Sharing your data with Spider

Max Mether
max@mariadb.com
@maxmether
Who is Max?
What is Sharding?

"A database shard is a horizontal partition of data in a database or search engine. Each individual partition is referred to as a shard or database shard"
Why Shard?

- The resources of one machine is not enough!
- Read scaling can be achieved through master-slave replication
  - Replication however only scales reads; every server still has to write every single change
- In order to achieve write scalability something else is needed
  - Sharding partitions the data into different “shards”
  - Shards can be stored on different servers
- The sharding algorithm can have a huge impact on performance
Disadvantages with Sharding

• Disadvantages with Sharding include:
  • Increased complexity of SQL
  • Management complexity
  • Multiple points of failure
  • Failover more complex
  • Backups more complex
  • Operational complexity added
When do we need Sharding?

- Large Datasets
  - I/O- and CPU-load is the bottleneck
  - Long execution times for queries
  - Effects creating indexes, statistics, maintenance of tables, ...
- When replication is not a solution
- When per instance partitioning does not help
Replication for Scaling

- All writes go to MASTER
- Reads can be scaled on slaves
Replication or Sharding?

• Master/Slave-Replication
  • Scaling for reads with a large number of connects or queries
  • Useful for scenarios with a heavy read ratio
  • Not the solution when you have long execution times for single queries and large data sets
  • Write load cannot be scaled
  • Each server needs to contain all data
Partitioning for Scaling?

MariaDB

P1

Write

Read

P2

Write

Read

P3

Write

Read
Partitioning Types

- **RANGE and RANGE COLUMNS Partitioning**

  ```sql
  PARTITION BY RANGE (store_id) (  
    PARTITION p0 VALUES LESS THAN (1000),  
    PARTITION p1 VALUES LESS THAN (2000),  
    PARTITION p2 VALUES LESS THAN (3000),  
    PARTITION p3 VALUES LESS THAN MAXVALUE);
  ```

- **LIST and LIST COLUMNS Partitioning**

  ```sql
  PARTITION BY LIST(store_id) (  
    PARTITION pNorth VALUES IN (3,5,6,9,17),  
    PARTITION pEast VALUES IN (1,2,10,11,19,20),  
    PARTITION pWest VALUES IN (4,12,13,14,18));
  ```

- **HASH Partitioning**

  ```sql
  PARTITION BY HASH(store_id)  
  PARTITIONS 4;
  ```

- **KEY and LINEAR KEY Partitioning**
Partitioning vs. Sharding

• Partitioning allows
  • Reducing the data set for queries, when an effective partitioning rule can be defined
  • Separating archive data and active data
  • Distribute I/O-Load on multiple Disks
• Resources of an instance need to be shared (CPU, RAM, Kernel-Process, ...)
• Locks are still per table
How to do Sharding

- Sharding is database partitioning across multiple instances
- Implementation of sharding using
  - Application logic
  - Connectors
  - Proxies: MySQL Proxy, MySQL Fabric, MariaDB MaxScale
  - Spider storage engine
Where can you shard?
Where can you shard?
Where can you shard?

Application

Proxy

DB-Server

Shard
Spider Storage Engine
Storage Engine Architecture

Application

CONNECTORS  C  JDBC  ODBC ...

MariaDB

REPLICATION
BinLog API  Parallel Slave  GTID  Multi-Source

INTERNALS
Connection Pool  SQL Interface  Optimiser  Parser  Cache/Buffer

PLUG-INS
Authentication  Auditing  HandlerSocket

STORAGE ENGINE API

Standard  NoSQL / Interoperability  Performance

XtraDB  InnoDB  Connect  Cassandra  TokuDB  Spider ...

OPERATING SYSTEM / FILE SYSTEM

© MariaDB Corporation Ab
Spider Storage Engine

• Developed by Kentoku Shiba

• Storage engine "partitions" tables across multiple database server instances

• Based on partitions with integrated sharding

• Virtual view on tables distributed across Instances

• Supports XA transactions

• Transactional storage engine

• Provides scale-out in combination with HA
  • Can also use other HA
Spider Architecture

Only the needed partitions stored on each node

No data stored for SPIDER table
Spider Internals

• When a Spider table is created it creates a link to
  the remote table
• The linked table can have any engine
• The linked table can use partitioning
• The remote server is not spider aware
• You can have multiple Spider nodes for the same
  underlying tables
General Concept for Spider Engine

• Application with connection to Spider proxy node

  • CREATE TABLE spider (... ) ENGINE=SPIDER ...
    • No data in Spider-Proxy

  • CREATE TABLE spider (... ) ENGINE=INNODB ...
    • Data in backend
Spider as a Federation
Sharding using Spider

Application

Spider

Backend 1

P1

T1

Backend 2

P1

T1

Backend 3

P3

© MariaDB Corporation Ab
Sharding with Spider and HA
Spider Availability

• Since Version 10.0.4 included in MariaDB
  • Spider 3.0
  • Spider 3.2.11 in MariaDB 10.0.14
• Spider with MySQL Server
  • [http://spiderformysql.com/download_spider.html](http://spiderformysql.com/download_spider.html)
  • INSTALL PLUGIN spider SONAME 'ha_spider.so';
Spider Installation

• Installation

```
mysql -u root -p < /usr/share/mysql/install_spider.sql
```

• Spider will be shown as active Storage Engine

```
SELECT engine, support, transactions, xa FROM information_schema.engines;
```

<table>
<thead>
<tr>
<th>engine</th>
<th>support</th>
<th>transactions</th>
<th>xa</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPIDER</td>
<td>YES</td>
<td>YES</td>
<td>YES</td>
</tr>
<tr>
<td>CSV</td>
<td>YES</td>
<td>NO</td>
<td>NO</td>
</tr>
</tbody>
</table>
Spider System Tables

• Spider creates tables in the system schema (**mysql**)

MariaDB> show tables like 'spider%';
+----------------------------------+
| Tables_in_mysql (spider%)        |
+----------------------------------+
| spider_link_failed_log           |
| spider_link_mon_servers          |
| spider_tables                    |
| spider_xa                        |
| spider_xa_failed_log             |
| spider_xa_member                 |
+----------------------------------+
6 rows in set (0.00 sec)
Spider Variables

• 93 Spider system variables will be added

```
MariaDB> show global variables like 'spider%';
```

• 4 Spider status values will be added

```
MariaDB> show global status like 'spider%';
```

• More Spider variables related to tables using CREATE TABLE
  • In MariaDB use COMMENT
  • In MySQL use CONNECTION
Spider UDFs

• Spider UDFs will be added
  • SPIDER_DIRECT_SQL
    • Execute SQL on backend server
  • SPIDER_BG_DIRECT_SQL
    • Execute background SQL statement on backend server
  • SPIDER_COPY_TABLES
  • SPIDER_FLUSH_TABLE_MON_CACHE
    • Reset Spider monitoring information
Simple Spider Example

• Table definition on Spider proxy node

```
CREATE TABLE spiderfederation(id INT NOT NULL, code VARCHAR(10), PRIMARY KEY(id))
ENGINE=SPIDER
COMMENT 'host "192.168.56.21", user "backend", password "backend", port "3306"';
```

• Table definition on backend nodes

```
CREATE TABLE spiderfederation(id INT NOT NULL, code VARCHAR(10), PRIMARY KEY(id))
ENGINE=INNODB;
```
Spider Example with Sharding

- Table definition on Spider proxy node

```sql
CREATE TABLE sharding(id INT NOT NULL, code VARCHAR(10),
  PRIMARY KEY(id))
ENGINE=SPIDER COMMENT='user "backend", password "backend", port "3306", table "sharding"
PARTITION BY RANGE(id)
(
  PARTITION p1 VALUES LESS THAN (100000) COMMENT 'host "192.168.56.21"',
  PARTITION p2 VALUES LESS THAN (200000) COMMENT 'host "192.168.56.22"',
  PARTITION p3 VALUES LESS THAN MAXVALUE COMMENT 'host "192.168.56.23"'
);
```
Spider Example with Sharding

- Table definition on backend nodes

```sql
CREATE TABLE sharding(
    id INT NOT NULL,
    code VARCHAR(10),
    PRIMARY KEY (id)
) ENGINE=INNODB;
```
Spider Example with Sharding

• Insert on proxy

MariaDB> insert into sharding values (90002,"shard1"),
(100100,"shard2"), (200050,"shard3");
Query OK, 3 rows affected (0.04 sec)
Records: 3  Duplicates: 0  Warnings: 0

• Shard 1

MariaDB> select * from sharding;
+------------+--------+
<table>
<thead>
<tr>
<th>id</th>
<th>code</th>
</tr>
</thead>
</table>
+------------+--------+
| 90002 | shard1 |
+------------+--------+
1 rows in set (0.00 sec)
Spider Example with Sharding

• Shard 2

```
MariaDB> select * from sharding;
+--------+--------+
| id     | code   |
+--------+--------+
| 100100 | shard2 |
+--------+--------+
1 rows in set (0.00 sec)
```

• Shard 3

```
MariaDB> select * from sharding;
+--------+--------+
| id     | code   |
+--------+--------+
| 200050 | shard3 |
+--------+--------+
```
No Automatic Rollback

MariaDB> begin;
Query OK, 0 rows affected (0.00 sec)

MariaDB> insert into sharding values (90003,"shard1");
Query OK, 1 row affected (0.01 sec)

MariaDB> insert into sharding values (100101,"shard2");
Query OK, 1 row affected (0.00 sec)

MariaDB> insert into sharding values (200051,"shard3");
ERROR 1429 (HY000): Unable to connect to foreign data source: 192.168.56.23
MariaDB> commit;
Query OK, 0 rows affected (0.01 sec)
Replicating From Spider

- Replication from Spider proxy to slave
- Spider proxy binary log includes the Transactions
- No direct writes on backend tables
Replicating from Backend

- Replication from backend with multi-source replication
- When federation setup and writes to backend are used

© MariaDB Corporation Ab
Clustering and High Availability

- Spider supports HA internally
  - Commit and rollback across all backends
  - Multiplexing to replicas using 2PC
  - Split-Brain-Resolution based on quorum
- You can also use other techniques for HA on the backend servers
  - Galera
  - Replication
  - DRBD
CREATE TABLE backend.sbtest
(
    id int(10) unsigned NOT NULL AUTO_INCREMENT,
    k int(10) unsigned NOT NULL DEFAULT '0',
    c char(120) NOT NULL DEFAULT '',
    pad char(60) NOT NULL DEFAULT '',
    PRIMARY KEY (id),
    KEY k (k))
ENGINE=spider COMMENT='wrapper "mysql", table "sbtest"
PARTITION BY KEY (id) 

PARTITION pt1 COMMENT = 'srv "backend1 backend2_rpl" mbk "2", mkd "2", msi "5054", link_status "0 0"',
PARTITION pt2 COMMENT = 'srv "backend2 backend1_rpl" mbk "2", mkd "2", msi "5054", link_status "0 0"');
Clustering and High Availability Example

CREATE SERVER mon
  FOREIGN DATA WRAPPER mysql
OPTIONS(
  HOST '192.168.0.201',
  DATABASE 'backend',
  USER 'skysql',
  PASSWORD 'skyvodka',
  PORT 5054
);

INSERT INTO `mysql`.`spider_link_mon_servers` VALUES
('%', '%', '%', 5054, 'mon', NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, 0, NULL, NULL);

SELECT spider_flush_table_mon_cache();
Spider and Performance

• Reading
  • Simple queries generally faster
  • Queries spanning all shards can be slower if conditions not pushed down
  • Joins and complex queries can be a lot slower
    • Performance optimizations available through spider functions and options

• Writing
  • INSERTS Generally faster as each node is independent
  • UPDATES depend on reads to get to rows so depends
Spider Features

- Complete list on 

- Performance
  - Index condition pushdown (MariaDB 10)
  - Engine condition pushdown for federated setup
  - Engine condition pushdown for shards setup (MariaDB 10)
  - Batched key access
  - Support for handler socket
  - Map reduced for ORDER BY ... LIMIT
Good To Know

- DDL statements will not be synchronized
- Efficiency of sharding depends on the partitioning rule
  - Sub-Partitions can be used for the backend nodes
- Query cache needs to be deactivated
- Log files per Instance
- Central syslog makes sense for Audit Plugin
- User privileges - Authentication Plugin?
- Spider storage engine is BETA
More Information

• https://mariadb.com/kb/en/mariadb/documentation/storage-engines/spider/
• https://mariadb.org
• http://spiderformysql.com/
• http://bazaar.launchpad.net/~kentokushiba/spiderformysql/spider-2.0-doc/files/head:/en/
Questions?

Max Mether
max@mariadb.com
@maxmether