Galera Replication – Best Practices

Seppo Jaakola
Codership
Finland

Keywords: MySQL, Replication, High Availability, Scalability, Clustering

Introduction

Galera is an external replication library, which can connect several MySQL/InnoDB instances to operate seamlessly together as synchronous multi-master cluster. Application can connect to any of the MySQL servers in the cluster and use the server for both reads and writes. Galera replication operates in the background and makes sure all writes will be broadcasted to the whole cluster without delay, in synchronous manner. Note that each node can be used also for write access, possible write-write conflicts will be resolved automatically.

Galera replication has following benefits:

- **no slave lag -** all nodes are constantly up to date with no delays
- **no lost transactions** – committed transaction is immediately safe in all cluster nodes
- **automatic node join** – new mysql server can join the cluster automatically
- **scalabiltiy** – Galera is very efficient making it possible to scale both reads and writes
- **WAN replication** – Galera works in LAN, WAN or cloud installations

Galera Cluster Architecture

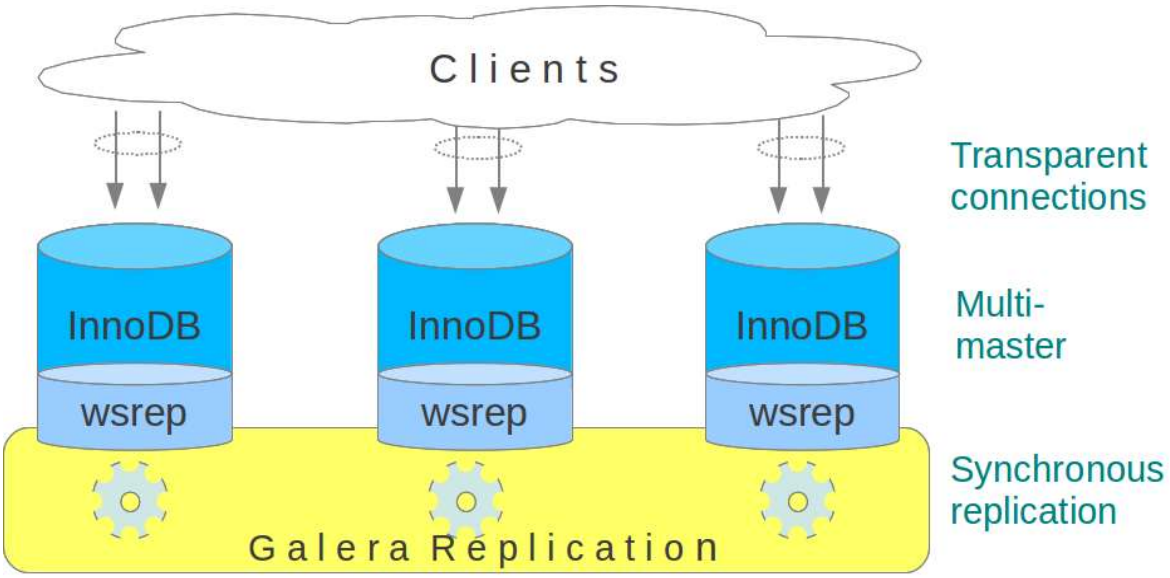

Illustration. 1: Galera Cluster Architecture

Galera Replication is external replication utility and uses replication API for interacting with DBMS. Our replication API definition is a separate open source project and we call it: wsrep API (as a shorthand for “Write Set REPlication API”). We have enhanced MySQL server with the implementation of wsrep API.

- MySQL/InnoDB implements the replication API (wsrep) and this makes it possible to utilize Galera replication.
- Clients can connect to any node in the cluster and issue both read and write queries. For application, the connection to the MySQL server looks like regular MySQL connection.
- There can be any number of nodes in the cluster and they operate conceptually in master role.
- Replication is synchronous.

Cluster Topologies

As the cluster nodes operate as regular MySQL servers, it is possible to build many different topologies for different use cases. The presentation will show different use cases and cluster topologies.

Here are some guiding principles when designing cluster topologies:

- For high availability reasons, there should always be at least three nodes in the cluster. It is possible to use a light weight arbitrator daemon to play the role of the third node, and with this the minimal configuration is two MySQL nodes and one arbitrator daemon.
- There is no upper limit for the size of the cluster. However if cluster is configured to use TCP transport for replication, the replication overhead will increase slightly every time when node count in the cluster goes up. Galera can also use multicast transport for replication, and in with that very large clusters can be built.
- Cluster can be used in multi-master or master-slave mode. The decision about masters is fully in the application side. The nodes where application sends writes to, operate as masters. And if writes are directed to only one node, then the cluster happens to operate as master-slave topology. The master fail over is lightning fast, cluster is always ready to accept writes at any node, so application must just decide to direct writes to some other node, and master fail over is completed.
- Cluster can also be distributed closer to the application. One viable topology is to have one cluster node running in same server as each application server. The benefit here is that application to MySQL communication path is minimal (no hops). As usually most data access is reads, this can save quite a lot of round trip time. Otoh MySQL and application server's may compete about server resources, so this architecture is not general purpose.
- Galera replication works fine or WAN connections as well. It is possible to build widely distributed clusters over WAN, and synchronous replication performance is on good level with many use cases. Note that long WAN latencies affect only commit operations, reads from the local node will be fast, and most data access is usually reading.
- For hardened security, replication transport can be configured to use SSL. It is also possible to run cluster over VPN tunnels.

Connecting to the Cluster

There are many alternatives for connecting to the Galera Cluster. The presentation will discuss the various connection methods, and pros and cons for each method:

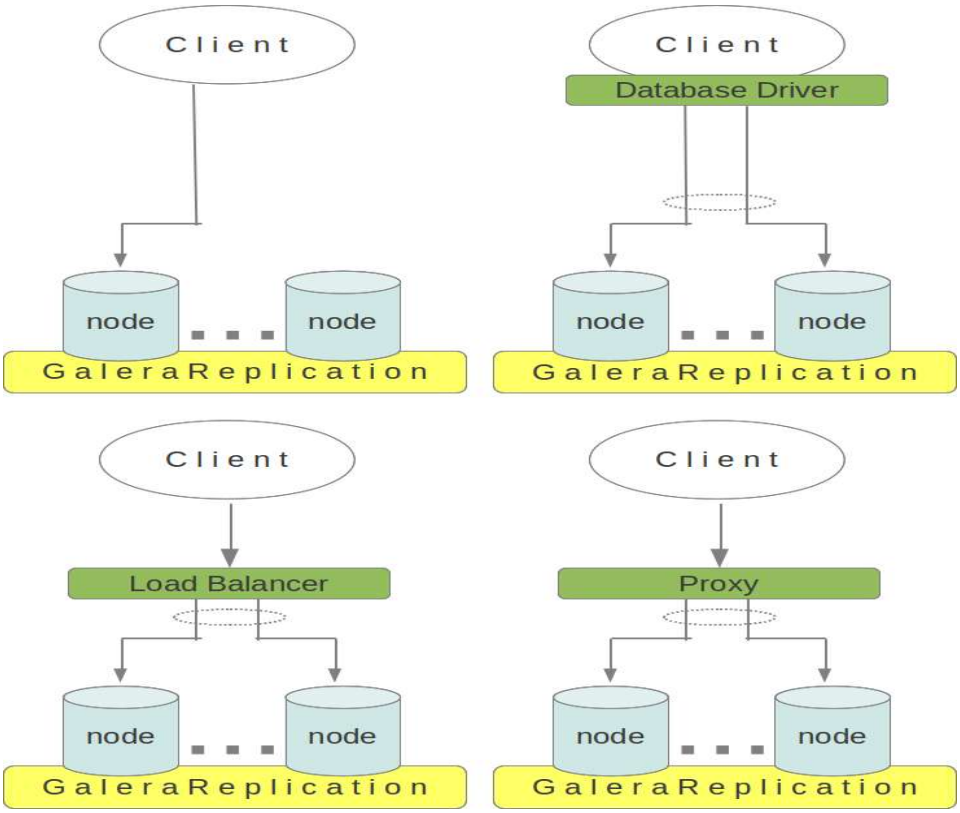

Illustration. 2: Galera Cluster Connectivity Alternatives

- Clients can connect directly to individual cluster nodes and this is the fastest communication path to the cluster. However, client must understand the cluster configuration and fail over to use another node if current host node drops from the cluster. This connection method is possible only for applications which can implement this cluster fail over functionality internally.
- For java and PHP clients there is possibility to have the cluster fail over functionality in the database driver level. Both Connector/J and mysqlnd_ms can be configured to manage cluster connections transparently to the application.
- Load balancer can implement cluster fail over for application. Load balancer will add one hop in the communication path so there will be unavoidable extra delay no matter how good load balancing solution is being used. There is a wide variety of load balancing solutions to choose from. Hardware based load balancing is fastest method but requires extra devices in the engine room. The load balancer should be runtime configurable, so that cluster configuration changes can be updated immediately for the load balancer.
- Proxy is the slowest alternative but offers the possibility to add intelligence in the load balancing. Proxy can do e.g. read/write splitting or avoid hot spots in the database access. Viable proxy alternatives are ScaleBase, which has exceptionally low overhead for performance and MySQL Proxy.

Joining New Nodes in the Cluster

New nodes joining in the cluster is an automatic process. To join a new node you just need to start the node and pass one valid IP address of the active cluster for initiating handshake with the cluster. The node joining process will go through following steps:

1. handshake with the cluster
2. cluster will elect one node to operate as state donor for the joining node
3. state transfer to the joining node will happen either as full state transfer (SST) or, if joining node already had some old state, a lightweight incremental state transfer (IST) can be used
4. catchup phase in the joining node will replay all missed transactions during the state transfer

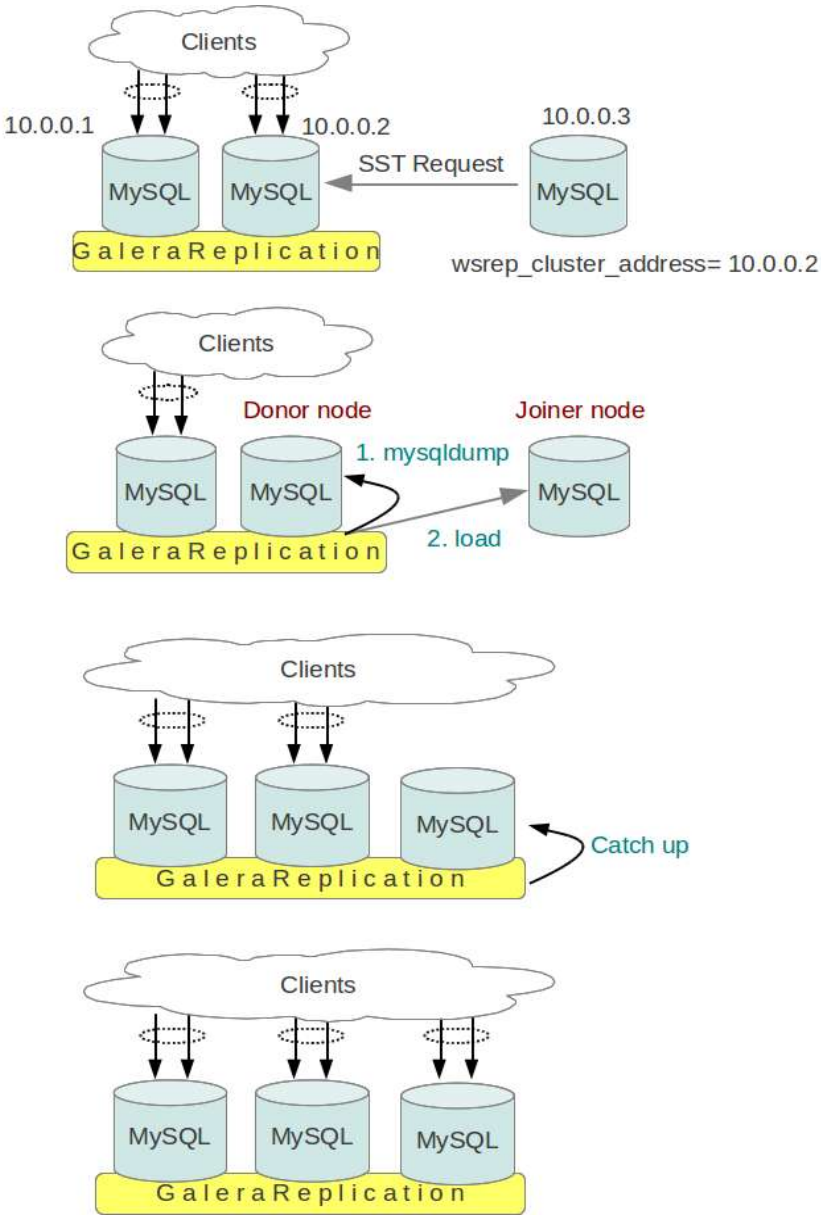

Illustration. 3: Node Join Process in Galera Cluster

There are three different methods for data synchronization (SST):

- mysqldump takes logical backup of the donor node and loads the dump to joining node
- rsync method stops the donor in quiescent state and sends the MySQL data directory to joining node. This is considerably faster than mysqldump, in general case. Joiner can accept rsync state transfer only at MySQL server startup phase, you cannot adapt rsync transfer to a running server.
- Xtrabackup method uses Percona's xtrabackup tool to take a hot backup from donor node and loads the backup to joiner node. This is least interfering method for donor node, the backup can be taken while the donor is active.

Schema Upgrades

Schema upgrades happen basically by data definition language (DDL) statements. Galera cluster has two strategies for processing schema upgrades:

- Total Order Isolation (TOI) method, processes the DDL statement in all cluster nodes within same global order transaction slot, and by isolated so that no conflicting operation can happen at the same time. This method will lock the affected table or database for the duration of the DDL execution in all cluster nodes. If DDL operation is long term and the locked table would be needed for other transactions, then TOI method is probably not a suitable choice.
- Rolling Schema Upgrade (RSU) method processes the DDL only in the node where DDL was issued, and furthermore the node is dropped from the clustering for the duration of the DDL operation. To use RSU, you must run the same DDL in all cluster nodes, one by one. Also the change for schema must be upwards compatible. Incoming transactions must succeed both for the old and new schema definition, as long as there are node upgrades happening.

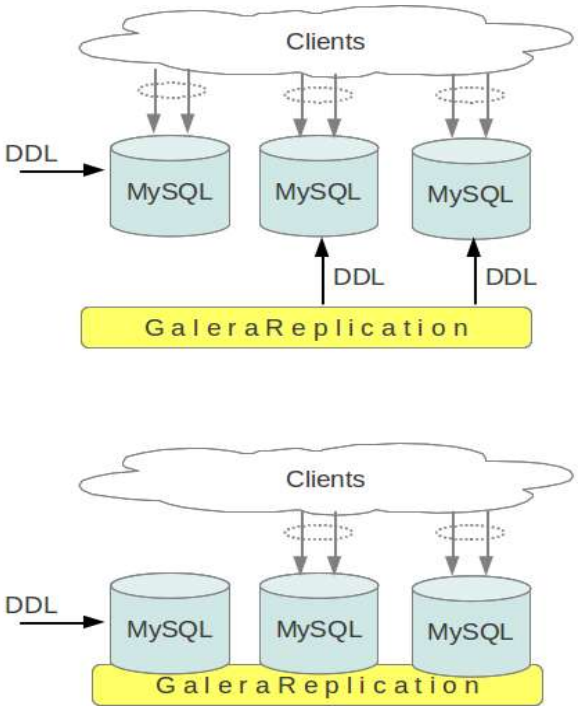

Illustration. 4: Online Schema Upgrade Methods (TOI and RSU)

There is configuration variable (wsrep_osu_method) for choosing the effective schema upgrade method for future DDL statements.

Backups

Basically all cluster nodes can be used for taking backups. However, it makes sense to take backups in such a way that the backup can be associated with Galera global transaction ID. Backups taken from a known point in the transaction history can be used .e.g. for promoting new nodes in the cluster with least effort.

Here are some methods for taking global transaction ID aware backups:

1. All cluster nodes contain up to date data. To take a backup it is sufficient just to deliberately drop a node from cluster and use whatever backup method on the idle dropped node. After backup operation the node can join back by using light weight IST method.
2. Use xtrabackup with –galera-info option
3. Use SST scripting interface and write a specific wsrep_sst_backup script for bringing the node in quiescent state for the backup

The presentation discusses different backup methods in detail.

Contact address:

Seppo Jaakola
Codership
Pohjolankatu 64 A
00600 Helsinki
Finland

Phone: +358(0)40-510 5938
Fax:
Email seppo.jaakola@codership.com
Internet: http://www.codership.com