Galera Cluster for MySQL

## **What is Galera Cluster?**

Galera Cluster is a synchronous multi-master replication plug-in for InnoDB. It is very different from the regular MySQL Replication, and addresses a number of issues including write conflicts when writing on multiple masters, replication lag and slaves being out of sync with the master. Users do not have to know which server they can write to (the master) and which servers they can read from (the slaves).

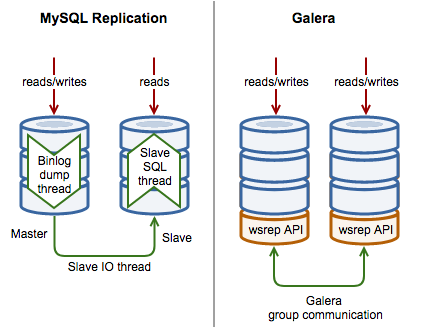
An application can write to any node in a Galera cluster, and transaction commits (row-based replication events) are then applied on all servers, via a certification-based replication. Certification-based replication is an alternative approach to synchronous database replication using Group Communication and transaction ordering techniques.

A minimal Galera cluster consists of 3 nodes and it is recommended to run with odd number of nodes. The reason is that, should there be a problem applying a transaction on one node (e.g., network problem or the machine becomes unresponsive), the two other nodes will have a quorum (i.e. a majority) and will be able to proceed with the transaction commit.

This plug-in is open-source and developed by Codership as a patch for standard MySQL. There are 3 Galera variants - MySQL Galera Cluster by Codership, Percona XtraDB Cluster by Percona and MariaDB Galera Cluster (5.5 and 10.0) by MariaDB. Starting from MariaDB Server 10.1, the Galera is already included in the regular server (and not anymore in a separate cluster version). Enabling Galera cluster is just a matter of activating the correct configuration parameters in any MariaDB Server installation.

## **3. Difference between MySQL Replication and Galera Cluster**

The following diagram illustrates some high-level differences between MySQL Replication and Galera Cluster:



### **3.1. MySQL Replication Implementation**

MySQL uses 3 threads to implement replication, one on the master and two on the slaves per master-slave connection:

* Binlog dump thread - The master creates a thread to send the binary log contents to a slave when the slave connects. This thread can be identified in the output of SHOW PROCESSLIST on the master as the Binlog Dump thread.
* Slave IO thread - The slave creates an IO thread, which connects to the master and asks it to send the updates recorded in its binary logs. The slave I/O thread reads the updates that the master's Binlog Dump thread sends and copies them to local files that comprise the slave's relay log.
* Slave SQL thread - The slave creates an SQL thread to read the relay log that is written by the slave I/O thread and execute the events contained therein.

MySQL Replication is part of the standard MySQL database, and is mainly asynchronous in nature. Updates should always be done on one master, these are then propagated to slaves. It is possible to create a ring topology with multiple masters; however, this is not recommended as it is very easy for the servers to get out of sync in case of a master failing. With introduction of GTID in MySQL 5.6 and later, it simplifies the management of the replication data flow and failover activities in particular, however, there is no automatic failover or resynchronization.

### **3.2. Galera Cluster Implementation**

Galera Cluster implements replication using 4 components:

* Database Management System - The database server that runs on the individual node. The supported DBMS are MySQL Server, Percona Server for MySQL and MariaDB Server.
* wsrep API - The interface and the responsibilities for the database server and replication provider. It provides integration with the database server engine for write-set replication.
* Galera Plugin - The plugin that enables the write-set replication service functionality.
* Group Communication plugins - The various group communication systems available to Galera Cluster.

A database vendor that would like to leverage Galera Cluster technology would need to incorporate the WriteSet Replication (wsrep) API patch into its database server codebase. This will allow the Galera plugin which works as a wsrep provider to communicate and replicate transactions (writesets in Galera terms) via group communication protocol. This enables a synchronous master-master setup for InnoDB. Transactions are synchronously committed on all nodes.

In case of a node failing, the other nodes will continue to operate and kept up to date. When the failed node comes up again, it automatically synchronizes with the other nodes through State Snapshot Transfer (SST) or Incremental State Transfer (IST) depending on the last known state, before it is allowed back into the cluster. No data is lost when a node fails.

Galera Cluster makes use of certification based replication, that is a form of synchronous replication with reduced overhead.

## **4. What is Certification based Replication?**

Certification based replication uses group communication and transaction ordering techniques to achieve synchronous replication. Transactions execute optimistically in a single node (or replica) and, at commit time, run a coordinated certification process to enforce global consistency. Global coordination is achieved with the help of a broadcast service, that establishes a global total order among concurrent transactions.

Pre-requisites for certification based replication:

* database is transactional (i.e. it can rollback uncommitted changes)
* each replication event changes the database atomically
* replicated events are globally ordered (i.e. applied on all instances in the same order)

*The main idea is that a transaction is executed conventionally until the commit point, under the assumption that there will be no conflict. This is called optimistic execution. When the client issues a COMMIT command (but before the actual commit has happened), all changes made to the database by the transaction and the primary keys of changed rows are collected into a writeset. This writeset is then replicated to the rest of the nodes. After that, the writeset undergoes a deterministic certification test (using the collected primary keys) on each node (including the writeset originator node) which determines if the writeset can be applied or not.*

*If the certification test fails, the writeset is dropped and the original transaction is rolled back. If the test succeeds, the transaction is committed and the writeset is applied on the rest of the nodes.*

*The certification test implemented in Galera depends on the global ordering of transactions. Each transaction is assigned a global ordinal sequence number during replication. Thus, when a transaction reaches the commit point, it is known what was the sequence number of the last transaction it did not conflict with. The interval between those two numbers is an uncertainty land: transactions in this interval have not seen the effects of each other. Therefore, all transactions in this interval are checked for primary key conflicts with the transaction in question. The certification test fails if a conflict is detected.*

*Since the procedure is deterministic and all replicas receive transactions in the same order, all nodes reach the same decision about the outcome of the transaction. The node that started the transaction can then notify the client application if the transaction has been committed or not*.

<https://severalnines.com/resources/tutorials/galera-cluster-mysql-tutorial>