2021 Advanced Database Systems Final Project
Hybrid Distributed Database Service Design

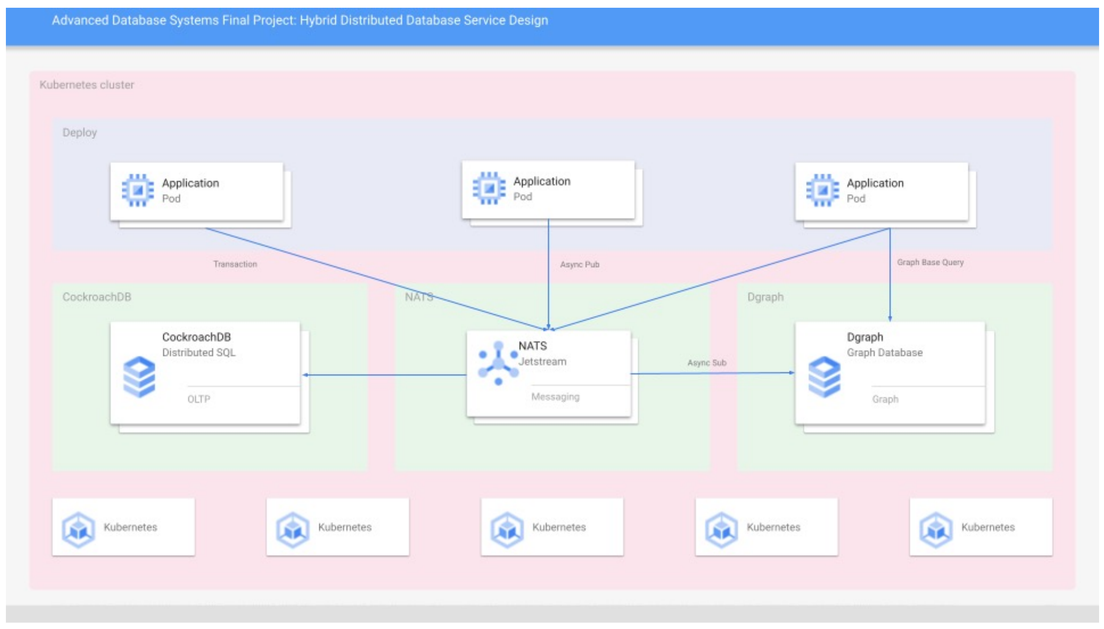
110065506 Ta-Chun Wu

I. Introduction

In this project, I build a distributed system that integrates 2 services. One of the services can process GraphQL (a graph base protocol that can query data purpose by Facebook) and also can process Full-Text search and Geo search, the other service can process online transactions.

Furthermore, the whole system is designed with high availability, high throughput, micro-service features.

II. Architecture



I. Kubernetes:

The State-of-the-Art of container orchestration, which supports self-healing, horizontal scaling, service discovery, load balancing, automated rollouts, and rollbacks. Kubernetes builds upon 15 years of experience of running production workloads at Google, combined with best-of-breed ideas and practices from the community.

II. CockraochDB

CockraochDB is a distributed SQL database designed for speed, scale, and survival. Which is based on Google Spanner but does not need GPS and True-Time Atomic Clock. This component handles the system ACID transactions, also supports the linear scale on SQL.

III. Dgraph

Dgraph implements a GraphQL backend on the database, which supports graph-based query, geo query, full-text query, HA, snapshot isolation level transactions. This component handles the complex search query in this system.

IV. NATS

The fastest messaging application provided by Cloud Native Foundation. This component helps other sub-systems and application servers to communicate with each other to achieve low latency and other issues in distributed systems.

III. Implementation Details

I. Infrastructue Management

Kubernetes uses YAML files to the config cluster states. This approach is always used to deploy the application server. But if you want to manage infrastructure I recommend using the KRM model, in other words, manage infrastructure application by using Operator and CRDs.

II. Distributed Messaging

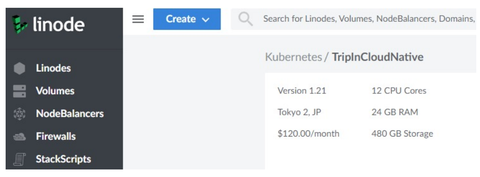
Infrastructure applications can provide function-level service in their domain. For example, CockraochDB supports the scalable ACID transaction, Dgraph supports the Full-text search and GIS feature. In traditional application design application servers direct request the API endpoints and operation what they need. But in the distributed world every component could crash anywhere at any time. So you need to use the messaging application to manage your workload synchronously or asynchronously.

For instance, I implemented a service that allows users to pay money in order to have a higher priority in the comment section. Because it is a transaction so must provide ACID level transaction to this service. But when you view the comment section, you can use the Full-Text search function which is provided by Dgraph, and Dgraph has lower TPS performance.

So when high concurrent requests visit the service, there will be a bottleneck in request Dgraph API. The better way to deal with this issue is using asynchronous requests. First, handle the payment with ACID transactions, then asynchronously update the comment to Dgraph. There are more useful features when you apply the messaging method to a large-scale distributed system this is just one example.

IV. Demo Snapshot

I. Infrastructure



II. GraphQL

III. Full-Text

IV. GIS

V. Messaging

VI. Transaction