Percona Live 2019

Austin, Texas | May 28th – 30th, 2019
MySQL Group Replication vs PostgreSQL Bi-Directional Replication

Mariella Di Giacomo
Dennis Rich
Viasat
Goal

Inspire everyone to use both MySQL Group Replication and PostgreSQL Bi-Directional Replication to take advantage of their numerous capabilities.
Outline

• Definition and brief introduction of Database Replication
• Brief introduction to PostgreSQL Replication
• Brief introduction to MySQL Replication
• Main features of PostgreSQL Bi-Directional Replication 1 (Postgres-BDR)
• Main features of MySQL Group Replication
• Similarities and Differences between MySQL Group Replication and Postgres-BDR
• Conclusions
Database Replication
Database Replication

• The process of copying data from one location to another.
• The frequent electronic copying of data from one or more databases in *one server to another server* so that all users share the same level of information.
• Key feature for **High Available** (Fault-Tolerant) and **Scalable** environments.
Database Replication

• Database servers can work together to allow a second server to take over quickly if the primary server fails (high availability), or to allow several computers to serve the same data in read and/or write mode (load balancing and high availability).

• PostgreSQL and MySQL open source relational database management systems (RDBMS) support database replication.
Database Replication

The main **high availability (HA)** solutions for MySQL and PostgreSQL can be achieved using one of the following replication designs (topologies):

- **Master Slave** (single master to one slave, standby, or to multiple slaves)
- **Master Master** (active master and passive master)
- **Multi-Master** (two or more active masters)
Database Replication

- Database Replication is used with many database management systems (DBMS), usually with a **master slave** relationship between the original and the copies.

- For example: the master (original) node logs the changes, which then ripple through to one or more slave (copy) nodes. Each slave node records a message stating that it has received the change successfully.

- The successful execution of a modification will allow subsequent changes.
Database Replication

• **Failover (master master)** replication (**high availability**) is a special case of master slave database replication.

• For example: one (or more) **passive slave** server(s) is replicating the master data in order to prepare for takeover in the event the master stops functioning. The active master is the only server that will accept writes from clients.
Database Replication

• **Multi-Master** replication: a method of database replication which allows data to be stored by a group of database servers, and updated by any member of the group.

• Servers are responsive to client data requests. The entire system is responsible for **two main tasks**:  
  1. Propagating data modifications made by each member to the rest of the group.  
  2. Resolving conflicts between concurrent changes made by different members.
Database Replication

Two topologies of Multi-Master Replication
PostgreSQL Replication
PostgreSQL Replication

PostgreSQL supports two main types of replication:

1. **Physical (Block Level) Streaming Replication**
   - Sends Write-Ahead Logging (WAL) transaction log records to replicas.
   - Is suitable for Local High Availability (HA), High Bandwidth.
   - Is streamed *before committed* (saved permanently and visible to users) changes (commit).

2. **Logical (Row Level) Streaming Replication**
   - Sends data only (row level changes) to replicas.
   - Is suitable for Remote High Availability (HA), Lower Bandwidth.
   - Is streamed *at committed changes* (commit).
PostgreSQL Replication

Physical (Block Level) Streaming Replication
PostgreSQL Replication

Logical (Row Level) Streaming Replication
PostgreSQL Internal Replication

PostgreSQL supports core (internal) solutions for replications:

- Physical Asynchronous Write-Ahead Logging (WAL) transaction log records (Warm Standby)
- Physical Asynchronous Streaming Replication (Hot Standby)
PostgreSQL Internal Replication

- Streaming Synchronous Replication

- Cascading Replication
PostgreSQL Internal Replication

- Logical Decoding
- Logical Streaming Replication (PostgreSQL version v10+)
- Replication over SSL
PostgreSQL External Replication

PostgreSQL supports external solutions for replications:

• **Focus: Bi-Directional Replication (BDR)** (Latest BDR3 requires PostgreSQL v10+)
  • Slony
  • Bucardo
  • Londiste
  • PGLLogical
MySQL Replication
MySQL Replication

MySQL supports **two main types** of replication:

1. **Event Based Replication**
   - Sends events to each slave server.
   - Sends events in statement, row or mixed format.

2. **Global Transaction Identifiers (GTID) Based Replication**
   - Sends GTIDs to each slave server.
   - Is completely transaction-based.
   - Sends the information using statement-based or row-based replication.
MySQL Replication

MySQL Replication with Events from Binary Logs
MySQL Replication

MySQL Replication with GTIDs from Binary Logs

- **Global Transaction Identifiers**
  - Generate a global transaction identifier on commit:
    - server_uid: unique identifier for the server
    - number: incremented for each transaction
  - Write GTID to binary log

- **Binary log**
  - Various GTIDs for different transactions

- **Database Server**
  - Transaction 1
  - Transaction 2
MySQL Internal Replication

MySQL supports core (internal) replication solutions:

- Asynchronous
- Synchronous and Semi-Synchronous
- Statement, Row or Mixed Replication with Events from Binary Log
- Replication with Global Transaction Identifiers (GTIDs) from Binary Log with Row and Statement Replication
- Replication over SSL
- Multi-Source Replication
MySQL External Replication

MySQL supports external replication solutions:

- Tungsten (replicator across different RDBMS)
- Galera Cluster
PostgreSQL Bi-Directional Replication (Postgres-BDR)
Postgres-BDR from 2ndQuadrant:

- a ground-breaking **asynchronous multi-master** logical replication tool for PostgreSQL databases with **high availability** across regions.
- designed for **geographically distributed** PostgreSQL databases for small and large environments.
Postgres-BDR

- Multi-master
- Asynchronous
- Eventually consistent
  - Does not prevent concurrent writes
  - Optimistic conflict detection (after commit)
  - Automatic conflict resolution
Postgres-BDR

• Postgres-BDR (starting from Postgres-BDR 1) has been in full production status since 2014.
• Postgres-BDR 1 is open source, freely accessible and supports PostgreSQL server 9.4.
• The latest release of BDR (June 2018), Postgres-BDR 3, supports the latest versions of PostgreSQL servers (used only by paying customers).
Postgres-BDR

Characteristics

• Postgres-BDR can be used in a multi-master or single master scenario.
• Each database node receives changes from other members and users.
• Each node in the group has its own copy of the data (not a shared-storage architecture).
• Changes made to one node are not replicated to other nodes (through asynchronous replication) before they are committed locally.
• Data is not the same on all nodes at once; some nodes will have data that has not arrived at other nodes. However, eventually the nodes will sync (“eventually consistent” architecture).
Postgres-BDR

- Utilizes logical (row-based, individual row values) replication.

- Ensures that constraints (unique constraints, check constraints, foreign keys, etc.) are always consistent with each node.

- Nodes can satisfy read operations (queries) without communicating with other nodes.
Postgres-BDR

Main System Characteristics

• Group Communication System
• Transactions
• Recovery
• Multi-Primary
Postgres-BDR

Group Communication System

- Is based on a **custom protocol** on top of standard logical replication.
- Is a **fully meshed** group of nodes.
- Requires a connection string for any new node to at least one other existing node.
Transactions (system behavior)

• At the **commit** time, propagates the changes to each node (through logical replication).
• Uses transparent **global sequences** to identify transactions.
• Provides “Eventually consistent” data (**asynchronous** replication).
  (Choice between Latency or Consistency is available in BDR-3).
Postgres-BDR

Recovery

- If a network link between the nodes goes down, the messages related to data manipulation (DML) changes are stored and when the link comes up again, the nodes will catch up.

- If a node goes down, the database will be still available for reads and data changes if the number of healthy (good) nodes is greater than the half of the nodes.

- If a node goes down, the Data Definition Language (DDL) commands will be blocking the execution in the databases (this has changed in BDR-3).
Postgres-BDR

Multi-Primary (each master is concurrently writable by the users)
• High availability
• Faster fail-over
• Limitations: conflict resolution (last update wins, etc.)
MySQL Group Replication
MySQL Group Replication

• Is a **plugin** (library) to MySQL server enabling a fault-tolerant, **highly available** group of MySQL database servers (when a node dies, the group can continue its business).

• **Replicates** (using Xcom, a variant of the Paxos protocol) the **system state** to a set of database servers.

• Provides a **highly elastic** (it is possible to add or remove a node on need) group of database nodes.

• Allows **read, and at times, write scalability**.
MySQL Group Replication

- Many clients
- Many servers
- Read scalability
- Write everywhere*
- Elastic, Highly available
- Agreement on:
  - Members
  - Message delivery
  - State updates
MySQL Group Replication

- Guarantees that the database service is continuously available, even after some of the database servers fail, as long as the number of failed nodes is not all or a majority.
- Assures the database service is still available, even after several database node failures which might cause degraded performance or scalability.
- **Isolates** server failures (isolated and independent).
MySQL Group Replication

- Is open source, freely accessible and has been production available since MySQL 5.7.17 (December 2016).
- Is the core engine of MySQL InnoDB Cluster, the built-in and open source complete High Availability solution for MySQL 8.0.
MySQL Group Replication

Main System characteristics

- Group Communication System
- Transactions
- Recovery
- Multi-Primary
MySQL Group Replication

Group Communication System

• Provides **membership**.
  All members agree on who is a member.
  Each member has the list of all the members.
• Guarantees **total order broadcast**.
  All members receive all messages and the messages are received in the proper order (as sent) using a variant of the Paxos protocol.
MySQL Group Replication

Transactions (system behavior)

- Guarantees “no split-brain”.
- Provides “Eventually consistent data” (commit executed using asynchronous replication).
MySQL Group Replication

Master 1
- execute
- certify
- binlog
- commit

Master 2
- certify
- relay log
- apply
- binlog
- commit

Master 3
- certify
- relay log
- apply
- binlog
- commit

Consensus
MySQL Group Replication

Recovery (special logic when a node joins or leaves a group)

- A node can **join** a group.
- A joining node catches up with old transactions. (and cache transactions that group applies during catch-up)
- A joining node catches up with cached transactions.

Example: Adding a Node to an Existing Group

- Server that joins the group will automatically synchronize with the others
  - It will retrieve the diff between its data and the rest of the group
  - Hint: provision the new node with base data (i.e., restore a backup) before joining an existing group
MySQL Group Replication

- A node can be **easily removed** from a group.
MySQL Group Replication

Multi-Primary (each master is concurrently writable by the users)

- Higher throughput
- Faster fail-over
- Limitations: conflict resolution *(first update wins, etc.)*
MySQL Group Replication

**Single Primary**

- Write Clients
- Read Clients

**Multi Primary**

- Write Clients
- Read Clients

Server S1 is the primary.

All servers are primaries.

**Multi Primary**

Write Clients

Read Clients

Write Clients

Read Clients

S1’s client connects to S3.
Similarities and Differences between MySQL Group Replication and Postgres-BDR
## Comparisons

<table>
<thead>
<tr>
<th>Feature</th>
<th>MySQL Group Replication</th>
<th>Postgres-BDR</th>
</tr>
</thead>
<tbody>
<tr>
<td>License</td>
<td>Freely Accessible and Open Source (GPLv2).</td>
<td>Freely Accessible and Open Source only for Postgres-BDR 1.</td>
</tr>
<tr>
<td>Network Latency</td>
<td>Low Network Latency. High bandwidth network connections are requirements for optimal performance of the group.</td>
<td>Geographically distributed nodes. Network Latency is not relevant.</td>
</tr>
<tr>
<td>Minimum number of nodes in a group</td>
<td>At least 3 nodes, to have a fault tolerant group after a node dies.</td>
<td>At least 3 nodes, to have a fault tolerant group after a node dies.</td>
</tr>
<tr>
<td>Maximum number of node in a group</td>
<td>At the moment, the maximum feasible number is around 9.</td>
<td>Theoretically the maximum number of nodes is around 48, but at the moment the feasible maximum number is around 9-10.</td>
</tr>
</tbody>
</table>
## Comparisons

<table>
<thead>
<tr>
<th>Feature</th>
<th>MySQL Group Replication</th>
<th>Postgres-BDR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial database node</td>
<td>Prior to the creation of a group at least a database node with MySQL Server should exist. That node, will be the first node of the group, after the cluster is created.</td>
<td>The initial Postgres-BDR 1 database node must be running and enabled, prior to adding any additional node.</td>
</tr>
<tr>
<td>Automatic Node Addition</td>
<td>Any additional database node may be added easily, elastically and concurrently to the group.</td>
<td>Any additional database node may be added concurrently to the group.</td>
</tr>
<tr>
<td>Automatic Node Removal</td>
<td>When a node dies, the automatic failure detection mechanism communicates to all the live nodes the event. It is extremely easy to remove a node from a group. It is not necessary to remove or clean a dead node to continue successful operations.</td>
<td>Postgres-BDR 1 does not have a built-in automatic mechanism that cleans every part related to a removed node.</td>
</tr>
</tbody>
</table>
## Comparisons

<table>
<thead>
<tr>
<th>Feature</th>
<th>MySQL Group Replication</th>
<th>Postgres-BDR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replication</td>
<td>Virtually Synchronous Replication</td>
<td>Asynchronous Logical Replication</td>
</tr>
<tr>
<td>Data Manipulation Language (DML) Commands</td>
<td>Virtually synchronously. A quorum (only a majority) of nodes must complete DML statements with success. When a node becomes faulty, it will not block any DML statement propagation.</td>
<td>Asynchronously</td>
</tr>
<tr>
<td>Data Definition Language (DDL) Commands</td>
<td>Virtually Synchronously. The DDL Statements are transmitted virtually synchronously and a quorum (only a majority) of nodes must complete them with success. When a node becomes faulty, it will not block any DDL statement propagation.</td>
<td>Synchronously</td>
</tr>
</tbody>
</table>
## Comparisons

<table>
<thead>
<tr>
<th>Feature</th>
<th>MySQL Group Replication</th>
<th>Postgres-BDR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Global DDL statements (CREATE ROLE, CREATE USER, etc.)</td>
<td>They are replicated like any DDL or DML statements.</td>
<td>They are not replicated in Postgres-BDR 1. They should be created on each node, if the created objects will be referenced through BDR.</td>
</tr>
<tr>
<td>Table Indexes</td>
<td>Tables must have Primary Key (or equivalent).</td>
<td>Tables must have Primary Key (or equivalent).</td>
</tr>
<tr>
<td>Transaction Identifier</td>
<td>Supports only GTIDs and binary logs in row format.</td>
<td>Logical Streaming Replication with row level.</td>
</tr>
<tr>
<td>Multi-Master Conflicts</td>
<td>The DDL and related DML statements should not be spread across database servers if the DML statements will follow immediately their correspondent DDL statements.</td>
<td>The database schema should be created and updated to avoid Multi-Master replication conflicts.</td>
</tr>
</tbody>
</table>
Conclusions
Conclusions

• MySQL Group Replication and Postgres-BDR started as freely accessible and open source, however at the moment only MySQL Group Replication latest releases are freely accessible and open source.

• Both products have overall automated administration and low database node maintenance.

• Postgres-BDR is a redundant geographically distributed solution, while MySQL Group Replication is a redundant solution that requires low network latency.

• Both designs support practically the same number of minimum and maximum database nodes.

• Both solutions support single-master and multi-master scenarios.
References

- https://en.wikipedia.org/wiki/MySQL
- https://en.wikipedia.org/wiki/PostgreSQL
- https://www.2ndquadrant.com
- https://blog.2ndquadrant.com
Thank you!