Evolution of a Data Warehouse Engine's Query Protocol

liyazhou@databend.com DatabendLabs

Databend

Agenda

01 Databend Introduction

02 Early Days of Databend's Query Protocols

03 The Road to a Modern Query Protocol

04 Ecosystem Integration

05 Future Plans

06 Question & Answers

Databend

What is Databend?

Databend is an open-source cloud data warehouse engine that serves as a cost-effective alternative to Snowflake. It features:

- High performance vectorized execution engine built in Rust.
- Fully separation of storage and compute for independent scaling.
- Object storage as the primary storage.
- ANSI SQL compliant, with semi–structured data support.
- High–throughput real–time data ingestion.

Databend

Databend Use Cases

Databend has been deployed in many mission-critical enterprise scenarios, successfully helped these customers reducing costs and improving efficiency:

Batch Analytics

Replace Hive and Spark workloads for efficient large–scale data processing

Real–time Analytics

Handle streaming data processing and power real–time recommendation systems

Log Analytics

Replace ElasticSearch for centralized log storage and analysis

Database Archival

Archive MySQL/TiDB historical data.

User Behavior Analysis

Track, analyze, and derive insights from user interactions and activities.

Databend

Databend Use Cases

When implementing these scenarios, ecosystem support is inevitably needed.

In the Databend ecosystem, rich support is provided for ingestion, ETL, BI and many other ecosystems:

Ingestion: Load data from Kafka, MySQL/Postgres, etc in real-time.

ETL: Transform data with dbt, Airflow, Airbyte, etc.

BI: Visualized with Metabase, Redash, Grafana, etc.

SDK: Programmatic access with JDBC, Python, Rust, Go, etc.

CLI: Interactive command line tool "`bendsql`" which supports syntax highlighting, autocompletion, etc.

All these integrations are built on top of Databend's **Query Protocol.**

Databend

Early Days of Databend's Query Protocols

In its early days, Databend intended not to introduce a dedicated query protocol.

To accelerate time to market, Databend uses the **MySQL protocol & Clickhouse HTTP protocol** for a faster development & faster user adoption.

Databend

MySQL Protocol

- MySQL protocol is widely adopted, and “mysql-client” is familiar to lots of programmers.

- In the early days, many users are actively using “mysql-client” to connect to Databend. It helps a lot on adapting early users.

```
Copyright (c) 2000, 2022, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> show tables;
+-----------------+
| Tables_in_world |
+-----------------+
| city            |
| country         |
| countrylanguage |
+-----------------+
3 rows in set (0.00 sec)

mysql>
```

Databend

MySQL Protocol's Issues

However, there are some issues:

> MySQL Protocol does not support returning progress information while executing a query.

> MySQL drivers, especially JDBC, often depend on some unusual system tables and fields, such as some complex joins over “ information_schema ” , some of them are hard to remake in Databend.

> Databend's SQL syntax choose to be ANSI SQL oriented, which is more similar to PostgreSQL, causing many users to question why something like `my_table` don't work.

Clickhouse HTTP Protocol

Clickhouse's HTTP protocol is pretty simple and easy to implement. And it's also familiar to our team members who were active in Clickhouse community.

```
$ curl 'http://localhost:8123/?query=SELECT%20numbers%283%29'
1
2
3
```

And it also returns a nice progress information while executing a query:

```
X-ClickHouse-Progress: {"read_rows":"2752512","read_bytes":"240570816","total_rows_to_read":"8880128","elapsed_ns":"662334"}
X-ClickHouse-Progress: {"read_rows":"5439488","read_bytes":"482285394","total_rows_to_read":"8880128","elapsed_ns":"992334"}
X-ClickHouse-Progress: {"read_rows":"8783786","read_bytes":"819092887","total_rows_to_read":"8880128","elapsed_ns":"1232334"}
```

Clickhouse HTTP Protocol

Databend

- Clickhouse's HTTP query is single long polling, and network jitter may cause it to fail.
- Many HTTP gateways like Nginx has a limit on the maximum timeout for HTTP long polling.

Databend

The Road to a Modern Query Protocol

After the Databend Cloud project started, we began to re-

- Should tolerate HTTP gateway restart and network jitter on long running queries.
- Should provide rich progress information during query execution.
- Should allow offloading data plane traffic to avoid high cross–AZ data transfer costs.
- Should be simple and easy to implement.

The RESTful Query Protocol

Databend

The RESTful Query Protocol is designed to meet the above requirements.

```
curl -u root: \
  --request POST \
  '127.0.0.1:8001/v1/query/' \
  --header 'Content-Type: application/json' \
  --data-raw '{"sql": "SELECT avg(number) FROM numbers(100000000)"}'
```

Databend

The RESTful Query Protocol


```
{
  "id":"b22c5bba-5e78-4e50-87b0-ec3855c757f5",
  "session_id":"5643627c-a900-43ac-978f-8c76026d9944",
  "schema":[
    { "name":"avg(number)", "type":"Nullable(Float64)" }
  ],
  "data":[ [ "49999999.5" ] ],
  "state":"Succeeded",
  "error":null,
  "stats":{
    "scan_progress":{ "rows":100000000, "bytes":800000000 },
    "write_progress":{ "rows":0, "bytes":0 },
    "result_progress":{ "rows":1, "bytes":9 },
    "total_scan":{ "rows":100000000, "bytes":800000000 },
    "running_time_ms":446.748083
  },
  "stats_uri":"/v1/query/b22c5bba-5e78-4e50-87b0-ec3855c757f5",
  "final_uri":"/v1/query/b22c5bba-5e78-4e50-87b0-ec3855c757f5/final",
  "next_uri":"/v1/query/b22c5bba-5e78-4e50-87b0-ec3855c757f5/final",
  "kill_uri":"/v1/query/b22c5bba-5e78-4e50-87b0-ec3855c757f5/kill"
}
```

Databend

The RESTful Query Protocol

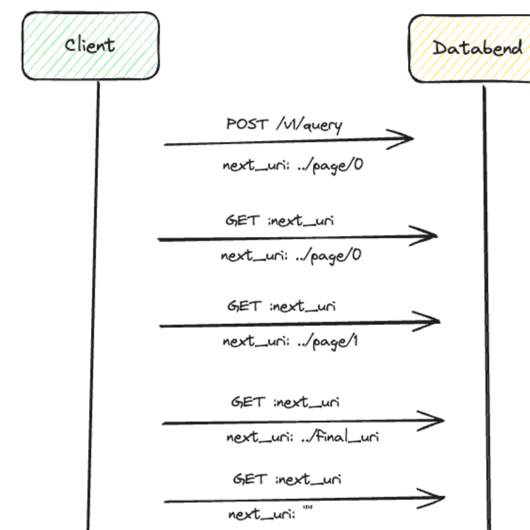
It's a pagination based protocol, and each page contains a partial of query results.

For every query, the client will get a “ next_uri “ to get the next page of results.

A “ next_uri “ can be safely retried.

When the query is still running, the “ next_uri “ may return an empty page after a while. And the client should keep retrying the “ next_uri “ until getting a non-empty page.

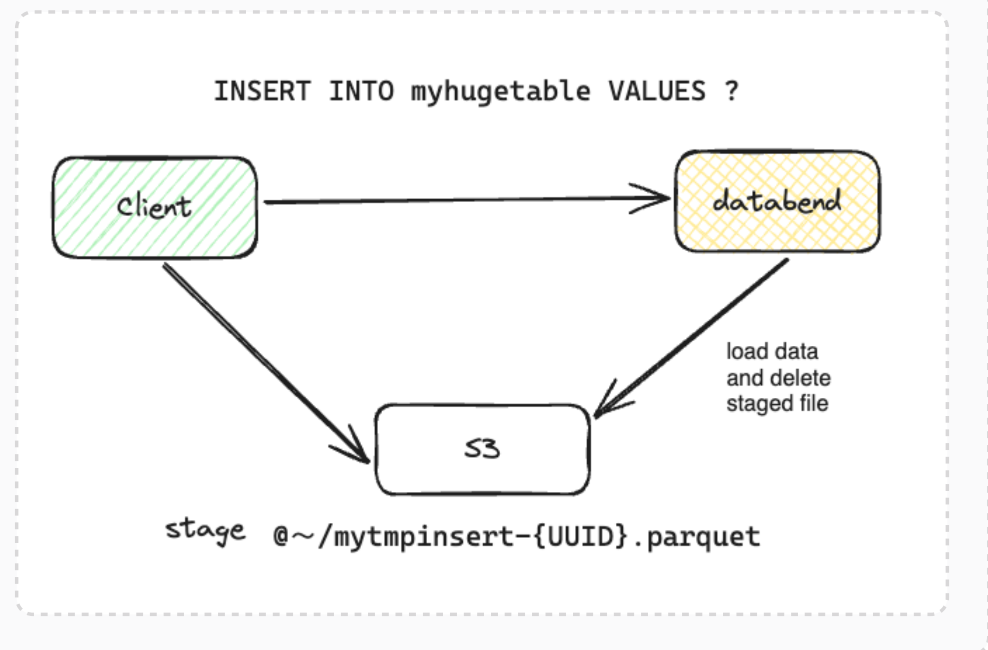
When the query is done, the “ next_uri “ will become the “ next_uri “ , and the client should call the “ next_uri “ to release the query resources in the server side.



Databend

Offloading Data Plane

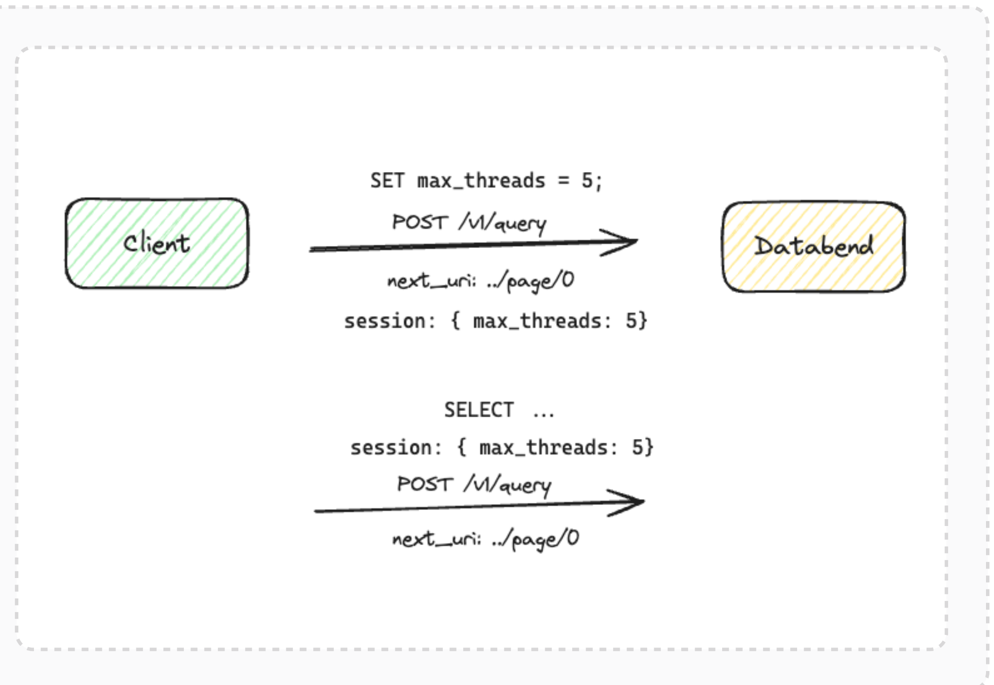
- We can not assume to co-locate in the same AZ in the cloud.
- To avoid high cross-AZ data transfer costs, we can offload the data plane to the S3 storage.



Databend

Save Session State

- At first, Databend's query protocol are mostly stateless.
- At some point, we need to support some session settings, like “max_threads” , “collation” , etc. Also, we need to track some metadata about transactions, etc.
- We still choose a RESTful style to save the session state.



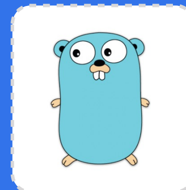
Databend

Developing Drivers

Before the ecosystem integrations, we have to build stable client drivers among different languages as the foundation.

In the driver, we need to handle these things properly:

- Correct pagination handling
- Correct data offload handling
- Correct session state management
- Correct ORM integration



Databend

Build the Ecosystem: CLI

The first step is build a CLI tool called “ `bendsql`” to replace `mysql-client` . In most of the time, CLI is the first interaction to users.

```
brew install databendcloud/homebrew-tap/bendsql
```

Databend

Build the Ecosystem: Ingestion, ETL, BI

JDBC is the underated. Many tools connects JDBC nicely, without the need to develop a new connector. Such as “dbeaver” , “metabase” , etc.

And the data ingestion ecosystem is also heavily based on JDBC, like Flink CDC and Kafka Connector are the two major ones.

To replicate the data from a MySQL/Postgres database, Debezium is the de facto standard, it's also based on JDBC.

The development cost of all these ecosystems integration is surprisingly not high as we expected, since the driver layer already handles the details of the query protocol nicely.

Databend

Future Plans

Better built-in load balancing capabilities

Reuse Rust core for every language's driver

Databend

Key Takeaways

- Starting with an existing protocol makes the early adopters easier to onboard.
- When you have to build your own protocol, it's better to keep it simple and easy to implement.
- On the Cloud, offloading data plane to S3 storage is the key.
- Having an Rust core can significantly reduce the work on developing a new language's driver.
- CLI is the first interation to users.
- JDBC is the underated.

Questions & Answers

Databend

Thank you！

Databend