Research on load balancing technology for microservice architecture

Hao Wang[1], *Yong* Wang[1,*], *Guanying* Liang[1], *Yunfan* Gao[1], *Weijian* Gao[1] and *Wenping* Zhang[1]

[1]Harbin Engineering University, College of Computer Science and Technology, 150001 Harbin, China

Abstract. With the emergence and development of new software architectures such as microservices, how to effectively handle the service load and ensure the service capability of the system has become an urgent problem to be solved. Load balancing technology needs to achieve high availability of microservices without affecting the delayed response of requests. According to different principles of adoption, mainstream load balancing technologies have emerged, such as polling methods, hash algorithms, and artificial intelligence technologies. This article categorizes and summarizes load balancing technologies for microservice architecture, and elaborates the methods and characteristics of current mainstream load balancing technologies. Based on the comparative analysis of existing technologies, this paper summarizes and points out the future development direction of load balancing technology.

1 Introduction

In recent years, the new generation of information technology, represented by big data and cloud computing, has triggered a new wave of scientific and technological revolution and industrial transformation. The demands of emerging industries and the new model of sharing economy have brought new challenges and requirements to the evolution of software architecture. Software system architecture is an abstract description of the overall structure and construction of software. It is the sketch of software system and the basis of building computer software practice. Typical architecture models include Monolithic Architecture, Distributed Architecture, and Service-Oriented Architecture(SOA), etc. Among them, Microservices, Middleground, and Cloud Native have become popular new development trends in the current software system architecture field.

The concept of microservices was jointly proposed by Martin Fowler and James Lewis in 2014[1]. Different from the traditional monolithic application that packages all functions into an independent unit, microservices decompose a single application into a set of small services. Each service runs independently in its own process, and the services coordinate with each other and work together to realize the value of the system. Microservice is an important model and typical technology of current software system architecture. It has the

* Corresponding author: wangyongcs@hrbeu.edu.cn

characteristics of lightweight, fast iteration, and cross-platform. It can make the deployment, management and maintenance of the system faster and more convenient, which has gradually become the main direction of system architecture technology development. Microservice has the advantages of independent services, decentralization and high fault tolerance, but also faces large-scale complexity among services communication and complicated service management.

Currently, microservice architecture mainly includes service registration and discovery, service monitoring, service communication, load balancing, elastic scaling, fault-tolerant mechanism, configuration center and other microservice governance technologies[2]. The microservice architecture can be simplified as shown in Figure 1 below:

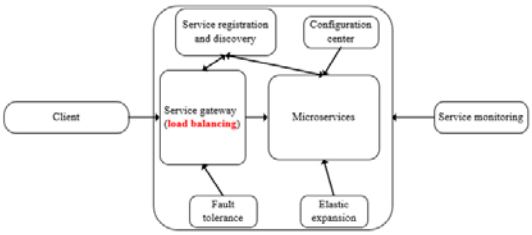

Fig. 1. Mainstream architecture of microservices.

Load balancing refers to the uniform forwarding of user's access requirements to the back-end service carrier through a certain strategy. Under the microservice architecture, cluster scheduling and the granularity of expansion is reduced to the level of microservice instances, and the microservice governance technologies such as service communication and invocation are facing greater challenges. Through the load balancing technology for microservice architecture, it can effectively reduce the request delay, make the system have higher availability, and effectively improve the service ability and resource utilization of the system.

2 Classification of load balancing methods for microservice architecture

According to the principle of whether the request can be allocated in real time according to the server status, the current mainstream load balancing algorithms can be divided into two categories: static load balancing algorithm and dynamic load balancing algorithm. Static load balancing algorithm mainly includes Polling Method, Random Method, Source Address Hashing Method, Consistent Hash Method, Weighted Polling Method, etc. Dynamic load balancing algorithms mainly include Minimum Connection Number Method, Fastest Response Method, etc.

According to different technologies for allocating service requests, popular load balancing technologies can also be divided as random method, polling method, hash method, optimization method, and artificial intelligence-based methods.

3 Mainstream load balancing techniques for microservice architectures

3.1 Load balancing technology based on polling method

The load balancing technique based on polling method can set up a group of servers such as {S0, S1, S2 Si. … Sn}, where Si represents a service. When the client sends a request, the

server will be requested from S0 to Sn in turn, until the end of one round, and then proceed to the next round of requests. The advantages of the polling method are its simple and efficient technical principle, while the disadvantages are that the network load and link call problems existing in the microservice framework are not considered, which will lead to uneven load of servers and work nodes.

Table 1. Mainstream load balancing method for microservice architecture.

Algorithm name	Classifications	Advantages	Shortcomings
Polling Method	Static State	It has the advantages of balance, high efficiency, simple implementation and easy horizontal expansion.	Without considering the performance and resource problems of the server, the overall performance of the cluster will be affected by the server with poor performance.
Random Method	Static State	It is efficient, simple to implement, easy to expand horizontally, and the more the number of requests, the closer to the even distribution.	Without considering the performance and resource problems of the server, the overall performance of the cluster will be affected by the poor performance of the server.
Source Address Hashing Method	Static State	The implementation is simple, and the same IP address can ensure that requests are sent to the same server.	The overall performance will be affected when the number of servers changes.
Consistent Hash Method	Static State	It has fault tolerance and is easy to expand horizontally.	In the case of small cluster, uneven server mapping may lead to load imbalance.
Weighted Polling Method	Static State	Taking the performance of the server into consideration, the efficiency of the cluster is optimized.	The production environment is complex and changeable, and the server performance cannot be dynamically adjusted after being statically set.
Minimum Connection Number Method	Dynamic	It can be dynamically adjusted according to the real-time connection of the server.	The complexity is increased, and the disconnection and connection of each request need to be counted.
Fastest Response Method	Dynamic	It can be dynamically adjusted according to the server's real-time response speed.	The complexity is increased, and the response speed needs to be calculated.

In order to solve the above problems, domestic researchers have improved and summarized it. By collecting the request load perceived by the microservice chain, Analyze the requested microservice invocation chain, and the load analysis model is constructed by using the relationship of microservice invocation chain. Then, the cost function is designed to traverse all service invocation combinations, and select the optimal path with the shortest time required, and carry out load balancing scheduling in turn, which effectively reduces the average delay of requests and significantly improves the availability of services[3].

The other strategy is to use the load balancing technology based on SDN, through the method of coupling the SDN network and the microservice architecture, to realize the more fine control and management ability of the network, and better transplant the load balancing technology to the microservice. At the same time, the tracking capability of SDN for service link is utilized to implement the load balancing algorithm based on link analysis, so

as to analyze the overall link load and provide current limiting function to prevent the access from affecting the high availability of the system too much[4].

3.2 Load balancing technology based on hash algorithm

The load balancing technology based on hash algorithm is widely used in the field of load scheduling and optimization in distributed systems, cloud computing and microservices. At present, it mainly includes load balancing technology based on hashing modulus strategy and load balancing technology based on consistent Hashing algorithm.

The principle of load balancing technology based on Hash modulus strategy is to design an appropriate Hash function to calculate the number of service requests and nodes to determine which node the request will be distributed to. Its characteristics are simple and efficient, and can ensure that the same requests of the system are processed by the same machine, reducing data migrations and nodes communication. However, in the actual environment, if the number of nodes changes dynamically due to the failure of the system, the calculation results will be greatly changed, and a large number of data migrations will be generated, reducing the availability of system services.

The load balancing technology based on consistent hashing algorithm is an improvement on the former. The algorithm principle is as follows [5] :

First, the corresponding hash value of the node is calculated and assigned to a circle of $0\text{-}2^{32}$. Second, Find the hash value of the service request and map it to the same circle. Finally, The lookup is performed clockwise from the location of the request, and the first node found is the visited node. The load balancing technique based on the consistent Hashing algorithm has good lateral scalability. Moreover, different from the dynamic change of the number of nodes based on the Hash modulus strategy, it can effectively solve the problem of data nodes change and data ownership, and avoid a large number of data migration problems caused by the above problems.

However, the load balancing technique based on the consistent hashing algorithm has no load balancing measures, which will cause data skew between nodes. In view of the above problems, relevant scholars have made lots of improvements, mainly including:

A load balancing technique based on division and greed. By dividing the hash ring, the servers that can provide corresponding services are divided. In addition, a physical server is mapped to multiple virtual nodes according to its functions. When a request comes in, it first looks for a suitable partition and then maps the characteristics of the request to a hash value. According to the greedy algorithm idea, it selects a service node between the lowest load and the load mean counterclockwise [6]. However, greedy algorithm does not consider the overall level but only considers the local optimal of each step, so it is more suitable for the scenario with uncertain data sets.

Based on dynamic weight, consistent hash algorithm load balancing technique is adopted. For the system whose data sets can be determined in advance, the virtual nodes partition method can solve the problem of data skew from the whole level. Mapping services to virtual nodes, constructing a virtual layer to solve the problem of too few virtual nodes. At the same time, the number of service requests is used to construct the dynamic weight. Based on the dynamic weight, the consistent hashing algorithm is used to balance the load. When the upper limit of service weight is not reached, this service will be selected; otherwise, other services will be selected. When all service weights reach the upper limit, a new service will be created [7].

3.3 Load balancing technology based on artificial intelligence

The other strategy is to use the SDN based load balancing technology, With the rapid

development of artificial intelligence technology, the optimization algorithm based on artificial intelligence is gradually applied to the field of load balancing. The typical load balancing technology is based on artificial bee colony and the load balancing technology based on ant colony optimization algorithm.

The load balancing technology based on artificial bee colony has the advantages of easy implementation and wide application range. In this algorithm, the service request is regarded as a bee, and the load node is taken as the honey source. The algorithm solves the load balancing problem with the idea of artificial bee colony algorithm: the nodes are arranged according to the service load from high to low. If the load of a node is too high, the task in the load will be removed, and the task will be migrated to the node with low load or high priority and less tasks. The simulation load and task priority are broadcast to all computing tasks [8].

In the load balancing technology based on ant colony optimization algorithm, heuristic function and pheromone function are set first, and then they are initialized according to the service resources of the node. In order to achieve the effect of load balancing, the pheromone will not only consider the load of nodes, but also consider the computing and communication capabilities of nodes. This technology can ensure the avoidance of falling into the problem of local optimal solution and effectively improve the reliability of the system and the availability of services [9].

4 Application of load balancing technology in mainstream microservice framework

4.1 Load balancing mechanism in Kubernetes

Kubernetes is an open source micro-service management platform. It can be used to manage containerized services. Through effective management of services and resources, the kubernetes platform can significantly improve the utilization rate of cluster resources and reduce maintenance costs. The load balancing mechanism in kubernetes system is mainly implemented through ingress, and the specific technical principles are as follows:

kubernetes starts each corresponding work instance of the application service, which is called a pod. The service is a resource defined by kubernetes at the management level to manage the dynamically and randomly assigned IP addresses when an instance is created.The IPs of pod and Service are accessible only within the cluster. If an external service is accessing it, ingress needs to provide an entry for external access and provide routing rules for access requests received by the cluster.

Ingress is the core of the load balancing mechanism of the kubernetes platform. Ingress consists of three parts:

Ingress-nginx: A reverse load proxy balancer that forwards requests through well-defined Ingress rules.

Ingress-nginx-controller: Used to interact with the kubernetes cluster apiserver, ingress-nginx-controller dynamically updates the rules within the nginx service when the user updates the routing rules of ingress.

Ingress configures： An ingress object for each nginx, and creates and updates ingress resources in the form of yaml in kubernetes.

Ingress-nginx currently supports load balancing methods such as Round-robin, Peak Ewma, Chash, and so on.

4.2 Load balancing mechanism in the spring cloud

Spring cloud is a collection of SpringBoot-style and encapsulated service frameworks. By taking advantage of SpringBoot's easy-to-develop features, a batch of distributed service components can be quickly built that ultimately form the Spring cloud framework. Ribbon is used primarily in the Spring cloud framework to achieve load balancing.

As a client load balancer, when a client is required to send a request to a server, Ribbon calls the Eureka Server, the service registry of Spring cloud, to get a list of service providers, and then accesses the corresponding address using a load balancing algorithm based on the registry address returned by the registry. Ribbon load balancer can provide a variety of load balancing mechanisms, including polling, randomization, and so on.

5 Conclusion and prospect

Microservice is a new development trend in the field of software system architecture. Load balancing is an important content of microservice architecture technology, which determines the service performance and resource utilization of the whole system. This paper classifies the load balancing methods for microservice architecture at first, describes the basic principles and advantages and disadvantages of mainstream load balancing technologies, and finally introduces the load balancing mechanism strategies of Kubernetes and spring cloud framework, which have absolute market share. However, most of the current load balancing technologies in microservices are migrated from the cluster system load balancing technology. In view of the current service load situation, assuming that the service requests are homogeneous and the node resources are fixed, and the inherent call chain, small granularity and single task of microservices are ignored, how to introduce machine learning technology to deeply analyze the massive historical log data for service load forecasting and constructing global optimal load balancing model according to the dynamic changes of node resources are the main research directions in the next step.

Reference

1. M. Fowler, J. Lewis, Viittattu, 28 (2014)
2. S. Newman, Building, O'Reilly Media, (2015)
3. L. Shi, Z. Zhu, J. Zhou, X. Li, J. Li, Comput. Eng (2020)
4. W. Jiang, S. Pan, J.COMPUT.SCI.TECH-CH, 30,**02** (2020)
5. K David. R Matthias, THEOR. COMPUT. SYST, 39,**6** (2006)
6. K. Q. Zhang, X. Y. Liu, X. Wang, C. S. Ji , X. Yan, Comput.Sci.Eng , 08, 42 (2020)
7. C. Wang, Q. Y. Li, N.S, 03, 38 (2018)
8. J. Jia, D. J. Mu, Sci. Tech. Eng, 16, 20 (2020)
9. H. Q. Zhang, X. P. Zhang, H. T. Wang, Y. H. Liu, M.A.C.S, 05, 32 (2015)
10. L. Abdollahi Vayghan, M. Saied, M. Toeroe, CLOUD 2018, **1** (2018)
11. 11. S. Buchanan, J. Rangama, N. Bellavance, Inside Kubernetes (Apress, Berkeley, CA, 2020)