Group Replication: A Journey to the Group Communication Core

Alfranio Correia (alfranio.correia@oracle.com)
Principal Software Engineer
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.
Program Agenda

1. Background
2. Group Communication Interface
3. Group Communication Engine
4. Performance
5. Conclusion
Background
MySQL InnoDB Cluster
MySQL Group Replication

• What is MySQL Group Replication?
  “Multi-master update everywhere replication plugin for MySQL with built-in automatic distributed recovery, conflict detection and group membership.”

• What does the MySQL Group Replication plugin do for the user?
  – Automates server failover in Single Primary
  – Provides fault tolerance
  – Enables update everywhere setups
  – Automates group reconfiguration (handling of crashes, failures, re-connects)
  – Provides a highly available replicated database
The Complete Stack

MySQL Server

API
Replication Plugin
API

Performance Schema Tables: Monitoring

MySQL

APIs: Lifecycle / Capture / Applier

Capture
Conflicts Handler
Applier
Recovery

Replication Protocol

Group Com. API

Group Com. Binding

Group Com. Engine

Network
Group Communication Interface
Design

• Abstract interface to support different solutions
  – Reconfigure the group and get membership information
  – Send and receive messages

• Uses the observer pattern
  – MySQL Group Replication listens to events

• Different implementations per Communication Systems

• Made the transition from Corosync easy
Semantics

• Closed Group
  – Only group members can send and receive messages

• Total Order
  – Messages are totally ordered among each other

• Safe Delivery
  – One cannot deliver a message if the majority can’t do so

• View Synchrony
  – Changes to membership are totally ordered with messages
Group Communication Engine
Built-in Communication Engine

- Based on proven distributed systems algorithms (Paxos)
  - Compression, multi-platform, dynamic membership, SSL, IP whitelisting
- No third-party software required
- No network multicast support required
  - MySQL Group Replication can operate on cloud based installations where multicast is unsupported
Paxos Family and Friends

- Multi-Paxos
- Cheap Paxos
- Flexible Paxos
- Generalized Paxos
- Fast Paxos
- Menclus
- Egalitarian Paxos
- Disk Paxos
- Byzantine Paxos
- Vertical Paxos
- Raft
Basic Paxos

• Get agreement on a value:
  – Next message/transaction to be delivered

• Members may have different roles:
  – Usually all members are proposers, acceptors and learners

• Need a quorum to make progress
  – Usually a majority
Prepare Phase

- Proposer sends a prepare request with number “n” to members (i.e. acceptors)
- If an acceptor has not received a request with a number greater than “n”, it will respond
- It will promise not to accept a request numbered less than “n”
- If the reply has a non-empty value, the leader will use that with the highest number
Accept Phase

- If the leader finds out that a non-empty value has been previously proposed, it will use it.
- Otherwise, it will propose a new value.
- Requires a network round-trip to get agreement.
Learn Phase

• It will inform other members about the decision
• Only one learner is required to have progress
• If the member already has the value, an ack is enough
Multi-Paxos

- Consensus round to decide on each slot’s content
- Replicated Log Stream
So what?

- They can easily become a bottleneck
- Multiple leaders: eXtended COMmunications
How does XCOM work?

- Every member is a leader so no leader election
- Every member owns a In-Memory Replicated Log
Nothing to Propose

- Only a learn message with a “nop” is enough
How is the optimization possible?

- Member “1” sends a learn message “(0, nop)” to member “4” and dies
- Non-leaders can only propose “nop”(s) on behalf of others
- They must go through all Paxos phases
Handling Failures/Suspicions
Implemented Optimizations in XCOM

• Pipeline
  – Proposes several “transactions” in parallel
  – Improves performance in high latency networks
  – Current value is “10”

• Batch
  – Improves CPU usage
  – Improves performance in high latency/low bandwidth networks
  – Current value is “5”
Implemented Optimizations in Biding

• Compression
  – Reduces bandwidth consumption

• Automatically reconfigure a group
  – Faulty members are expelled
Performance
Configuration

- Multiple writers – One per Server
- Single writer – Just one client
- Oracle Server X5-2L with two Intel Xeon E5-2660-V3 processors
  - 20 Cores
  - 40 Hardware Threads
- Oracle Enterprise Linux 7, kernel 3.8.13-118.13.3
- 10 Gbps ethernet
- Used “tc” to throttle network
Multiple writers (256 Bytes)

- Compression improves performance in Metropolitan
- Headers are not compressed (~200 bytes) though

![Bar chart showing performance comparison between uncompressed and compressed payloads with different network latencies and number of members.]

- **Uncompressed 256 byte payload**
  - 3 members: 10Gbps network with 0.1ms latency (~140,000 messages per second)
  - 5 members: 10Gbps network with 0.1ms latency (~120,000 messages per second)
  - 7 members: 10Gbps network with 0.1ms latency (~100,000 messages per second)

- **Compressed 256 byte payload**
  - 3 members: 200Mbps network with 7ms latency (~80,000 messages per second)
  - 5 members: 200Mbps network with 7ms latency (~60,000 messages per second)
  - 7 members: 200Mbps network with 7ms latency (~40,000 messages per second)
Multiple writers (1K Bytes)

- Check whether compression may help or not
- Usually helps when bandwidth is a problem

![Graph showing messages per second sent for different network conditions and member counts for uncompressed and compressed 1K payload data.]

- 10Gbps network with 0.1ms latency
- 200Mbps network with 7ms latency

### Messages per second sent

<table>
<thead>
<tr>
<th>Members</th>
<th>Uncompressed 1K payload</th>
<th>Compressed 1K payload</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>100000</td>
<td>34364</td>
</tr>
<tr>
<td>5</td>
<td>100000</td>
<td>34364</td>
</tr>
<tr>
<td>7</td>
<td>100000</td>
<td>34364</td>
</tr>
</tbody>
</table>

---

31 of February  Copyright © 2017, Oracle and/or its affiliates. All rights reserved. | Oracle / Fosdem 2017
Single Writer (1K Bytes)

- The scale out effect with multiple writers is small
- Compression does not help here

![Graph showing performance metrics for single writer with different network conditions and member counts.](image)

- **Uncompressed 1K payload**
  - 3 members
  - 5 members
  - 7 members

- **Compressed 1K payload**
  - 3 members
  - 5 members
  - 7 members

Messages per second sent:

- 10Gbps network with 0.1ms latency
- 200Mbps network with 7ms latency
Conclusion
Current Status

• Has made into MySQL 5.7.17 release
• GA in December 2016
Future

• Configurable Paxos role(s)
  – Leader/Acceptor/Learner or Acceptor/Learner or Learner

• Multiple leaders only if needed:
  – Avoids the skip message
  – Improves CPU and network usage

• Not all members need to make messages network durable
  – Reduces resilience but improves performance
Future

• Expose some configuration options:
  – Batch
  – Pipeline

• Compression at low level layers as well

• Write to network in parallel

• Overlay networks
Where to go from here?

• Packages

• Documentation

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