LogStore: Architecting Log Storage System for Cloud-Native Database

*

1st Fanzhong Wang
11911824

2nd Yifan Zhang
1191

3rd Jialin Li
11812701

Abstract—With the prevalence of cloud computing, more and more companies move their applications to cloud infrastructures. Key advantages are highly scalability and availability with a lower cost. when enterprises use cloud-based services to run their applications, they are expected to work stably, efficiently and securely. But traditional database server are usually underutilized much of time. The key to achieve it is to architect a cost-effective log storage for cloud applications, which also can help users better understand the status of their applications running on the cloud. Traditional log processing systems cannot satisfy all these requirements.

Just as said before, there are some disadvantages of trandditional log systems. So we choose the cloud-native log database, LogStore. It combines shared-nothing and shared-data architecture, and utilizes highly scalable and low-cost cloud object storage, while overcoming the bandwidth limitations and high latency of using remote storage when writing a large number of logs.

I. Description

A. Log

Logs are critical in the applications. We are familiar with the logs when a bug occurs when running java programs. With such log we can debug more easily. However it's far from enough to analyze logs only at the beginning or end, or only on bugs appearing. Logs during a metho even more useful, which contains: time, level of the (warn, error, etc.), call chain id(optional), thread na class name, content of the log, and some logs of level w or error have exception stacks.

Logs are stored in a steady file named in some templ is writing or has rolled back. However, logs can addec to a very large size, so we must compress each piec log as a string before storing it.

B. Cloud native

With the evolution of cloud native, it has many vers of definition, including Pivotal, CNCF and the twe factor app. Raised by Heroku(2011) [1], the twelve-fa app can be applied to any language besides database, and is considered as the earliest technical features of cloud native applications, which says:

- Use standard processes for automatic configuration on interest of cost of study
- Seperate from os as much as possible for most portability in every systems
- Suit for deployment on morden cloud computing platform to save resources
- Decrease the difference between the environments of development and production and use continuous delivery
- Achieve extension without obvious modification of tools, architecture and development process

The cloud is defferent. Data is stored on dedicated local storage in traditional database. Usually two or three copies of the database are typically maintained. This architecture wastes resources and increases the cost of services in a cloud environment. Scaling out compute by adding read replicas is slow and expensive because a completely new copy of the database needs to be created for each replica added. Large, multi-terabyte databases cannot be supported because operations, such as backup and restore, simply take too long.

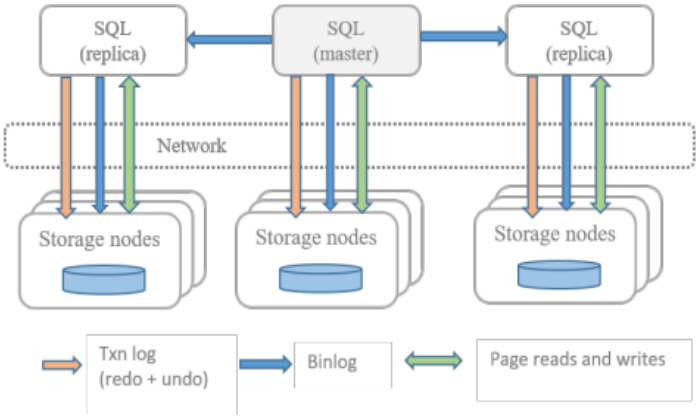

Fig. 1. MySQL with two replicas deployed in a cloud environment

C. Cloud-Native database

The cloud-native database has some main features [2]:

- the computation and storage system are seperated, and cumputation nodes are few or none in certain state.
- it is consistent based on the logs
- storage system consists of small pieces, which is easy to expand th capacity.
- the storage uses multiple replicas and consensus algorithm
- functions of backup, recovery and snapshot are done in the storage system.

II. Current Progress

To architecting LogStore, we've been learning basic data structure and database key technology, and distributed principle and application development. So far we have implemented B trees, filters, transactions and concurrency control.

A. B tree and B+tree

They are introduced respectively in the books Introduction to Algorithms [3] and Database system concept [4]. We have brought them out in the language of ts according to the introduction and Pseudo code in the book. For testing we have tested operating numbers and strings. Next step we will rewrite them in c++ language and enbed the data structures into the storage system.

B. Hash

The thaining plan contains two hash method to learn, consistent hash [5] and chord [6]. We have implemented the consistent hash in golang and c++, which maintians enough virtual and real servers(A1, A2, C1, C2) when the real ones are not enough(A1, C1).

Chord chooses SHA-1 [7] as hash function and we have implemented the SHA-1 function, which transforms a string into a 160 bit data. Hash functions are used together with compression method when the string data is very large. In the method introduced in the next paragraph, hash and compresstion will be used.

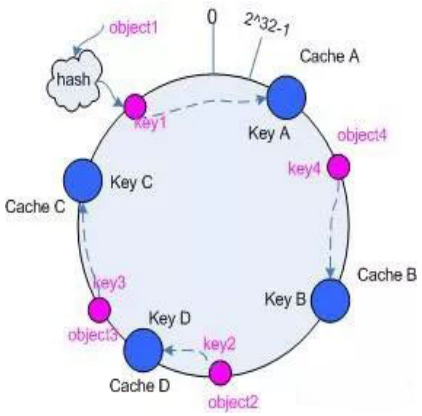

Fig. 2. Consistent Hash

C. Mapreduce

Based on attached resourse of the course Mit6.824 [8], we have implemented a mapreduce algorithm in c++, which has function of the example(section 2.1). Our inspired by the question that how to classify the logger of logs in such a multi-tenant system. So we will use mapreduce to classify logs of the same logger: same tenent, same application and same module or class. The Map function takes an log and identifies the logger of the log, then prodeces a set of intermediate logger/rest of log pair.Then hash-compression method will be applied to either component of the pair.

References

[1] Alibaba cloud native architecture practice, 2021.4, China Machine Press
[2] The Twelve-factor App, retrieved from https://12factor.net/zh_cn/
[3] Introduction to Algorithm, 2016.10, China Machine Press
[4] Database system concepts, 2020.10, China Machine Press
[5] consistent hash, retrieved from https://www.cnblogs.com/cloudgeek/p/9427036.html
[6] chord, retrieved from https://www.cnblogs.com/zfyouxi/p/3871553.html
[7] SHA-1, retrieved from https://blog.csdn.net/u010536615/article/details/80080918
[8] MapReduce: Simplified Data Processing on Large Clusters, Jeffrey Dean and Sanjay Ghemawat, 2004.