Optimizing DBMS Communication: Apache Thrift RPC vs. MySQL RPC

Jupyung (JP) Lee
Meta Platforms

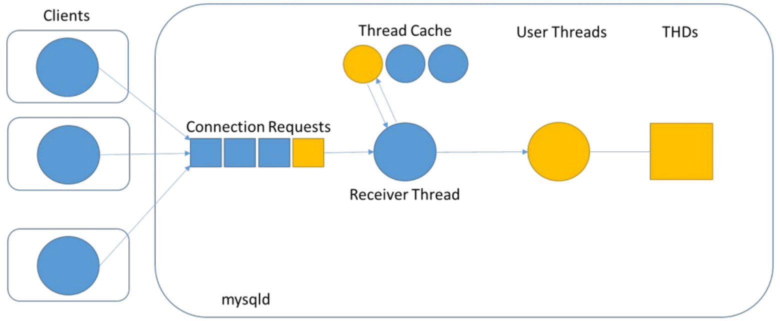
Introduction

- Choosing the right DBMS communication protocol and tuning well are crucial for the performance and CPU efficiency
    - Threading model
    - Opening/closing connections
    - Sending/parsing queries
    - Receiving/converting result sets
- Changing DBMS communication protocol, especially for DBMS serving production traffic, is challenging but can lead to huge reliability/CPU wins
- In this talk, Meta's journey of replacing MySQL RPC with Apache Thrift RPC will be introduced.

MySQL RPC vs. Apache Thrift RPC

MySQL RPC

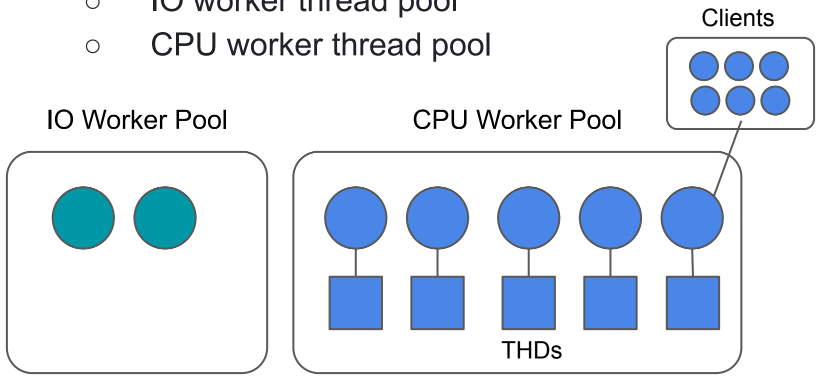
- MySQL server's own communication protocol
- Consist of connection phase and command phase
- When a user *connects* to the database, a *user thread* is created inside mysqld, running queries for the user until the user disconnects.
- There is always a one-to-one correspondence between a user connection and a thread (THD)



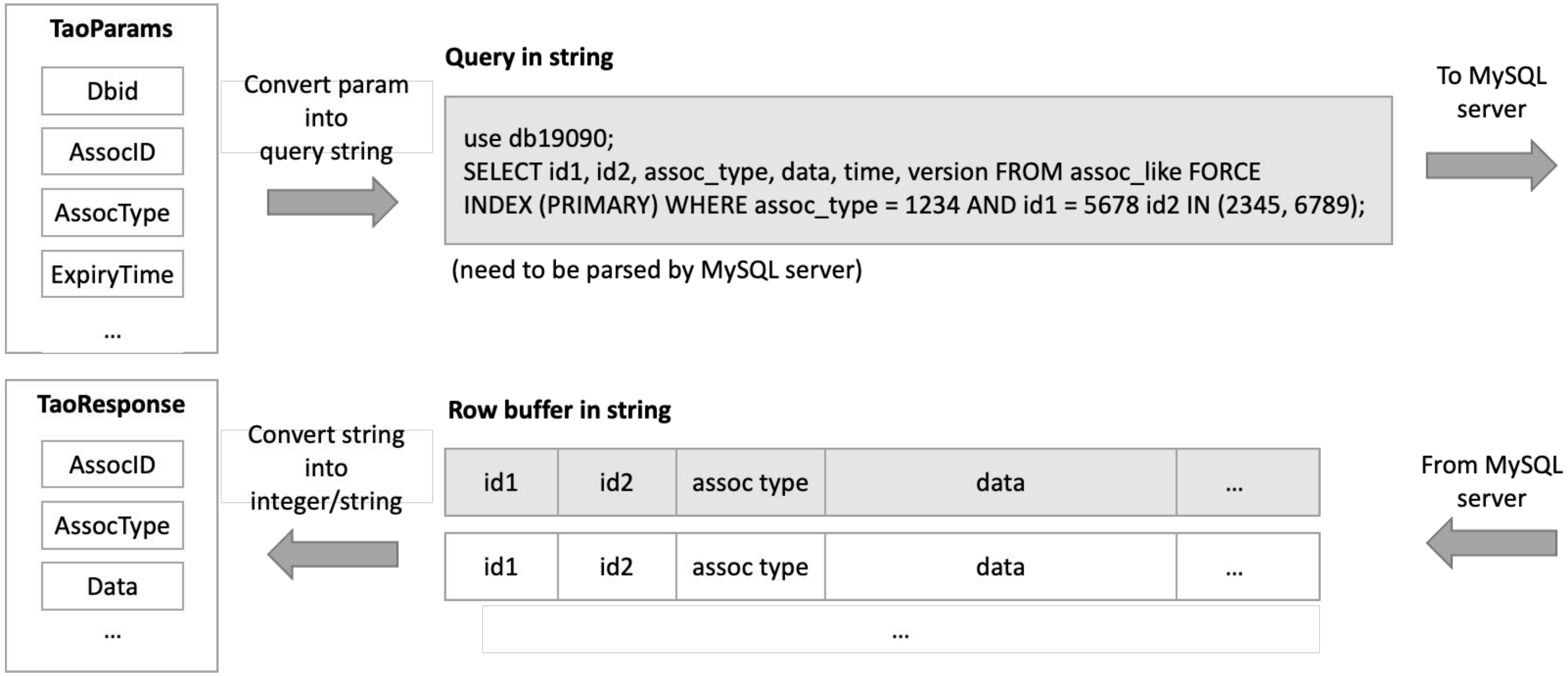
Source: https://dev.mysql.com/

Thrift RPC

- Lightweight, language-independent software stack for point-to-point RPC implementation
- Provide abstractions and implementations for data transport/serialization and application level processing
- Requests/Responses are schematized
- ThreadManager threading model (default one)
  - IO worker thread pool
  - CPU worker thread pool



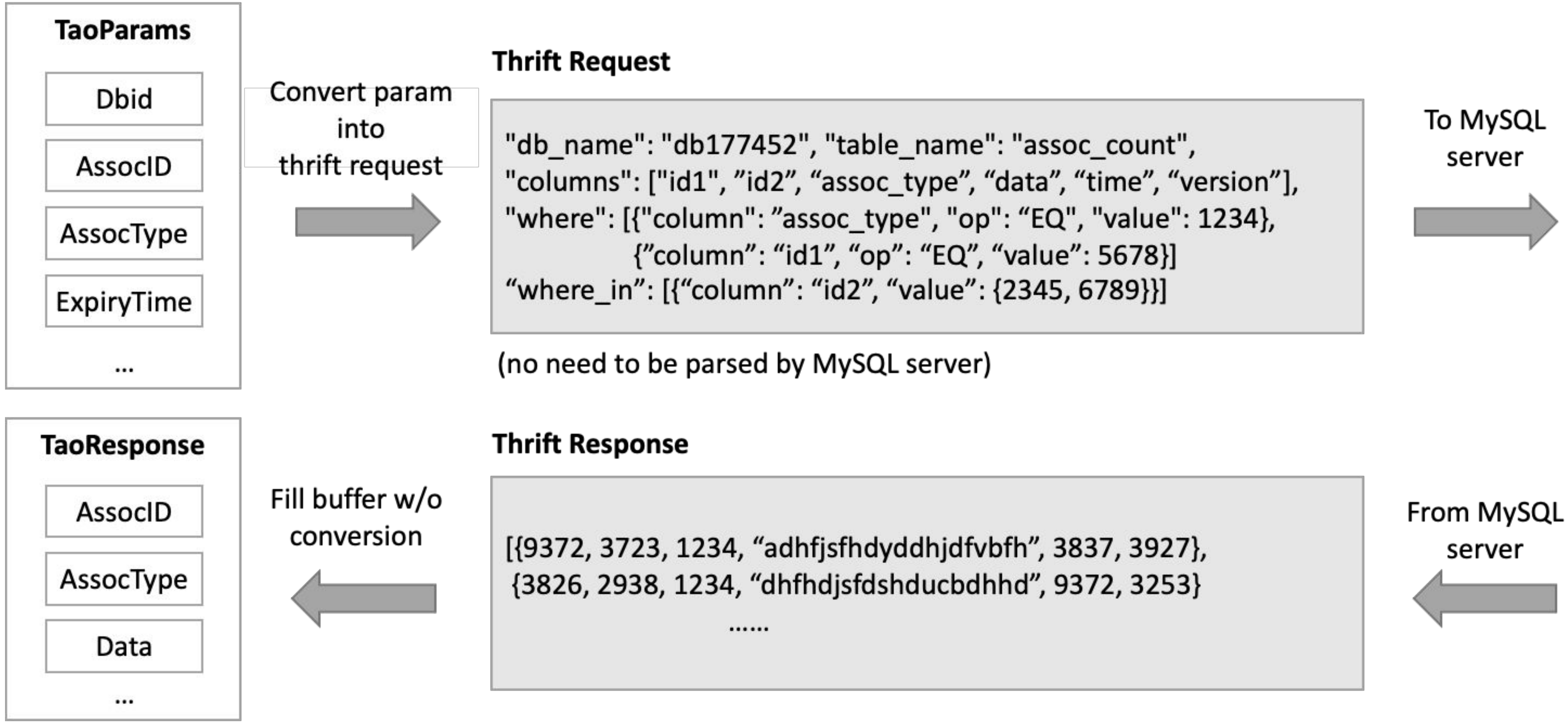
Example: MySQL RPC



```
use db19090;
SELECT id1, id2, assoc_type, data, time, version FROM assoc_like FORCE
INDEX (PRIMARY) WHERE assoc_type = 1234 AND id1 = 5678 id2 IN (2345, 6789);
```

id1	id2	assoc type	data	...
id1	id2	assoc type	data	...
		...		

Example: Thrift RPC



```
"db_name": "db177452", "table_name": "assoc_count",
"columns": ["id1", ”id2”, “assoc_type”, “data”, “time”, “version”],
"where": [{"column": ”assoc_type", "op": “EQ", "value": 1234},
          {”column”: “id1”, “op”: “EQ”, “value”: 5678}]
“where_in”: [{“column”: “id2”, “value”: {2345, 6789}}]
```

```
[{9372, 3723, 1234, “adhfjsfhdyddhjdfvbfh”, 3837, 3927},
 {3826, 2938, 1234, “dhfhdjsfdshducbdhhd”, 9372, 3253}
                 ......
```

Example: Thrift IDL (Interface Description Language)

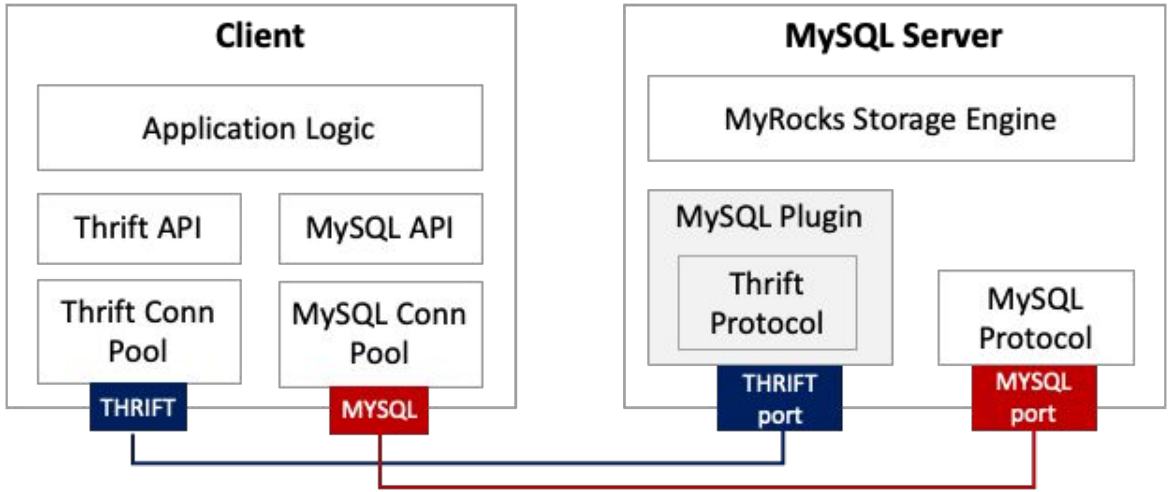
```
struct MySQLRequest {
 1: string db_name;
 2: string table_name;
 3: list<string> columns;
 4: list<WhereItem> where;
 5: list<WhereInItem> where_in;
 6: optional i64 limit;
 7: optional string force_index;
 8: optional list<OrderByItem> order_by;
 9: optional i64 limit_offset;
}
```

```
union ColumnValue {
 1: bool isNull;
 2: bool boolVal;
 3: i64 unsignedIntVal;
 4: i64 signedIntVal;
 5: double doubleVal;
 6: string stringVal;
}

struct Header {
 1: ColumnType type;
 2: string name;
}

struct QueryResultRows {
 1: list<Header> header;
 2: list<list<ColumnValue>> rows;
}
```

This talk will deal with only stateless select queries over Thrift protocol.

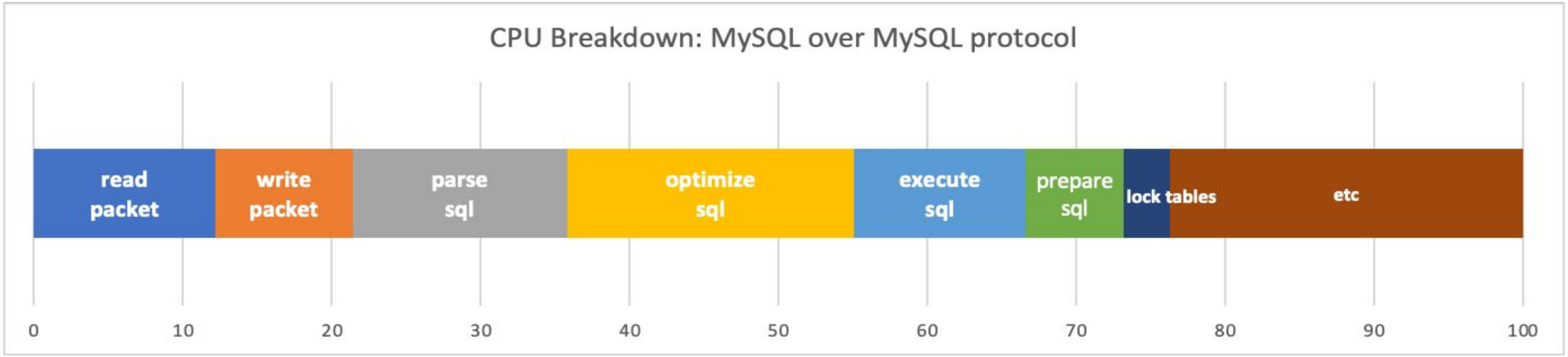
Adding Thrift RPC to MySQL Server

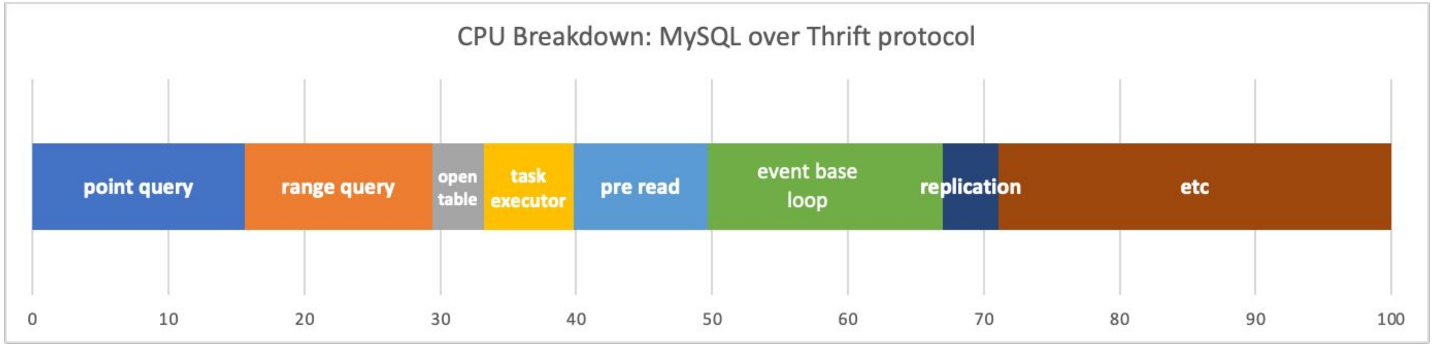
- Thrift server can be built as MySQL plugin module, which can be independently installed/uninstalled
- Defining thrift port equally distant from mysql port makes it easy to reuse existing database discovery service for discovering thrift server



CPU Breakdown: MySQL Protocol vs Thrift Protocol

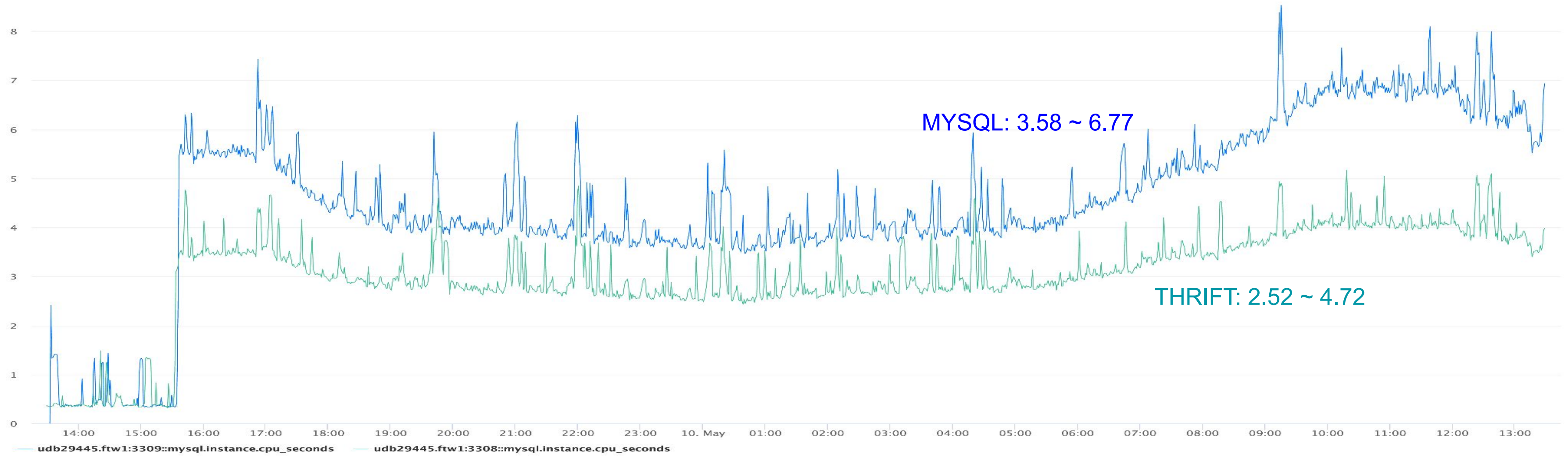
Used production select queries against social graph data (~25k QPS)





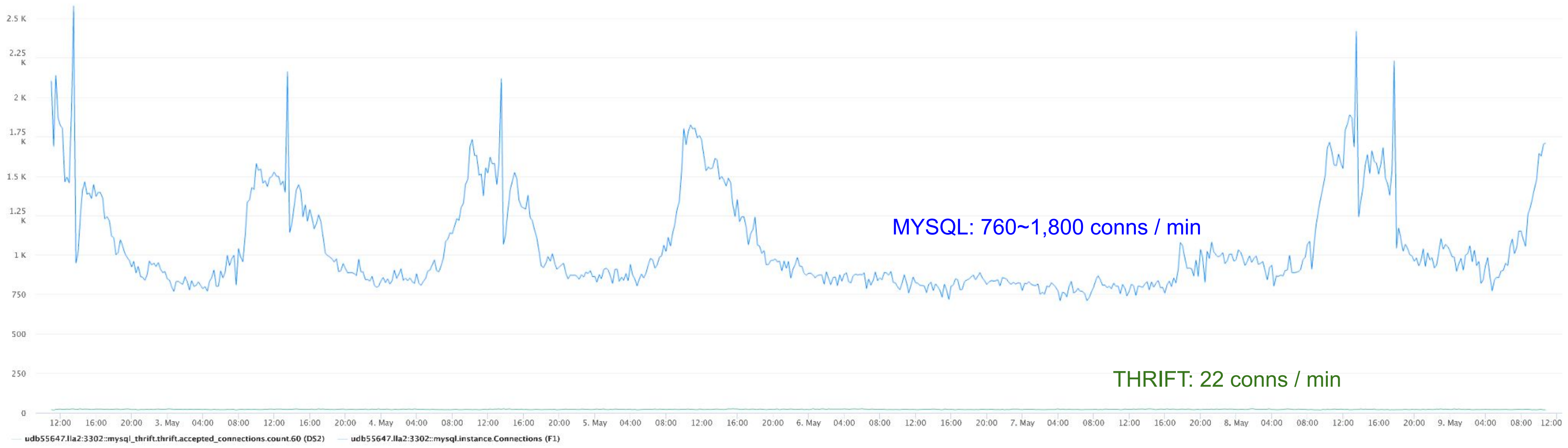
CPU Utilization: MySQL Protocol vs Thrift Protocol

MySQL server with Thrift protocol saved CPU by 20~30% on average



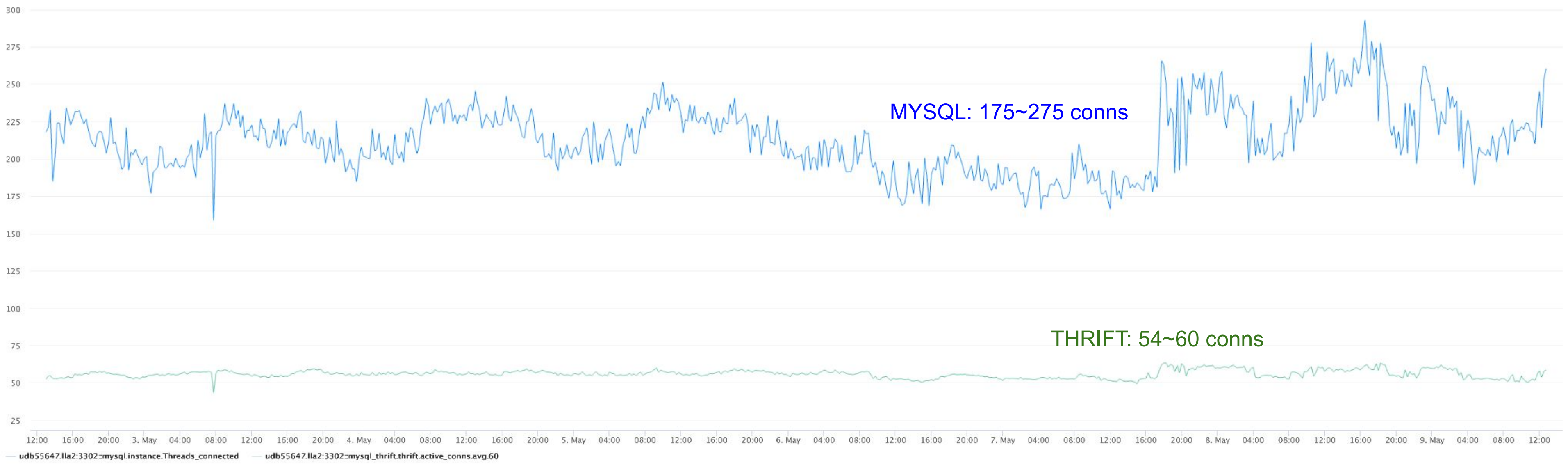
Connection/Min: MySQL Protocol vs Thrift Protocol

Thrift connection is much better sharable/reusable than MySQL connection, because it's not tied to single user session and supports multiplexing



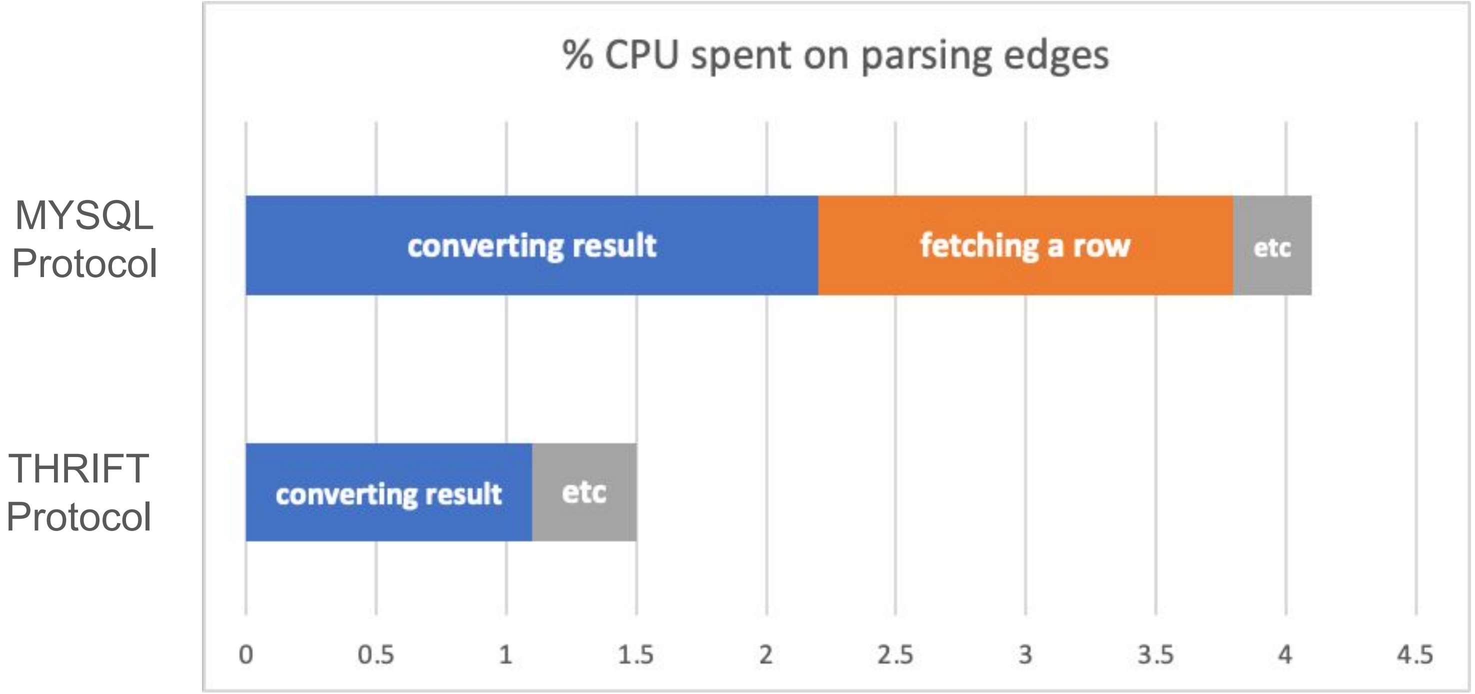
Num Active Connections: MySQL vs Thrift Protocol

Thrift protocol keeps 1/4 of connections MySQL protocol keeps



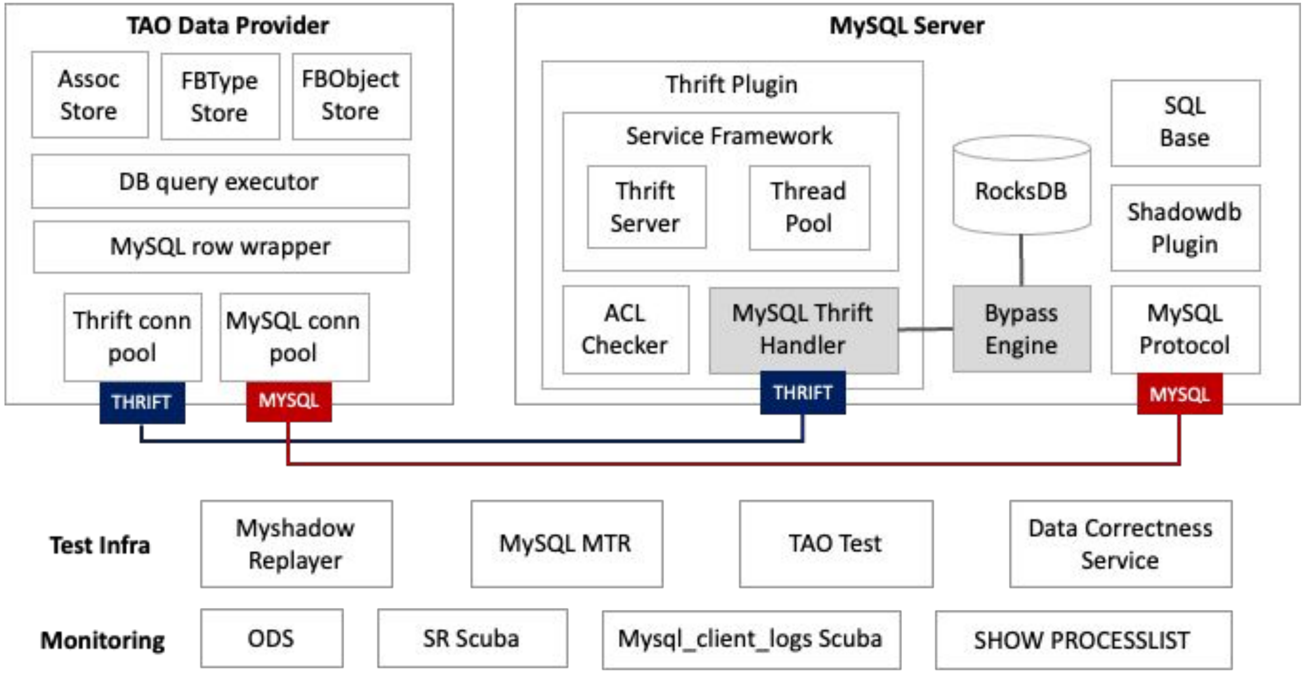
Parsing Result Sets: MySQL vs Thrift Protocol

Clients (TAO in this example) spends less CPU parsing result sets with Thrift



Overall Diagram: MySQL over Thrift Protocol in Practice

Investing heavily in client library (clean APIs with dual protocol support) and test infra (running correctness test against MySQL and Thrift protocol) is critical to the success of the project.

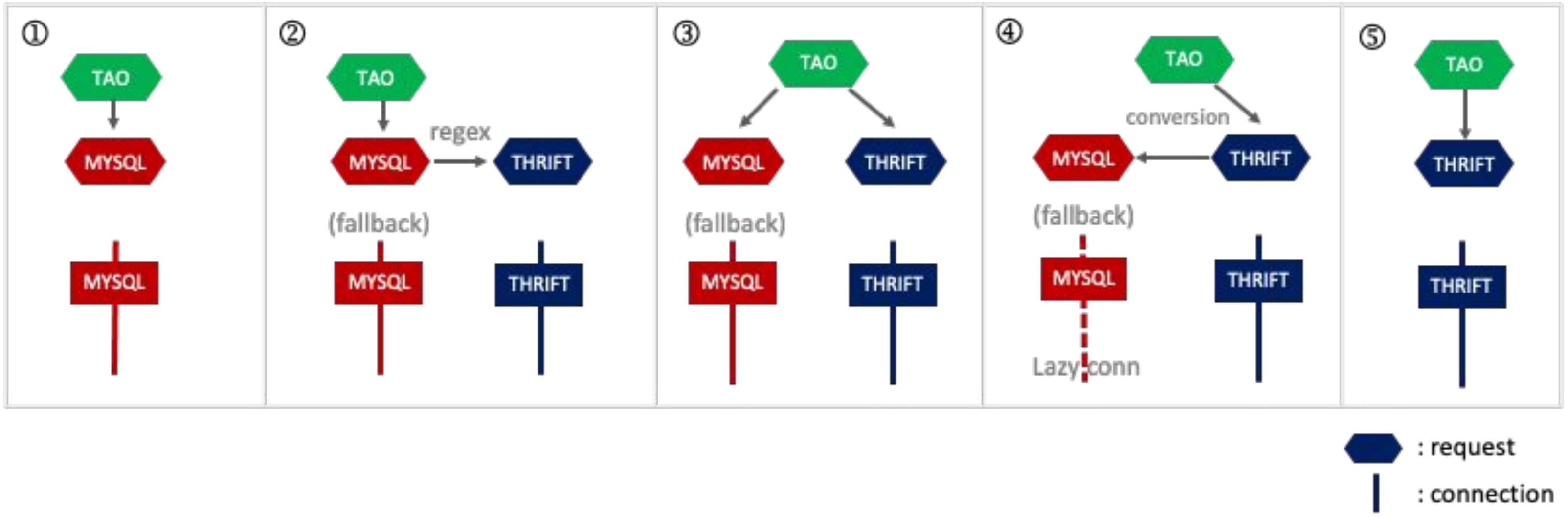


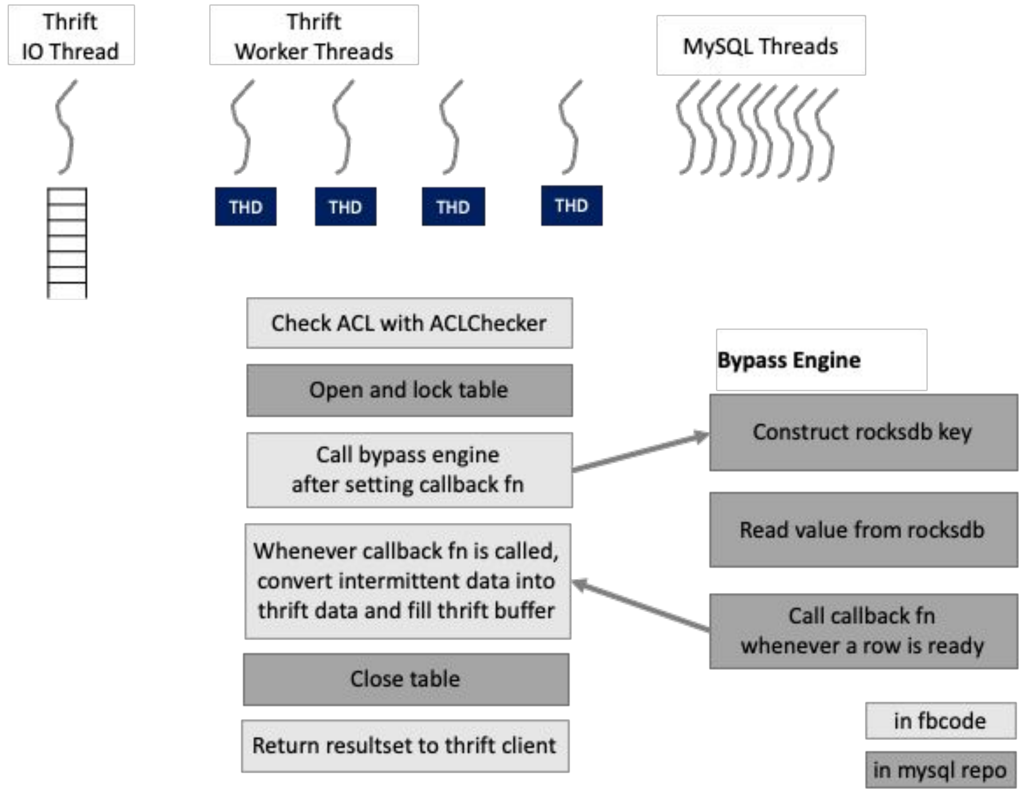
Conclusion

- Thrift RPC offers many advantages over MySQL RPC for our workload
  - Reducing 20~30% server CPU utilization
  - Drastically reducing connection rate, thanks to its multiplexing capability
  - Reducing client CPU spent on parsing results and opening/closing connections
- Adding a new RPC protocol in production DBMS requires careful rollout/rollback plan
- Support for new client library (clean APIs with dual protocol support, conn pool for both protocols) and test infrastructure is essential

Appendix

Rollout and Rollback Strategy




Thrift IO Thread
Thrift Worker Threads
MySQL Threads
THD
THD
THD
THD
Check ACL with ACLChecker
Open and lock table
Call bypass engine after setting callback fn
Whenever callback fn is called, convert intermittent data into thrift data and fill thrift buffer
Close table
Return resultset to thrift client
Bypass Engine
Construct rocksdb key
Read value from rocksdb
Call callback fn whenever a row is ready
in fbcode
in mysql repo