How to make MySQL work with Raft

Diancheng Wang & Guangchao Bai
Staff Database Engineer @ Alibaba Cloud
About me

- Name: Guangchao Bai
- Location: Beijing, China
- Occupation:
  - Staff Database Engineer @ Alibaba Cloud
  - Focus on MySQL kernel
Agenda

- Background
- ApsaraDB on the Alibaba Cloud
- Architecture of RDS Advanced Edition for MySQL
- Review of RAFT Algorithm
- Detailed implementation of MySQL-RAFT
Background

- Traditional master/slave mode

- Unfortunately something below may happen
  - Data loss
  - Data inconsistency between master and slave
ApsaraDB on the Alibaba Cloud
For your data safety, For your application stability

- 2003: * Internal business
- 2011: * RDS for MySQL 5.1
- 2014: * RDS for MySQL 5.6
- 2017: * RDS for MySQL 5.7  
  * RDS Advanced Edition for MySQL 5.6

https://github.com/alibaba/alisql
MySQL for Cloud——Cost Analysis

Self-built database

- Hardware cost
- Management Cost
- Human Cost
- Opportunity Cost

- Monitor
- Backup
- Middleware
- Low utilization
- DBAs
- DBAs
- Hinder innovation

Save cost 30%

Use RDS

- Hardware Cost
- Management Cost
- Human Cost
- Opportunity Cost

- Support OpenAPI
- Reduce work by 70%
- Nothing
- Focus on business

Buy on demand
Run right now

- MySQL for Cloud——Cost Analysis
- PERCONA LIVE
RDS for MySQL —— Enterprise Safety

- **Basic DB**
  - MySQL 5.7
  - Cost-effective

- **High-available DB**
  - MySQL 5.5/5.6/5.7
  - Continuity

- **Advanced Edition**
  - MySQL 5.6
  - Greatest stability
New scenarios are emerging, and new requirements are also raised.

We must ensure that data cannot be lost or confused any time.

So we developed such a new MySQL database product, RDS Advanced Edition for MySQL based on RAFT.

Features & Scenarios for RDS

- **Basic Edition** (Single-node)
  - Features
    - Simple, but most cost-effective
  - Scenario
    - Tutorials and training
    - Micro website
    - For developing and testing

- **Advanced Edition** (3-node)
  - Features
    - One master and 2 slaves with sync-replication
    - Most available and stable
  - Scenario
    - Core databases for customer which require higher data security, such as finance, securities, insurance and soon.

- **High-available Edition** (2-node)
  - Features
    - Classic architecture
    - High-available
    - Balanced in all aspects
  - Scenario
    - Database for enterprise
    - Database for internet, IOT
    - More than 80% scene
Architecture of RDS Advanced Edition for MySQL
MySQL Raft Architecture
MySQL Raft Architecture
MySQL Raft Architecture
MySQL Raft Architecture

Master

Slave-1

Slave-2

channel

RAFT
MySQL Raft Architecture

Failover module

RAFT module

Transaction module

Binlog module

Follower

Leader

Follower

Follower

REPL channel

RAFT channel

REPL channel

RAFT channel
About me

- Name: Diancheng Wang
- Location: Beijing, China
- Occupation:
  - Staff Database Engineer @ Alibaba Cloud
  - Focus on MySQL kernel
Review of RAFT Algorithm
RAFT basic

- Each server can be in one of three states
  - Leader
  - Follower
  - Candidate (to be the new leader)

- Followers are passive:
  - Simply reply to requests coming from their leader
RAFT states

- **Follower**
  - Starts up
  - Times out, starts election
  - Discovers current leader or new term

- **Candidate**
  - Times out, new election
  - Discovers server with higher term

- **Leader**
  - Receives votes from majority of servers
RAFT term

term 1  term 2  t3  term 4

election  normal operation  no emerging leader

terms
Log replication

- Leaders
  - Accept client commands
  - Append them to their log (new entry)
  - Issue AppendEntry RPCs in parallel to all followers
  - Apply the entry to their state machine once it has been safely replicated
    - Entry is then committed
Log entry organization

Colors identify terms
Election restriction

- The log of any new leader **must** contain all previously committed entries
  - Candidates include in their `RequestVote` RPCs information about the state of their log
    - Details in the paper
  - Before voting for a candidate, servers check that the log of the candidate is at least as up to date as their own log.
    - Majority rule does the rest
Detailed implementation for MySQL-RAFT
Overview of MySQL-Raft implementation

• Each node creates replication channels to others with Semi-Sync enabled and system variable settings:

  • `rpl_semi_sync_master_timeout = -1`
  
  • `rpl_semi_sync_master_wait_for_slave_count = floor(nodes / 2)`

• Detect failure by Raft heartbeat message

• Elect leader node using Raft protocol when failure occurs
Extra election restriction in MySQL-Raft (I)

- Vote by comparison of variable gtid_executed
  - Vote it iff candidate’s GTID set include its own
- No data will be lost if leader crashes because new leader must be the one synchronized with old leader
Extra election restriction in MySQL-Raft (II)

- Prerequisite of voting
  - `Set super_read_only to be TRUE`
  - `All relaylogs are applied`
  - `IO thread is stopped`
  - `SQL thread is running`
Processing unsynced transactions

- Unsynchronized transaction cases
  - Flushed to binlog file but not transferred to followers yet
  - Only transfer to minority
  - These transactions will be flashed back on other nodes if the leader doesn't include unsynchronized transactions
To process user threads waiting acks in SemiSync on leader when election occurs, Failover thread do following steps:

- Set flag in SemiSync to indicate the leader is stepping down
- Wake up user threads

User threads check the stepping down flag

- Close connection to client directly
- Continue to commit transaction (not wait slaves' ack any more)
- Flashback the transactions if other new leader is elected
What’s Flashback

• Rolling back a MySQL/MariaDB instance, database or table to a previous snapshot.

• By full image row format binary logs.
  • `binlog_format = ROW`
  • `binlog_row_image = FULL`

• Implement on Server-Level, so it supports all engines.

• It’s a feature inside mysqlbinlog tool (with `--flashback` option).

• Developed by Lixun Peng @ Alibaba Cloud, Already Contributed to MySQL and MariaDB
Binlog and Raft log (I)
Binlog and Raft log (II)
Leadership transfer

- Can only operate on leader
- Set `super_read_only` to `TRUE` at beginning of leadership transferring
- Trigger leadership transfer operation
  - The prior leader send `TimeoutNow` request to target server
  - The target server starts a new election
- The prior leader sets back `super_read_only` to `FALSE` if leadership transfer does not complete after about an election timeout
QA?
Thanks