facebook
MySQL Replication and HA at Facebook
Part-II

Jeff Jiang
Production Engineer
Facebook, Inc
jjj@fb.com
Agenda

❖ MySQL HA: theory and Facebook solutions
❖ Facebook MySQL HA Automations
  • MySQL replication management at Facebook
  • FB MySQL Semisync and strong consistent failovers
❖ Disaster Recovery Practices
  • Enforcement of Semisync failure domains
  • Maintain availability during power loss and network cut
  • Practice disasters: large scale testbed and drills
MySQL HA: theory and Facebook solutions
MySQL HA: the theory

- Master-Slave replication + Master Failover = MySQL HA
  - A single MySQL instance is not reliable
    - In contrast, a group of MySQL instances are more reliable
    - MySQL master-slave replication spins up a group of instances
  - A single MySQL master is not reliable
    - If a group of instances are available, we can failover
MySQL HA: Facebook solution

- Master-Slave replication + Master Failover = MySQL HA
  - Master-Slave asynchronous replication to achieve read HA
  - Master failover to achieve write HA
  - Lossless MySQL Semisync to achieve data consistency
- At Facebook, we develop automations to manage replications and master failovers
MySQL HA automations at Facebook
MySQL HA Automation: an overview

❖ Facebook HA automation is production driven
  • **Discovery**: automatic discovery of replication topology
  • **Monitoring**: actively polling the state of master and slave, trigger remediations and alerts when failure happens.
  • **Remediation**: automatically fixing issues
MySQL HA automation: discovery (1)

- To achieve high-availability, we create master-slave replication topology.
- The “model” of replication topology is defined in config manager service.
  - Where is the master? where are the slaves?
  - How many slaves are in location X?
- The materialized topology is stored in the discovery service.
MySQL HA automation: discovery (2)

Discovery of master/slaves is critical for both clients and automations

Config Manager Service

Preferred Master: California
Fallbacks: Iowa, Oregon
Read-only: Sweden

Clients and Automations

Discovery Service
MySQL HA automation: monitoring (1)

- Planet-scale materialized replication topologies have to be monitored
  - Many master-slaves replication topologies: The Replicaset
  - Failures are frequent and normal
- DBStatus: distributed Facebook’s MySQL replication monitoring
  - Monitoring replication behavior on a single node
  - Quorum based voting to decide the topology’s healthiness
MySQL HA automation: monitoring (2)

Once replication topology is discovered, we need to monitor it.
MySQL HA automation: monitoring (3)

- Different roles of DBStatus on master and slaves
  - DBStatus on slave is responsible for monitoring the replication status of the slave itself
  - DBStatus on master is responsible for monitoring that quorum of the slaves are online and healthy
  - DBStatus on slaves also send heartbeat writes to master
  - All DBStatus polls master status from others and vote for master being offline
MySQL HA automation: remediation (1)

Large scale auto-alarming naturally leads to large scale auto remediation

- Human DBAs cannot effectively deal with regular failures / disasters from a planet scale fleet
- At Facebook, we automate the traditional DBA routines into DBStatus to automatically remediate most failures
  - Disable/replace bad slaves
  - Master failover
  - Repoint slaves
MySQL HA automation: remediation (2)

Handling of a broken slave
MySQL HA automation: remediation (3)

But what if master dies? Automation does failovers: FastFailover

- DBStatus talks with each other and votes that master is offline
- One DBStatus gets the coordinator lock and elects the new master
- The coordinator DBStatus continues to finish the rest of master failover
  - Do replication catch-up on the candidate new master
MySQL HA automation: remediation (4)

Semisync is deployed to assist replication catchup in FastFailover

- Catch up candidate master with the offline master
  - lossless Semisync is deployed by developing Binlog Server (BLS)
MySQL HA automation: remediation (5)

Node-fence: another way of stopping writes on master

- Lossless Semisync in FB MySQL 5.6 waits for Semisync ack to come back to the master before engine commit

- **Node-fence** automation: stopping Semisync acking to effectively disable write on the master
  - Especially effective when master itself is inaccessible or cannot respond to ‘SET SUPER_READONLY = 1’
MySQL HA automation: remediation (6)

Case study: failover away from a broken master by node-fencing
MySQL HA automation: remediation (7)

Repointing of slaves are needed when network partition happens.

- Network partition can cause slave pointing to a previous master, repointing it back to the current master is the fastest remediation.
  - GTID auto-position makes repointing straightforward.
FastFailover and Semisync enhancements

Failover is easy, data consistency is not

- Async slaves can go ahead of Semisync
  - Sacrifice failover availability by enforcing check on all slaves?

- Semisync might be turned off accidentally
  - `rpl_semi_sync_master_enabled`
  - `rpl_semi_sync_master_timeout`

- BLS not in topology might still be aking the master
FB Semisync: Async Behind Semisync (1)

FastFailover only needs to check BLS during a failover

- Vanilla MySQL 5.6/5.7/8.0 does not guarantee that Semisync slaves are ahead of async slaves
  - Master prepares TX1 then dies, async slave gets TX1 but Semisync slave might not
  - Failover has to check ALL slaves to protect against phantom read
- FB MySQL can enforce that Async slaves are always behind of Semisync slaves
FB Semisync: Async Behind Semisync (2)

FastFailover only needs to check BLS during a failover

**Vanilla MySQL 5.6/5.7/8.0**

- **Master**
  - Prepare M:123
  - Binlog Commit M:123

- **BLS**

- **Slave**
  - Prepare M:123
  - Binlog Commit M: 123
  - Engine Commit M:123

**Async Behind Semisync**

- **Master**
  - Prepare M:123
  - Binlog Commit M:123

- **BLS**
  - M:123

- **Slave**
  - Prepare M:123
  - Binlog Commit M: 123
  - Engine Commit M:123

**Question:** what to do?

**Catch-up from BLS is enough**
FB Semisync: “Safe-Turnoff” of Semisync

No need to worry about Semisync is accidentally turned off

- Accidental turning off Semisync leads to data drift
  - On slaves, we turn off Semisync for replication performance
  - On masters, rpl_semi_sync_master_timeout may be set to a too short duration

- FB Semisync feature: server automatically exit when Semisync is turned off and there are pending transactions
  - Dynamic variable rpl_semi_sync_master_crash_if_active_trxs
At Facebook scale, BLS replacements is regular events
• Unhealthy BLS is removed from the Discovery Service

Automations might not be able to force strayed BLS to stop
• Strayed BLS might come back into life afterwards

FB MySQL enforces that only acks from whitelisted Semisync slaves are respected by master
• Dynamic variable rpl_semi_sync_master_whitelist

FB Semisync: Semisync Whitelist (1)
BLS can become strayed and stealthily send acks to the master
FB Semisync: Semisync Whitelist (2)

Safe replacement of temporarily unresponsive Binlog Server
- BLS_B becomes unresponsive
- Replacement happens by updating Semisync Whitelist first
- Node-Fence happens
- BLS_B reconnects, and is rejected (master dump thread exits)

![Diagram](image)
FB Semisync: Trim Binlog To Recover (1)

Cleaning up the leftover of FastFailover is non-trivial

- After FastFailover, node-fenced instance cannot rejoin replication
  - Node-fenced instance cannot take replication writes
  - Executed_Gtid is ahead of storage engine on the instance
- FB MySQL truncates uncommitted transactions in Binlog during crash-recovery
  - Static flag trim-binlog-to-recover
  - Automation can then rejoin the slave instance into replication
FB Semisync: Trim Binlog To Recover (2)

Light-weighted recovery of node-fenced instance

- FastFailover happens
- New writes reaches original master
- Semisync master timeouts, master restarts
- Crash-recovery happens and prepared binlog is truncated
- Original master is repointed to the new master
MySQL Disaster Recovery at Facebook
MySQL Disasters: the killer of SLA

Maintaining 4 9s with disasters >> maintaining 6 9s without disasters

❖ Disasters is the 1st killer of SLA
  • A fleet with >48 min of downtime per year is below an SLA of 4 9s
  • A 10K-instances fleet with 1 instance always down is still above SLA of 4 9s

❖ Disasters for large scale MySQL deployments are unavoidable
❖ Disasters usually bring down many masters/slaves at once, and take longer to recover
Disasters are failures, but at large scale

Handling Disasters = Handling Failures At Large Scale

- At Facebook, Disaster Recovery automations are developed on top of solid regular HA mechanisms
  - **Enforce Semisync Failure Domains:** The support and deployment of different failure domains for CAP trade-offs
  - **Power Loss Signaling:** Special in-rack battery based mechanism to evacuate when AC power is lost
  - **DR Drills:** continuous drilling of doom-day scenarios
Enforce Semisync Failure Domains (1)

Failure domain is the container of failures

- Failure domain is the domain that confines defined failures
  - Rack
  - Datacenter building
  - Geographical Regions (Iowa, Oregon, etc)

- To survive disasters, master and its 2 Binlog Servers have to be deployed on 3 different failure domains
  - Still need to balance between commit latency and disaster risk
Enforce Semisync Failure Domains (2)

FB Datacenter Design And Failure Domains
Choose the right failure domain, then the work is almost done

- Choose the most suitable failure domain to deploy Semisync
- Balance between application’s requirement of commit latency and disaster recovery
  - In-DC and Cross AC-power Main Switch Board: latency < 125us
  - In-Region and Cross-DC building: 100us < latency < 250us
  - Cross-Region: 10ms < latency < 300ms
Power Loss Signaling (1)

Master evacuation during power outage

- Power Failures are common when there are lots of datacenters
- In Facebook Datacenters, racks are equipped with batteries to remediate power failures
  - Batteries bridge the power supply transition to generators
  - When rack is on battery power, a special GPIO pin signal is raised and BMC can read it
  - The power loss GPIO pin signal is relayed to all hosts under the rack
Power Loss Signaling (2)

Power Loss Signaling and MySQL failovers

- RSW runs gpio_mon on BMC
- BMC reads GPIO signals
- RSW detects GPIO changes
- RSW multicasts signal to hosts
- Hosts receives signals
- Hosts runs remediation
**Disaster Recovery Drills**

We need a way to test and exercise our disaster recovery solutions

- Disasters are relatively rare events
- Disaster Recovery solutions are complicated and has to be verified continuously
- At Facebook, we invest resource to do regular Disaster Recovery drills
  - The maintenance of large-scale Disaster Recovery testbed
  - Exercise of the existing disaster recovery solutions
  - Predict and test new ‘doom-day’ scenarios
Recap

❖ MySQL HA: theory and Facebook solutions
❖ Facebook MySQL HA Automations
  • MySQL replication management at Facebook
  • FB MySQL Semisync and strong consistent failovers
❖ Disaster Recovery Practices
  • Enforcement of Semisync failure domains
  • Maintain availability during power loss and network cut
  • Practice disasters: large scale testbed and drills
May our production be free of failures!
facebook