MySQL Replication vs Galera: which is better for your workload?

Nickolay Ihalainen (Percona)
Replication is better than standalone

- Standalone
- Less parts leading to less faults
- New transactions reading committed data instantly
- You know where to find a database
Replication is better than standalone

- Database service is alive on multiple node faults
- Easy to increase READ performance by adding servers
- Proxy servers distribution the load
Statement (SQL) replication

• Non-deterministic queries crashing replication
  – UUID(), RAND()
  – DELETE/UPDATE + LIMIT без ORDER BY

• READ COMMITTED / READ UNCOMMITTED
ROW based replication

• How to find a row?
  – Primary Key!

• What to transfer?
  – row before
  – row after
  – partial row
Replication files

• binary log
• relay log
• master.info relay-log.info
  – could be stored inside InnoDB tables
Replication: faults

• semi-sync
  - Does the slave receive replication event?
  - COMMIT waits for the confirmation from at least single slave
    - rpl_semi_sync_master_wait_for_slave_count

• crash-safe slave (default 8.0)
  - relay-log-info-repository=TABLE
  - relay-log-recovery=ON
Replication: group commit

- InnoDB => MySQL: Transaction is ready!
  - PREPARE
    - multiple transactions committed in parallel
- MySQL => InnoDB: binlog fsync finished
- binlog_group_commit_sync_delay
- binlog_group_commit_sync_no_delay_count
Replication: GTID

- binlog file name and position is different everywhere
- server’s UUID + seqno(transaction on server)
- Snapshot identified by: (uuid1:seq1, uuid2:seq2...)
  - long sets...
- CREATE SELECT – problem
- tmp tables in transactions – also not supported
Replication: WRITESET

• Replication event:
  − db.table.PK
  − row
• For each row calculate XXHASH64(PK)
• Foreign keys are not supported
Parallel slave

• `slave_parallel_type`
  - DATABASE
  - LOGICAL_CLOCK

• `binlog_transaction_dependency_tracking`
  - COMMIT_ORDER
  - WRITESET, WRITESET_SESSION
It’s almost a multi-master!

• Synchronous multi-master is a reliable solution
  – transaction write conflict? ROLLBACK
  – error applying the row?
    • recreate the node
Group Replication

• New network protocol
  – eXtended COMmunication
• cluster members state monitoring
• single-primary, multi-primary
  – primary role switchover
Group Replication: XCom

• Based on Paxos (Mencius)
• Strict transaction commit order in cluster
• Dynamic membership
• Fault detection
Paxos

1) Elect the leader
2) Transfer the transaction
3) Majority accepts the transaction
4) COMMIT!
Multi-Paxos

- Elect the leader only once
- Skip elections (prepare)
- Leader proxies messages
Mencius

• Every node has a leader slot
• nothing to suggest?
  – send a SKIP
XCom: optimizations

• handles empty slots
• batch processing for multiple transactions
• full transaction data transferred just once
XCom: restrictions

• up to 9 nodes
• long message processing? Node evicted from cluster
  - group_replication_member_expel_timeout = 5 seconds
  - group_replication_transaction_size_limit = 143MB
  - group_replication_communication_max_message_size = 10MB
Group Repl.: Single Primary

• Better than async replication!
  - automatic recovery after fault
  - guarantees the same row values on all nodes
Group Repl.: Multi Primary

- Advanced mode
- no Gap Locks, READ COMMITTED
- no SERIALIZABLE
- DDL – problem
- FK – problem
Group Repl.: consistency

• `group_replication_consistency`
  - EVENTUAL – do not wait
  - BEFORE_ON_PRIMARY_FAILOVER
  - BEFORE – wait for previous transactions to commit
  - AFTER – wait for transaction to be applied everywhere
  - BEFORE_AND_AFTER
Group Repl.: instrumentation

- Performance Schema tables
  - replication_group_member_stats
  - replication_group_members
  - replication_connection_status
  - replication_applier_status
Repair: incremental

- Do you have a server with older GTID?
- Do you have old binary logs?
- Applies the difference between old state and current cluster state incrementally
Repair: manual

- `mysqldump`
  - too slow for real databases
  - `mysqlpump` – better

- `Xtrabackup`
  - same as GTID-replication setup

- MySQL Enterprise Backup
  - not open source, subscription required
CLONE Plugin

- Similar to MEB/Xtrabackup
- FILE COPY
- PAGE COPY
- REDO COPY
InnoDB Cluster: MySQL Router

- proxy MySQL network protocol
- Monitors cluster membership
- Run it directly on application server
- Different TCP ports for RW and RO
InnoDB Cluster: MySQL Shell

• X Dev API
• Admin API
• Shell API
• SQL
• Python & JavaScript library
InnoDB Cluster: MySQL Shell

- Checks server configuration
- fixes the configuration with mysql-auto.cnf
- Creates the cluster
- Can add new nodes with CLONE
- cluster.status()
Summary: InnoDB Cluster

• Over 3 years after release, many changes implemented during last year
• Use with MySQL 8.0.17+ !
• There is no WAN optimizations
• Good encryption for network and storage
• NoSQL by using X protocol
Percona XtraDB Cluster

• Synchronous replication
• Since 2012 (5.5)
• Galera-based
• Current development focus:
  - autonomous usage (reduce ops)
  - fix bugs
Galera

- Full database on each node
- Slow as a slowest node
- Virtually synchronous
- Error for each query after losing quorum on the node
- COMMIT – can return error
- COMMIT – at least RTT long
- Problematic with large transactions (improved with Galera 4/PXC8)
Galera: Binlog

• binary logs are not used directly
• binlog could be disabled
• Uses hooks in InnoDB code
• ROW events saved in Gcache
• write-set: all rows modified by transactions
Galera: DML processing

• BEGIN;
• queries...
• COMMIT:
  – Extract write-set
  – Get a Transaction ID
  – write-set transfer
  – wait for certification
  – return OK to the client
Galera: consensus, trx id

• Totem single-ring ordering and membership
• Every node certifies all transactions
• seqno incremented globally in the cluster
Galera: *_seqno

- global_seqno (x,y,z)
- local_seqno n1(a,b,c) n2(a,p,r) n3(m, n, o)
- last_seen_seqno
  - for the trx under certification
  - helps to detect certification boundaries
- depends_seqno
Galera: Flow Control

• Async write-set copy
• Async apply
  – global transaction ordering
• Receive queue
  – Flow Control (PXC: 100+ transactions)
Galera: readings after DML

• Virtually synchronous

• `wsrep_sync_wait` – SELECT waits until proper seqno on all nodes

• Galera4/PXC8: functions for wait
  
  ```
  $transaction_gtid = SELECT WSREP_LAST_SEEN_GTID();
  SELECT WSREP_SYNC_WAIT_UPTO_GTID($transaction_gtid);
  ```
PXC: ProxySQL

• Intelligent load balancer
• Implements MySQL network protocol
• Can balance:
  - queries
  - transactions
  - users
• SELECT / INSERT+UPDATE+DELETE separation
ProxySQL: setup

• Clustering: multiple ProxySQL servers
  - Automatic sync for settings and state
  - not a single point of failure
  - you can run it on application server directly
• Uses mysql network protocol for configuration MySQL, TCP/6032.
ProxySQL: setup

• Stores details for all nodes at `mysql_servers`
• checks node availability
• Multiple server roles (reader, writer, backup writer) in `mysql_galera_hostgroups`
ProxySQL: users

• ProxySQL stores all users in `mysql_users`
• MySQL should have same users and passwords
  - add users on one PXC node
  - setup access rights (GRANT/REVOKE) in MySQL
• `proxysql-admin --config-file=/etc/proxysql-admin.cnf --syncusers`
ProxySQL: routing

• mysql_query_rules:
  - SELECT: processed by «readers»
  - SELECT ... FOR UPDATE: processed by «writer»
  - other queries: processed by writer to reduce conflicts

• Query routing:
  - RegEx
  - by user name (prod_ro, prod_rw)
PXC: WAN

- Voting weights to calculate quorum
- Arbiter
- Multiple settings for different timeouts
- Segments: reduce WAN traffic
PXC: DDL

• Total Order Isolation
  - block all queries on all nodes
  - wait for ALTER TABLE applied in parallel everywhere
  - pt-online-schema-change helps a lot

• Rolling Schema Upgrade
  - apply node by node
  - hard to use
PXC: Recovery

- SST: Full backup and restore
  - xtrabackup
  - rsync (disabled in PXC8 due to REDO logging changes)
  - mysqlbackup (deprecated, removed from PXC8)
- IST: incremental
  - node gets the difference from donor’s Gcache
PXC: version upgrades

• Major version:
  - Stop whole cluster
  - update OS packages
  - start without galera: --wsrep-provider='none'
  - mysql_upgrade
  - repair other nodes by SST
PXC: version upgrades

- Minor:
  - stop the node
  - upgrade OS packages
  - start without galera: --wsrep-provider='none'
  - mysql_upgrade
  - repeat operation on other nodes
PXC8: version upgrades

• Major and minor:
  - JOINER can connect to older cluster
  - After SST: mysql_upgrade stats automatically

• Is a DONOR an async slave?
  - RESET SLAVE ALL executed automatically
PXC: instrumentation

• Performance Schema
  - wait & stage instruments
  - mutex/cond variables
  - files
  - threads

• SHOW STATUS
  - Used by Percona Monitoring and Management
  - PXC8: wsrep_monitor_status
Galera4/PXC8: instrumentation

- mysql.wsrep_cluster
- mysql.wsrep_cluster_members
- mysql.wsrep_streaming_log
PXC8: big transactions

- Galera 4 feature
- Streaming Replication
  - splits transaction in parts
  - after first part certification
    - conflicting transactions are rolled back
    - use READ COMMITTED!
PXC8: Streaming replication

- Too many rows:
  - fragment replicated before the COMMIT

- Hot rows
  - Use manual SR to get high priority lock:
  - `START TRANSACTION;`  
    `SET SESSION wsrep_trx_fragment_unit='statements';`  
    `SET SESSION wsrep_trx_fragment_size=1;`
How to evaluate replication?

- Kubernetes
  - Percona K8S Operator for PXC
  - MySQL Operator
- dbdeployer
dbdeployer

- Linux or OS X

$ VERSION=1.42.0
$ OS=linux
$ origin=https://github.com/datacharmer/dbdeployer/releases/download/v$VERSION
dbdeployer/downloads/list
$ wget $origin/dbdeployer-$VERSION.$OS.tar.gz
$ tar -xzf dbdeployer-$VERSION.$OS.tar.gz
$ chmod +x dbdeployer-$VERSION.$OS
$ sudo mv dbdeployer-$VERSION.$OS /usr/local/bin/dbdeployer
$ dbdeployer downloads list
dbdeployer: download

- get a tar.{gz,xz} from the official site

- Check libraries with: ldd bin/mysqld

  $ dbdeployer downloads list
  $ dbdeployer downloads get \ mysql-8.0.18-linux-glibc2.12-x86_64.tar.xz
  $ dbdeployer unpack \ mysql-8.0.18-linux-glibc2.12-x86_64.tar.xz
Innodb Cluster

- Locally
- Uses different TCP ports

```
dbdeployer deploy --topology=group \replication 8.0.18 --single-primary
```
PXC

- PXC8 currently a bit more complex

```
dbdeployer deploy --topology=pxc \ replication pxc5.7.27
```
Summary

<table>
<thead>
<tr>
<th>Feature</th>
<th>PXC</th>
<th>InnoDB Cluster</th>
</tr>
</thead>
<tbody>
<tr>
<td>Automatic recovery</td>
<td>+</td>
<td>8.0.17</td>
</tr>
<tr>
<td>Load balancer</td>
<td>ProxySQL</td>
<td>MySQL Router</td>
</tr>
<tr>
<td>Multi-Master</td>
<td>+</td>
<td>default: single</td>
</tr>
<tr>
<td>API/cmd for control</td>
<td></td>
<td>mysqlshell</td>
</tr>
<tr>
<td>WAN</td>
<td>+</td>
<td></td>
</tr>
<tr>
<td>Big transactions</td>
<td>8.0</td>
<td></td>
</tr>
<tr>
<td>“mature”</td>
<td>+</td>
<td></td>
</tr>
<tr>
<td>Supported by Percona</td>
<td>+</td>
<td>+</td>
</tr>
</tbody>
</table>
Questions?

• ?

• ?

• nickolay.ihalainen@percona.com