.7)

3.1 Binary Log Enhancements
3.2 Operations
3.3 Monitoring
3.4 Performance
3.5 Security
3.6 Other
Changes to defaults in MySQL 8

High performance replication enabled out-of-the-box

- Binary log is **on** by default.
- Logging of slave updates is **on** by default.
- Replication metadata is stored in **InnoDB tables** by default instead of files.
- Row-based applier falls back into **hash scans** to find rows instead of table scans.
- Transaction write-set extraction is **on** by default.
- Binary log expiration is set to **30 days** by default.
- Server-id is set to **1** by default instead of **0**.
Other MySQL 8 Group Replication Enhancements

- **Monitoring**: Group Replication threads instrumented and shown in performance schema
- **Monitoring**: Group Replication conditional variables and mutexes instrumented and shown in performance schema
- **Operations**: SAVEPOINT support when write sets are being extracted
- **Operations**: Support hostnames in Group Replication whitelist
- **Operations**: New options to fine tune the cluster automatic flow control.
- **Operations**: Cross-version replication policies for GR
Other MySQL 8 Group Replication Enhancements

- **Performance:** More efficient code path between network layer and replication layer.
- **Performance:** Configurable communication pipeline size for group replication.
- **Performance:** Enhanced large transactions support for group replication.
- **Performance:** Support for protocol compression in mysqlbinlog.
- **Troubleshooting:** Dynamic and high performance debugging of group replication inter-node messaging.
- **Security:** Encryption of transient replication files.
- **Infrastructure:** IPv6 support.
Roadmap
The Road to MySQL 8 Group Replication and InnoDB Clusters

- MySQL 8.0.0 DMR
  More replication enhancements
- MySQL 8.0.1 DMR
  GR is released with 8
  InnoDB Cluster is GA
  Lots of replication enhancements
- MySQL 8.0.2 DMR
  bug fixes
  partial JSON updates
  monitoring enhancements
- MySQL 8.0.3 RC1
  bug fixes
- MySQL 8.0.4 RC2
  bug fixes
  monitoring
- MySQL 8.0.5
  Instrumentation
- MySQL 8.0.6
  Member auto-shutdown
- MySQL 8.0.11 GA
  bug fixes
- MySQL 8.0.12 GA
  MySQL 8.0.13 GA
  Select Primary Monitoring
  Performance Relax Eviction
- MySQL 8.0.14 GA
  Consistency Encryption
- MySQL 8.0.16 GA
  Large Trxs Auto-rejoin
- MySQL 8.0.17 GA
  Clone Support Encryption enhancements
- MySQL 5.7.9
  5.7 is GA lifecycle interfaces
  P_S tables for GR server side changes
- MySQL 5.7.17
  GR is GA
- MySQL 8.0.0.2 labs
  Hello world!
- MySQL 8.0.17
  MySQL 5.7.18
  MySQL 8.0.18
Conclusion
Conclusion

Latest MySQL 8 GA is out:

• Performance/efficiency improvements
  – Automatic Cloning of donors in Group Replication means one less provisioning step required from the user.

• More encryption features
  – Encrypt even transient replication data that touches the disk.

• Improved Operations and DBA experience
  – Mysqlbinlog supports protocol compression
  – Enhanced cross-version replication protection in Group Replication.
  – Enhanced distributed recovery by integrating the clone plugin. **Seamless and automatic snapshotting, provisioning and catch up.** Reduced operations overhead.
Where to go from here?

• Packages

• Documentation

• Blogs from the Engineers (news, technical information, and much more)
  – http://mysqlhighavailability.com