Composable Gateway for Databases (Lightning Talk)

Aditya Jalan[1], Muthu Annamalai[1]

[1]*Meta Platforms, Inc., Bellevue, USA*

Abstract

Database offerings in production are generally served to clients via gateways(proxies). Instead of recreating them from scratch for each offering, a composable architecture allows for different gateways to share code, providing increased reliability, engineering efficiency and better business impact.

Keywords

Database Gateway, Database Proxy,

1. Introduction

At scale, database offerings are generally fronted by gateways(proxies), which acts as an intermediary between applications and one or more database instances. At Meta for example, we have multiple databases (relational, non-relational, graph, etc.) and each one is fronted by its own gateway service. If we take a step back and squint, it is observed that they all share common functionality. Some of the key functions of a gateway include:

- `Connection Management`: It provides connection pooling, abstracts and simplifies logic of establishing connections to database instances.
- `Authentication and Authorization`: Simplifies the responsibility of databases and ensures only valid and authorized clients can access them.
- `Tenant isolation and Load balancing`: These are one of the most critical pieces of a shared database offering, where gateway is responsible for providing safety to one application or database from another noisy neighbor.
- `Routing and Discoverability`: Gateway abstracts the layout and discoverability of backing physical DB instances from the client and provides routing via logical entities.
- `Caching`: The gateways can implement caching mechanisms to store frequently accessed data, reducing the load on the databases and improving response times for clients.
- `Monitoring and Analytics`: This is bread and butter, to give insights into the usage pattern for a database or application by providing a single choke point.

2. Composable Gateway

These functionalities, when implemented correctly in a generic modular way, can be reused and stacked together to provide a gateway for any database offering. Building a composable Gateway has major benefits in terms of developer productivity, reliability, availability of advanced features across all offerings, reduced maintenance, consistent experience for customers. Keeping these functions in mind, at Meta, we have been exploring building a gateway with pluggable components for our internal Database offerings, MySQL[1] and ZippyDB[2]. Currently both have service specific gateways and client stacks because of how each has different connection protocols, query experiences, and one being open source and other being home grown on native technology. To resolve this, as a first step, we are migrating MySQL to use thrift as the communication protocol end-to-end, bringing the first stage of parity for both the datastores. At the same time, we are teasing out these common functionalities some of which are stated above into pluggable modules, which can be combined together into a fully composable gateway to serve both the offerings. This architecture provides a robust, extensible and standardized solution for accessing databases, with easier maintenance and scalability at core.

References

[1] H. Lee, P. Nayak, Migrating facebook to mysql 8.0, 2021. URL: https://engineering.fb.com/2021/07/22/data-infrastructure/mysql/.

[2] S. Masti, How we built a general purpose key value store for facebook with zippydb, 2021. URL: https://engineering.fb.com/2021/08/06/core-data/zippydb/.

Joint Workshops at 49th International Conference on Very Large Data Bases (VLDBW'23) — Second International Workshop on Composable Data Management Systems (CDMS'23), August 28 - September 1, 2023, Vancouver, Canada

aditj@meta.com (A. Jalan); muthu@meta.com (M. Annamalai)
0009-0007-1812-166X (A. Jalan)

CEUR Workshop Proceedings (CEUR-WS.org)