LogStore: Architecting Log Storage System for Cloud-Native Database

1st Fanzhong Wang
11911824

2nd Yifan Zhang
11911109

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. For enterprises who use cloud-based services to run their applications, they are expected to work stably, efficiently and securely. But traditional database server are usually under-utilized 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 traditional 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.

***Index Terms*—LogStore, Cloud native, Distributed database**

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 f enough to analyze logs only at the beginning or the enc only on bugs appearing. Logs during a method is even r useful, which contains: time, level of the log (warn, e etc.), call chain id(optional), thread name, class name, cor of the log, and some logs of level warn or error have excep stacks.

Logs are stored in a steady file named in some templat writing or has rolled back. However, logs can added up very large size, so we must compress each piece of log string before storing it.

B. Cloud native

With the evolution of cloud native, it has many vers of definition, including Pivotal, CNCF and the twelve-factor app. Raised by Heroku(2011) [1], the twelve-factor 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
- Separate 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 different. 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.

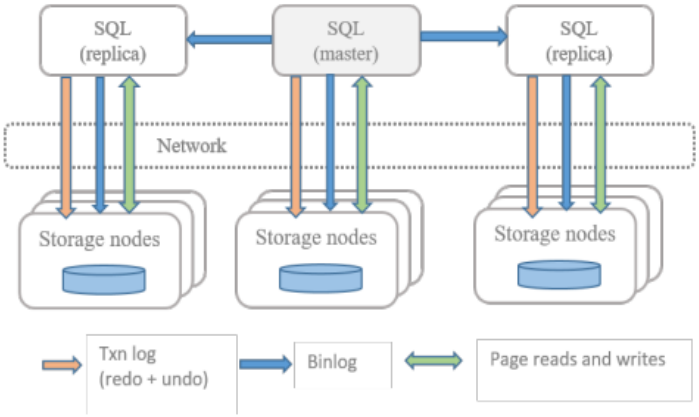

Fig. 1. MySQL with two replicas deployed in a cloud environment [9]

C. Cloud-Native database

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

- the computation and storage system are separated, and computation 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 the capacity.
- the storage uses multiple replicas and consensus algorithm
- functions of backup, recovery and snapshot are done in the storage system.

D. LogStore

Some pioneers have architected a cloud native log database called LogStore. There are many alternative architectures in the implementation of parallel computing architecture, including: [3]

- shared-memory: Multiple cpus share the same memory and they are connected by interconnection network.
- shared-disk: Every cpu has a private memory, and each can directly access the whole disk system.
- shared-nothing: Every cpu has private memory and disk and can connect by interconnection network. However, each pair of cpu cannot access the same disk area.
  The pioneers combines shared-nothing and shared-data(the first two shares) architecture to make storage highly scalable and low-cost. For our project, we choose shared-nothing architecture, shared-nothing is capable of decreasing waiting time and recieving better Scalable performance, which qualifies our requirement for the distributed storage system.

E. InnoDB

The ACID(short for atomicity, consistency, isolation and durability) of a transaction is based on logs. D is guaranteed by redo log and A is achieved by undo. Innodb is an engine of mysql and is the default storage engine. Its biggest feature is to support transactions that ACID are compatible with. And it manages redo and undo Now we are not very familiar of this engine and will practice using in the next stage of the project.

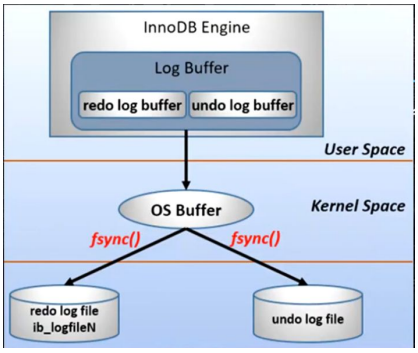

Fig. 2. InnoDB engine structure [12]

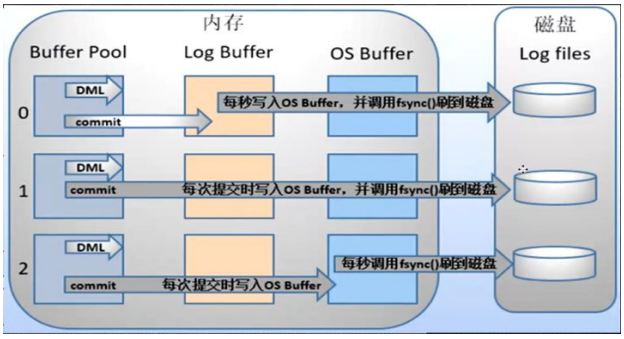

Fig. 3. InnoDB [11]

II. Shedule

The superviser has given us a training plan of distributed database development.

In the first stage of the project, we are aimed to study concpets and algorithms in the plan. On interest of time cost, we will study concepts and implement the pseudo-code of algorithms mainly rely on the books Introduction to Algorithms [4] and Database system concept [5]. Besides, we will read essays on introduction to some cloud native database.

In the second stage, we will implement the practice examples in the plan and develop our system of log storage. Zhangyifan is responsible for the storage system and Wangfanzhong for computation system. We will implement the storage and computation system respectively.

Except continuing to study algorithms and datastructure, we will:

- apply datastructures(Btree, filter) and algorithms to practices and our system, and later optimize them.
- test different compression functions to compress every log
- practice using innodb more fluently.
- Select a data structure to implement concurrent read and write to ensure data consistency

Final, besides the two tool books above, we will study from the reference of other successful databases like Spanner and Aurora and upgrade our algorithms and datastuctures. The difficulty is that they uses better services privided by other products like OSS of aliyun, however, we would find alternatives to implement the support systems.

To architect the LogStore, we plan to divide it into two parts: Computing Layer and Storage Layer. The first part need a set of brokers to process SQLrequests. Then, the broker merges the responses from each shard and returns the final result to the client. This layer will be implemented by two teammates, and the Storage Layer is in charge of another teammate.

III. Current Progress

To architecting LogStore, we've been working on 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. Consistent Hash

The plan contains two hash method to learn, consistent hash [5] and chord [6]. We have implemented the consistent hash in golang and c++, which maintains enough virtual and real servers(A_1, A_2, C_1, C_2) when the real ones are not enough(A_1, C_1).

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 compression will be used.

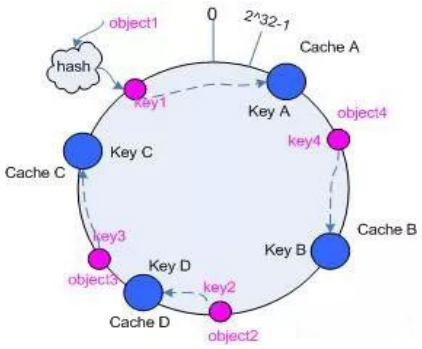

Fig. 4. Consistent Hash [10]

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 produces a set of intermediate logger/rest of log pair.Then hash-compression method will be applied to either component of the pair.

Algorithm 1 Mapreduce Algorithm

Input: $value \leftarrow logString$
1: **function** $MAP(value)$
2: $key, subvalue \leftarrow SPLIT(value)$
3: $EmitIntermidiate(key, subvalue)$
4: **endfunction**
5: **function** $SPLIT(log)$
6: $logger \leftarrow findLogger(log)$
7: return: $logger, FILTER(log, loger)$
8: **endfunction**
9: **function** $REDUCE(key, value)$
10: add value to set of key
11: **endfunction**

D. Concurrency

Concurrent programming divides a program into separate, independently running tasks. By using multi-threading, each of these individual tasks will be driven by the thread of execution. Concurrency can make full use of CPU resources and improve the performance of programs running on a single processor.

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.
[9] LogStore: A Cloud-Native and Multi-Tenant Log Database. Wei Cao, Xiaojie Feng, Boyuan Liang, Tianyu Zhang, Yusong Gao, Yunyang Zhang, and Feifei Li. 2021. Association for Computing Machinery, New York, NY, USA, 2464–2476. https://doi.org/10.1145/3448016.3457565
[10] https://mmbiz.qpic.cn/mmbiz_jpg
[11] https://pic3.zhimg.com/80
[12] https://pic1.zhimg.com/80