Our Journey to Better MySQL Availability Using Global Transaction IDs, ProxySQL and Consul

Speakers: Stephane Combaudon, Yurii Olshanetskyi
AGENDA

About the company
Historical architecture (~ 2015)
  ● Overview
  ● Limitations
First set of improvements (2016 - 2017)
  ● MHA, MySQL 5.7, GTID
Current architecture (2018 - 2019)
  ● ProxySQL, Consul
  ● In-house HA software
Future plans
  ● What will keep us busy in the coming months
Q&A
The largest, most representative e-commerce panels in the world, precisely measures what others have only been able to approximate, revealing new insights about online consumer trends, retailers, brands and logistics companies.

Rakuten Intelligence is relied upon by more than 200 brands, retailers, and investment professionals to understand the forces driving the digital economy.

Our data influences critical decisions everyday:

- Competitive visibility
- Buyer behavior
- Loyalty and attribution
- Expand local markets
- Optimize logistics
- Deal sourcing
- Investing
AGENDA

About the company

> Historical architecture (~ 2015)
  • Overview
  • Limitations

First set of improvements (2016 - 2017)
  • MHA, MySQL 5.7, GTID

Current architecture (2018 - 2019)
  • ProxySQL, Consul
  • In-house HA software

Future plans
  • What will keep us busy in the coming months

Q&A
2015

HISTORICAL ARCHITECTURE

- All servers hosted in AWS
- We had 5 DB clusters
- Datasets ranging from 20 to 1000GB
- Each DB server had its own DNS name
HISTORICAL ARCHITECTURE

This architecture suffered from many problems:

- Problem with upgrade MySQL version
  IaaS didn't support any newer version.

- Bring up new db server took days
  Logical snapshot had long time restore process “data by request”

- No emergency failover
  IaaS provider didn’t support it.

- The cost was too much
  Extra 20% of aws costs.
AGENDA

About the company
Historical architecture (~ 2015)
  ● Overview
  ● Limitations

> First set of improvements (2016 - 2017)
  ● MHA, MySQL 5.7, GTID

Current architecture (2018 - 2019)
  ● ProxySQL, Consul
  ● In-house HA software

Future plans
  ● What will keep us busy in the coming months

Q&A
Regular position-based replication

The same transaction gets different coordinates on each server.

- INSERT INTO ...
  - Binlog file: 123
  - Binlog position: 200124

- INSERT INTO ...
  - Binlog file: 126
  - Binlog position: 56378

- INSERT INTO ...
  - Binlog file: 217
  - Binlog position: 59185342
What about switching replication master?

- For each slave, need to find the binlog coordinates of the new master.
- Not difficult, but
  - Error-prone, because `SHOW SLAVE STATUS` is confusing.
  - What if replication is lagging? (Solved in 5.7).
  - What if we have chained replication?
- All in all, it’s difficult to do it reliably.
GTID-based replication

A single GTID is generated on the master and shared by all servers.

INSERT INTO ...
Binlog file: 123
Binlog position: 200124
GTID: xxx-6875633

INSERT INTO ...
Binlog file: 126
Binlog position: 56378
GTID: xxx-6875633

INSERT INTO ...
Binlog file: 217
Binlog position: 59185342
GTID: xxx-6875633
What about switching replication master?

- For each slave, just give the IP of the new master.
- The replication protocol will automatically take care of finding the binlog coordinates from where to start replication.
- Easy to automate!
GTID REPLICATION - RECAP

Pros

- Switching masters is just a matter of switching IPs.
- Master and slave automatically negotiate transactions to be sent over.
- Replication topology changes are trivial.

Cons

- Some operations are more complex (e.g., skipping a transaction).
- \texttt{START SLAVE} can be slow.
- GTID-replication protocol provides new ways to break replication.
- Need to adjust your backup/restore tools for GTID compatibility.
Train of thoughts

- Most critical problem: no HA solution
- Switching to a new master is difficult, except with GTIDs.
- Therefore we need to use GTIDs.
- But GTIDs are a MySQL 5.6+ feature.
- And our IAAS provider only supports MySQL 5.5.
- So what’s the plan?
Action plan

- Create our own DB provisioning automation.
  - Saving on infrastructure cost.
  - Implementing a better restore solution with XtraBackup.
- Migrate to MySQL 5.7.
- Migrate to GTIDs.
- But this takes time, we need a temporary solution for MySQL switchover.
  - Let’s use an existing MySQL HA solution.
MHA (Master High Availability)

- Open-source, created by Yoshinori Matsunobu (Facebook)
- Supports many replication topologies.
- Supports switchover and failover - aka planned and unplanned master switch.
FIRST SET OF IMPROVEMENTS

$ master_ha_switch --new_master_host=B

MHA (Master High Availability)

- Open-source, created by Yoshinori Matsunobu (Facebook)
- Supports many replication topologies.
- Supports switchover and failover - aka planned and unplanned master switch.
FIRST SET OF IMPROVEMENTS

2015

2016 - 2017

MHA (Master High Availability)

- Open-source, created by Yoshinori Matsunobu (Facebook)
- Supports many replication topologies.
- Supports switchover and failover - aka planned and unplanned master switch.

$ master_ha_switch --new_master_host=B
While improved, our architecture was not perfect:

- MHA doesn’t have good GTID support.
- MHA comes with too many features that we don’t use and that create false errors.
- Downtime is still needed when switching MySQL masters because a DNS switch is needed.
- MHA needs an accurate list of servers at all times.
AGENDA

About the company
Historical architecture (~ 2015)
  • Overview
  • Limitations
First set of improvements (2016 - 2017)
  • MHA, MySQL 5.7, GTID
Current architecture (2018 - 2019)
  • ProxySQL, Consul
  • In-house HA software
Future plans
  • What will keep us busy in the coming months
Q&A
CURRENT ARCHITECTURE

To get rid of this DNS dependency, we need 2 features:

- Automatic discovery of the DB endpoints.
- Load-balancer/proxy to easily stop traffic if needed and switch it to a new server.
Discovery service

Consul

"Consul provides an opinionated framework for service discovery and eliminates the guest-work and development effort. Clients simply register services and then perform discovery using a DNS or HTTP interface. Other systems require a home-rolled solution" - consul.io

MAIN FEATURES

- Service discovery through REST and DNS
- Simple registration using REST API
- Distributed KV store for configuration
- Provides extensive health checking
Proxy

- MySQL Router

ProxySQL

- Large user base
- Understands the MySQL protocol
- Many advanced features

HAPROXY
CURRENT ARCHITECTURE

- High availability
- Load balancing
- DB is abstracted from external factors
- Maintenance downtime decrease from 1 hour to ~10 seconds
- Current uptime: 99.998%

user.proxy.service.consul
What about MySQL master change?

- With GTID replication, reconfiguring replication is easy.
- But we also need to make sure ProxySQL knows the new master.
- And of course, we don’t want to send writes while replication is reconfigured (who is the master??)

- So we wrote a tool to automate all this.
- We call it Metronome.
CURRENT ARCHITECTURE

Application Layer

ProxySQL Layer

MySQL Layer

Human triggers
MySQL switchover

$ metronome switch
**CURRENT ARCHITECTURE**

**Application Layer**

**ProxySQL Layer**

ProxySQL blocks new incoming connections

**MySQL Layer**

Stop new connections

Master

Slave1

Slave2

$ metronome switch
CURRENT ARCHITECTURE

Application Layer

ProxySQL Layer

MySQL Layer

ProxySQL blocks new incoming connections

2015

2016 - 2017

2018 - 2019

$ metronome switch
CURRENT ARCHITECTURE

Application Layer

ProxySQL Layer

MySQL Layer

Grace period for running trx to finish

$ metronome switch
CURRENT ARCHITECTURE

Application Layer

ProxySQL Layer

MySQL Layer

Remaining connections are killed

Master

Slave1

Slave2

Metronome

$ metronome switch
MySQL replication is reconfigured

```bash
$ metronome switch
```
MySQL replication is reconfigured

$ metronome switch
CURRENT ARCHITECTURE

Application Layer

ProxySQL Layer

MySQL Layer

ProxySQL configuration is updated

$ metronome switch

Master (old)

Master (new)

Slave2

Update config
CURRENT ARCHITECTURE

Application Layer

ProxySQL Layer

MySQL Layer

We're back to business!

$ metronome switch

Metronome
AGENDA

About the company
Historical architecture (~ 2015)
  ● Overview
  ● Limitations

First set of improvements (2016 - 2017)
  ● MHA, MySQL 5.7, GTID

Current architecture (2018 - 2019)
  ● ProxySQL, Consul
  ● In-house HA software

Future plans
  ● What will keep us busy in the coming months

Q&A
FUTURE PLANS

MySQL 8.0

- Roles.
- Invisible indexes.
- Improved JSON support.
- Group replication.
- Clone plugin.
- (Many many more features...).
- Rollout in progress.
FUTURE PLANS

ProxySQL 2.0

- Currently using 1.4.
- A bug prevented us from upgrading.
- Will allow full SSL from application components to MySQL.
Group Replication

- Standard MySQL HA solution.
- Would replace our custom HA software.
- Would allow automatic MySQL failover.
- But is it stable enough for production?
- What companies have successfully deployed it?
FUTURE PLANS

2015

2016 - 2017

2018 - 2019

2019+

Sharding

Business objects

Running threads: 926, 37, 9

Users Data

Meta

Shard 1

Shard 2

... 

Shard N

1. Managing 1TB+ compressed tables is a major pain.
2. Restores take ages.
3. Decreases blast radius in the event of a failure.
Q&A

Thanks for attending!