Research and Application of High-Performance Data Storage and Transmission Technology in Microservice Environment

Kunying Li, Yu Ding, Ying Shi, Liling Wang, Zebing Zhen

Computer Application Technology Department,
PetroChina Research Institute of Petroleum Exploration & Development,
Beijing 100083, China
likunying@petrochina.com.cn

Abstract. With the development of the Internet, various types of application systems are emerging, which brings more and more server access, more and more content, and more and more strong interaction between users and the system. The new generation of microservice software development framework brings great pressure for data transmission and storage. In order to improve the stability of data storage and read and write speed, this paper proposes a cache service architecture designed by sentinel mode. Redis is used as the core intermediary layer of high-performance storage to solve the lightweight and high concurrent data, and multiple linear regression analysis is used to determine those data that need asynchronous transmission and heavyweight transmission with RabbitMQ message queue for transmission. Finally, it realizes the efficient storage, expansion and effective and timely information communication of data in the whole microservice environment.

1. Introduction

With the advent of the big data era and the growing number of applications. Then the use of shared memory storage in traditional software development will greatly increase the data loading time, and ultimately make the system response time too long, seriously affecting the normal use of users. Therefore, how to ensure the high efficiency and stability of massive data interaction is a problem worthy of discussion in this field. At present, more and more enterprises have made breakthroughs in the technology of data transmission layer and achieved good results. Through the implementation of network interface equipment, the data message is directly stored in the address space that can be accessed by the application program in DMA mode [1], so as to avoid the memory operation when the data message is transmitted in the kernel state and shorten the walking path of the data message. The method of data storage and data curve realization in the software are studied. The dynamic chain table is used to realize the efficient chain dynamic storage of real-time data, and the dynamic display of real-time and historical data curve is realized by calling windows API function [2]. When more data is stored in the dynamic array, the data operation efficiency of the application program on the dynamic array is lower [3]. A realization method of chain dynamic storage is designed to solve the problem of reading and writing the dynamic array. A reliable chain information frame transmission method is proposed, which can greatly improve the utilization of serial communication bus, and realize high-speed and reliable information transmission by using simple communication body such as twisted pair [4]. It can be seen that currently there are solutions for data storage and transmission from algorithm to hardware, with obvious breakthroughs in performance and technology. However, in many cases, a single technology
cannot solve all the problems, because the different pertinence leads to the different solutions of the technology to the problems, which ensures the efficient transmission of data, and often cannot guarantee the data can be stored for a long time. Therefore, this paper uses the high memory server to cache the data needed to be transmitted on the whole cluster, improves the performance of the management node, and avoids the disk space bottleneck of the microservice master node. Redis (remote dictionary server) is used for storage on these high memory cache servers [5]. It is a high-performance key value database based on memory reading and writing. It supports a lot of data types including string, linked list, set and hash. Redis is obviously not suitable for asynchronous data transmission and large-scale data processing, so on this basis, it uses multiple linear regression analysis to divert these data to [6] RabbitMQ, a message queue specialized in batch data asynchronous processing, parallel task serialization and high load task load balance. Finally, a set of perfect application system of high-performance data storage in the micro service environment is realized, and its good expansion ability is guaranteed, which provides a compatible mode for other project construction in the future.

2. Optimization design of high-performance data storage

The scheme of data storage and transmission designed in this paper is to establish an intermediate media platform between users and servers, servers and servers, which is specially used to handle the upload, subscription, publication and deletion of received data. The middle layer needs to face the test of coming and all kinds of data and capacity. It should not only have high usability, expansibility, high availability, but also maintain the actual demand for data real-time consumption, asynchronous delay transmission, persistent storage, and finally achieve the response efficiency without affecting the business layer. Therefore, in this study, Redis and RabbitMQ are combined to give full play to the advantages of instant data analysis and cache processing, and solve the problems of asynchronous data processing and high load tasks. The specific technical architecture is shown in Figure 1:

![Data transmission flow chart](image)

The data sender sends the data to the cache server first, then transmits the lightweight data to Redis for storage and distribution through classification calculation, and sends the rest of the large data that needs special processing to RabbitMQ for saving, then subscribes and forwards according to the design parameters, and the last two types of data will be fed back to the data receiver.

2.1. Cache server architecture design

In this study, the cache server is set to cluster mode deployment, in which the primary node is used as the receiving read-write operation and the secondary node is used for data backup to prevent data loss. There are 16384 hash slots in this architecture. The storage slots obtained by each primary node can be obtained according to the following algorithms:

$$HASH\_SLOT=CRC16(key) \mod 16384 \quad (1)$$

Through the method of dividing slots, each master node only stores a part of all the data, and maximizes the use of memory. Test data shows that a Redis cluster environment with a total node number of 200 is deployed on 20 physical machines, each of which is divided into 10 nodes. If the timeout is set to the default 15 seconds, then the Ping / Pong message occupies 25mb of broadband at this time, but if the timeout is set to 20, then the consumption of bandwidth is reduced to below 15MB.
When the single access data is too high, the efficiency becomes low, but when the content length is relatively short and the receiving speed is fast, the occupancy rate of bandwidth decreases obviously. According to the above findings, we set a sentinel service in the cache server, as shown in Figure 2. In the operation of data reading and writing, we use the sentinel service to traverse and analyze the packet information, obtain its length, sending object, sending urgency and service availability. In the process of analyzing data package, the cost of CPU and memory calculation is very small, so that the impact of data allocation in the whole process can be ignored.

When there is high concurrent access, first of all, according to the characteristics of real-time monitoring of the service for the use of Redis and RabbitMQ, you can effectively know the operation of the entire cache server, and according to the judgment results of multiple linear regression analysis, make data to the corresponding data processing services for storage and transmission. In order to prevent repeated and repeated reading of the same data in the process of direct reading, the sentinel service uses a one-way output mode based on time series. When the record is successfully dispatched, it will be deleted from the memory to ensure the uniqueness of the sentinel environment and the data dispatch processing layer, so as to avoid data conflict and deadlock.

2.2. Design of message communication pipeline in big data
Most of the time, the system will involve the centralized processing of large-scale data. For example, when the file is processed uniformly, the data will often reach the level of GB and need to be transferred asynchronously through a dedicated channel. At this time, a stable message channel is needed to serve it. In this architecture design, RabbitMQ’s message model technology is used, which is composed of message publisher, message subscriber and message server. The message publisher adds the data to the server, and the server requests the task allocation processing module through different virtual hosts. Each virtual host contains an independent data exchange module and message queue. As the core data storage and transmission body, the message queue sends the data to the specified subscribers according to the order of the data in the stack. If the transmission fails due to an abnormal error or network reason, the data will be added to the sending task sequence by the exchange module, and all the data will be stored in memory or hard disk to ensure that the data will not be lost. In this way, the whole process of sending and subscribing message pipeline is realized.

3. Server load distribution processing
Background server load distribution is to ensure the reliability of data storage and transmission, the efficiency of service access, and save storage space. In order to establish the server load task allocation model, we need to consider various factors that affect the server, and then carry out quantitative evaluation. The CPU utilization (C), memory utilization (M), disk read-write efficiency (D) and current time interval (P) in the whole environment are calculated by a multiple linear regression analysis method.
If the above four conditions are taken as variable conditions and the final data execution time ($T$) is taken as result variable, then the multiple regression equation can be expressed as:

$$T = \alpha C + \beta M + \gamma D + \delta P \quad (2)$$

Then by monitoring the CPU, memory, disk read-write speed and the current time period, it should be noted that the concurrency rate in the daytime is much higher than that at night, and the service restart time set by many servers is also set in the early morning. Observe the service progress that C, M and D have a great influence on in the daytime and at night, and record the actual time when the final data is executed. After regression analysis, we get $\alpha = 0.145$, $\beta = 0.267$, $\gamma = 0.129$, $\delta = 0.247$. Under this condition, the time of $T$ is less than the default time 200ms set by the cache server, which can be used as the established threshold.

In addition, a message queue dual execution optimization algorithm [8] AssesModel based on the definition of periodic execution and category priority is also introduced in the design of message pipeline. The algorithm is realized by associating the operation behaviors of user role, system and time, and combining the defined message importance label. When calculating the resource $TAP_{t,i}$, it is found that the message data to be executed is about to reach the full value. It and all the message data waiting for the queue behind it will be evaluated by the AssesModel evaluation model. If the evaluation value is greater than the set reserve resource access standard, the batch of messages will be sent to the priority execution queue after load balancing. This scheme effectively solves the problem of filtering, sorting and priority execution of massive message throughput in the distributed working environment, and reduces the performance and delay risks caused by the complexity of integrated system information.

4. Experimental results and analysis

In this experiment, 64-bit Windows Server 2012 R2 is used as the server operating system and. Net core is used as the development language. Then IIS is used to deploy the service application. At the same time, Redis and RabbitMQ are deployed in the cluster mode, and sentinel service is set to monitor and assign tasks. Finally, data is set to be saved in the log document every second. The test data samples are randomly generated 100000 files of different sizes, which are tested for storage and transmission in different time periods. The experimental contents are as follows: (1) test whether the overall data storage and flow is feasible. (2) Compare the CPU, memory and disk I/O flow with the system without this scheme. (3) Monitor data flow to ensure that tasks are assigned according to the specified setup parameters. (4) According to the monitoring data, the performance effect of the scheme is tested.

Based on the algorithm of multiple regression, the load distribution of the server is processed, and the specific monitoring data is as follows:

| T/ms | C/%  | M/%  | D/(Byte/s) | P/(daytime|night) |
|------|------|------|------------|-------------|
| 47   | 27   | 33   | 16.02      | daytime     |
| 53   | 14   | 35   | 19.44      | daytime     |
| 32   | 33   | 16   | 115.67     | night       |
| 45   | 38   | 18   | 43.15      | night       |
| 93   | 58   | 35   | 67.42      | daytime     |
| 76   | 53   | 48   | 32.26      | daytime     |
| 125  | 48   | 56   | 55.61      | daytime     |
| 43   | 12   | 14   | 77.83      | night       |

From the above experimental data, it can be concluded that the disk write efficiency is the highest at night, and the response time is lower than that of daytime on average. From the overall analysis, the whole environment is within the scope of 200ms, which meets the requirements of the experiment.
Finally, according to the execution rate of the log statistical analysis task after the execution of the experimental sample data, it can be seen from the above figure that in this experiment, the design model has obvious different channel flows for the data, most of which are small data that can be processed in time through Redis, while the RabbitMQ channel is responsible for large file processing. Therefore, from the above results, we can see that the optimized scheme can reasonably allocate resources to the corresponding processing module, ensure the effective operation and flow of data, improve the utilization rate of memory and the reliable transmission and storage of data.

5. Conclusion
In order to achieve high performance and high stability of information and data storage in the transmission capacity, this paper constructs a cache service architecture based on sentinel mode design, and on this basis, classifies data using Redis and RabbitMQ storage and transmission methods for scattered sub nodes. By using multiple linear regression analysis and message queue double execution optimization algorithm, the server load distribution processing model is established, and the data throughput is accurately screened, sorted and prioritized to form an effective distribution cooperation mode. Finally, the efficient storage, expansion and effective and timely information communication of data are realized in the whole microservice environment.

Acknowledgments
This work has been supported by the information management department of the group company. Thank them again for their efforts in this project.

Reference
[1] WANG Bai Ling FANG Bin Xing YUN Xiao Chun. The Study and Implementation of Zero-Copy Packet Capture Platform. Chinese Journal of Computers,2005,28(1):46-52.
[2] ZHANG WeiQiang LU Mo-wu. Study on the implementation of data storage and data curve in monitoring and control system software based on VB6.0. Journal of Shenyang Institute of Aeronautical Engineering,2003,20(1):31-34.
[3] LI Weidong, HAN Laping, GUO Junwen. Implementation of Dynamic Storage Based on Link in VB. Computer Engineering,2002,28(2):255-256.
[4] Huang Tianshu He Ping. A METHOD OF RELIABLE LINKED-FRAME COMMUNICATION. journal of Computer Applications,1999,19(11):18-20.
[5] TU Xue-zhen. A Large-Scale Redis Cluster Cloud Operation and Maintenance Technology. Computer Knowledge and Technology(Academic Exchange),2019,15(3Z):1-3.
[6] Zhou Jian. RabbitMQ in Integrated Management and Control Platform of City Smart Pipe Gallery. Industrial Control Computer,2019,32(9):12-13.
[7] Zhao Yun; Tang Xuqing. Multiple Linear Regression Problem Based on Hierarchical Structure Data. Journal of Data Acquisition & Processing, 2019, 34(5): 883-892.

[8] Kunying Li, Dexin Qiao, Xiaolian Li, Yu Ding, Liling Wang, Message Queue Optimization Model Based on Periodic Execution and Category Priority, 2019 The 2nd International Conference on Computational Intelligence and Intelligent Systems, 2019.