Journal of Combinatorial Mathematics
and Combinatorial Computing
www.combinatorialpress.com/jcmcc

Combinatorial Press

Research on load balancing techniques based on numerical analysis in distributed systems

Huaijiang Teng[1], Zhuo Jiang[1,] ✉

[1] *Heilongjiang Open University, Harbin, Heilongjiang, 150080, China*

ABSTRACT

Based on the demand of load balancing in distributed system scenarios, this paper introduces the concept of dynamic priority in the algorithm and designs the dynamic feedback load balancing (DFLB) algorithm for numerical analysis. Through the closed-loop process of collection-feedback-utilization-collection, the overall performance of the system is realized. The Mininet tool and the Floodlight controller are used when building the load balancing system experimental environment to verify the reliability of the algorithm from the response delay, throughput and other indicators. The study shows that the DFLB algorithm reduces the response time of the system by about 20% compared with the static deployment method, and the DFLB algorithm reduces the load variance, saves computational resources, and makes the load of the system more balanced and efficient. The average throughput of the DFLB algorithm is improved by about 10% compared with the PALB algorithm and DALB algorithm, and 6% compared with the PALB algorithm and DALB algorithm, respectively. Starting from 1000 concurrent connections, the DFLB algorithm has a higher access rate. Thus, the algorithm leads to an improvement in the overall performance of the system.

Keywords: distributed systems, DFLB algorithm, dynamic feedback, load balancing

1. Introduction

Distributed systems are becoming an increasingly important part of modern computer systems. Distributed systems use multiple computers to work together and can provide higher performance and reliability [1-2]. However, in a distributed system, the load distribution of individual computers may be unbalanced, resulting in some computers being overloaded while others are idle, which requires load balancing mechanisms to balance the load among computers [3-6].

Load balancing refers to the rational allocation of tasks or requests to individual computers in a distributed system so that the load on each computer is balanced as much as possible. Load

✉ Corresponding author.
E-mail address: jz527109182@163.com (Z. Jiang).
Received 05 March 2024; Revised 20 June 2024; Accepted 15 November 2024; Published Online 16 April 2025.
DOI: 10.61091/jcmcc127b-204

balancing helps to improve the performance and scalability of the system and also improves the reliability of the system by preventing the failure of one computer from causing the entire system to crash [7-10]. There are various load balancing algorithms commonly used in distributed systems, one of the common algorithms is the polling algorithm. Polling algorithm assigns the requests to each computer in a sequential order and the cycle repeats itself, this algorithm is relatively simple and each computer has the opportunity to perform the task but it cannot take into account the actual load of the computers [11-14]. Another common load balancing algorithm is an algorithm that is dynamically adjusted according to the load of the computer, for example, according to the CPU utilization, memory utilization and other indicators to determine the weight of task allocation. This algorithm can adjust the task allocation more flexibly, but it requires real-time monitoring of the computer's load situation, which increases the complexity of the system [15-18].

Load balancing plays an important role in distributed systems, by using load balancer, the load of different servers or computers can be effectively balanced, which improves the scalability, reliability, performance, security and management efficiency of the system. Literature [19] proposes a solution based on load balancing and efficient utilization of resources in distributed systems and simulation modeling of fractal parameters of input flows reveals that the characteristics of multiple fractal flows have a significant impact on system imbalance, while the method is able to improve the performance and processing speed of the system. Literature [20] analyzed and verified the parallel computing method for computer distributed systems, showing that the distributed parallel computing system not only retains the advantages of the original system, but also effectively improves the distributed availability of the parallel computing system. Literature [21] explored a component-level based simulation system to solve the load imbalance problem and proposed a new integer planning model, revealing that the multi-objective particle swarm optimization algorithm outperforms other algorithms in solving the problem, while the proposed strategy effectively solves the load imbalance problem of the system. Literature [22] proposes a dynamic load balancing mechanism for distributed control of the load of the computing nodes, which significantly improves its performance in terms of performance as compared to the earlier distributed load balancing mechanisms. Literature [23] examined the load balancing problem for distributed systems consisting of a finite set of servers and a finite set of users and proposed the Distributed Load Balancing Algorithm (DLBA), which was validated by comparing DBLA with other load balancing methods. Literature [24] reviewed the research on task allocation and load balancing according to the characteristics of different types of distributed systems, and pointed out that the research in this area can be better understood by meeting the general characteristics of distributed systems. Literature [25] introduced the concept of dynamic load balancing to effectively manage the factors to be populated in distributed networks, and pointed out that load balancing has physical and logical characteristics, and optimization of time and cost can be achieved through load balancing. Literature [26] provides a systematic review of static, dynamic and nature-inspired load balancing techniques in cloud environments, aiming to cope with the problems of data centers in terms of response time and overall performance is, and mentions a fault-tolerant framework that analyzes other frameworks in the existing literature. Literature [27] aims at designing dashboards of consolidated load balancing algorithms that can be adapted to various medium environments, introduces a mechanism for task and resource allocation, describes in depth the mathematical model of the load balancing process in the system and proves the effectiveness of the methodology, which reduces the probability of system failures. Literature [28] mentions a dynamic load balancing algorithm based on monitoring server loads, self-similar characteristics of traffic, which is characterized by high performance, short response time, and low losses, and provides a comprehensive measurement of the overall imbalance level of the system. Literature [29] describes the classification of commonly used load balancing algorithms in distributed systems, and based on the classification a comparative analysis of the types of load

balancing algorithms is carried out, describing the strengths and weaknesses of various algorithms, and providing performance metrics to characterize the algorithms.

In this paper, in order to solve the load balancing problem of distributed clusters, a dynamic feedback load balancing algorithm is designed based on the dynamic data concept of numerical analysis. In the task allocation of load balancing, the processing capacity and load of the nodes are collected, and every once in a while, the load information of each node in the cluster is fed back. Based on the feedback information, the parameters of cluster load distribution are modified. Thus forming a closed loop process of collection - feedback - scheduling (utilization) - collection of dynamic feedback. Define the topology of the distributed system through Mininet, Iperf and Floodlight, and build the load balancing experimental environment. Test each functional module of the load balancing system and the performance of the system to verify the effectiveness of the DFLB algorithm.

2. Numerical analysis, distributed clustering and load balancing

2.1. Numerical analysis theory

Numerical analysis is a discipline that studies algorithms and summarizes industrial experience. The birth of numerical analysis methods can be traced back to the 15th century, but due to the limited technical means and computing power, the development of numerical analysis is relatively slow. 1946, with the rapid development of information technology and computer science, numerical analysis methods have been rapidly developed. In the 1980s, the birth and development of numerical analysis software, numerical analysis methods in aerospace, water conservancy and hydropower, rail transportation, petroleum and petrochemical, mechanical design and marine engineering and other fields of scientific research and engineering design has been widely used.

2.2. Systems with distributed servers

In the first decades of computer development, single node was a commonly adopted approach. A single node implies the use of only one server to house all the programs and data, and as time passes and business grows, the amount of data and pressure on the node becomes greater and greater. Moore's Law also ensures that the hardware performance of computers moves forward at the same rate to meet the demands of software. However, with the emergence of Internet technology in the last decade or so, the rise of the website visits have exploded, and the traditional single-node approach has exposed more and more problems: increased energy consumption. Large servers consume a tremendous amount of power, and a staggering amount of power is consumed to keep a large number of servers and devices such as routing up and running. Many companies are therefore placing their servers close to rivers, where they can get a lower price for their energy consumption through hydroelectric power plants, thus reducing costs.

Poor scalability. With the increase in business, the original node has been unable to meet the demand, the only way is to buy higher performance, larger capacity node equipment, in addition to the expensive price, the original server can only be idle processing, resulting in a huge waste. Difficult to ensure the quality of service. Although the stability of the server is higher than the average home computer, but in the case of high traffic, there is no guarantee that there will be no downtime. High software costs. Traditional server manufacturers, such as Hewlett-Packard and other companies, in the sale of hardware will be accompanied by a lot of commercial software, even if you do not buy new hardware, annual software upgrades, maintenance and other costs is also a considerable amount of money.

It is because of the many shortcomings of the traditional server, has become a bottleneck for many companies, especially the new Internet company's business development. In this context, clusters based on distributed systems emerged. Alibaba in the beginning of the year, launched a “go”

campaign, the traditional server system of proprietary systems, databases and storage devices were replaced with ordinary systems, databases and solid-state hard disk. After such a replacement, the original centralized architecture into a distributed architecture as shown in Figure 1: Cluster is a broad concept, it is a system, usually consists of a group of a greater or lesser number of computers, these machines are interconnected in some way. When providing a service externally, its internal structure is completely transparent to the caller and can therefore be seen as a single computer.

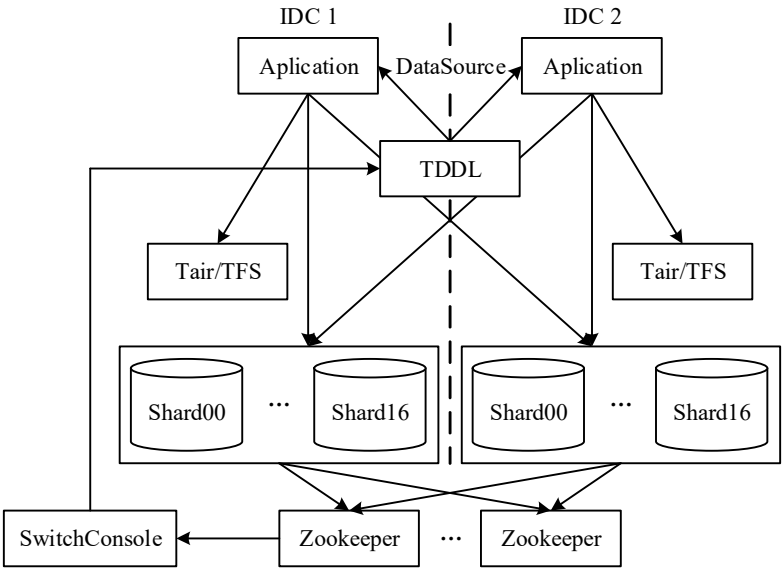


Fig. 1. Distributed architecture

With the emergence of distributed clusters, load balancing technology has become increasingly important. On the one hand, the request can be effectively dispersed to the nodes, really play the role of the "cluster", on the other hand, the entire cluster in the form of a single machine to the user, shielding the technical details, the user only needs to write the program as in the past, send a request to get the return result. Dynamic scheduling algorithm relative to static scheduling, dynamic scheduling algorithm added some feedback mechanism, this mechanism can dynamically collect the current cluster information, including the load of each node, the type of new requests, the number of new requests, bandwidth utilization, request delay and so on. Using this information, we can dynamically change the server's weights and some other attributes to change the direction of new connections. Such measures can play a good role in regulating the load peaks and unstable times, avoiding service skewing and downtime, and improving the overall performance of the cluster. However, there is also some performance overhead in order to collect information about the nodes and send it to the load balancer, which, if handled correctly, is insignificant compared to the benefits. Dynamic algorithms, due to their flexibility and huge room for improvement, have been gaining attention and research in recent years, and are increasingly being used in commercial software. While focusing on performance, we also need to focus on the overhead of the algorithms themselves, so there are also many strategies that combine static and dynamic algorithms, so that we can do both to reduce the overhead, but also to increase the flexibility appropriately.

3. DFLB-based load balancing technology for distributed systems

3.1. Load dynamic feedback

The concept of numerical analysis algorithm is to dynamically input, process and output data. As one of the more mature and accurate algorithms in the field of mathematics, numerical analysis algorithms have been widely used in various fields, and their reasonableness and scientificity have

been recognized by the industry, especially in the fields of path planning and goal control. This paper applies this numerical analysis concept to load balancing technology and proposes a dynamic feedback distributed system load balancing technology, i.e., dynamic feedback load balancing (DFLB). In the dynamic feedback load balancing algorithm, we are going to use the probabilistic or optimization level of numerical analysis to ensure that a large amount of data can be distributed to different servers in the most reasonable way In addition, numerical analysis is used to accurately control the error in the computation process during the operation of the dynamic feedback load balancing algorithm.

The simplest of the cluster load distribution strategies are random selection and polling. The idea of random selection is that for a long period of time, statistically speaking, each node in the cluster has basically the same probability of being assigned a task, if the cluster has N nodes, then each node is selected with a probability of 1/N whenever a new task is submitted. If the cluster is isomorphic, and the tasks submitted to the cluster are not very different, then the load of each node is counted over a long period of time, then they share the load roughly equally, and the system is basically stable. If the cluster is heterogeneous, that is, when the nodes in the cluster do not have the same processing capacity, the cluster load is not balanced, then if only the cluster nodes are heterogeneous, then we can increase the weight of each node to form a weighted random, assuming that each node's task execution capacity is $C_1, C_2, ..., C_n$, respectively, the probability of the node i being selected P_i as expressed in equation (1):

$$P_i = C_i \Big/ \sum_{j=1}^{n} C_j \quad i = 1, 2, ..., n \tag{1}$$

Dynamic load balancing takes into account the current load situation of the cluster, can be dynamically adjusted according to the node load situation, relative to static scheduling this type of scheduling is more conducive to cluster load balancing, we assume that the cluster has n slave node $N = \{N_1, N_2, \ldots, N_n\}$, the weight of each node N_i for $W(N_i)$, $T(N_i)$ for the number of the current tasks of the i node, and the total number of tasks of all the nodes of the cluster $T_{sum} = \sum T(N_i), i = l, 2, ...n$. There is a new task to submit selected to satisfy the formula (2) of the node N_k:

$$\frac{T(N_k)/T_{sum}}{W(N_k)} = \min\left\{\frac{T(N_i)/T_{sum}}{W(N_i)}\right\}, i = 1, 2, \ldots n \tag{2}$$

Since both the left and right sides of equation (2) contain T_{sum}, when T_{sum} is not zero, equation (3) can be obtained by simplifying (2):

$$T(N_k)/W(N_k) = \min\{T(N_i)/W(N_i)\}, i = 1, 2, ...n \text{ and } T_{sum} \neq 0 \tag{3}$$

This way of using weights to represent the processing capacity of nodes and the number of tasks to represent the load of each node does not take into account the differences in processing capacity among nodes, although it does not take into account the differences in the tasks to be processed. Representing the load by the number of tasks does not reflect the actual load situation of the current node, because nodes can provide multiple resources such as CPU, storage, I/O, etc., and the usage of each resource by individual tasks may be different, so the number of tasks does not really represent the load situation. In addition, after a period of time, the load situation of each node in the cluster will change, and the calculated load situation is not easy to modify, after a long period of time, the cluster load will become more and more unbalanced. To solve these problems it is necessary to design new algorithms so that the cluster load distribution will not become more and more unbalanced over time, so that the resources of each node are fully utilized, and the algorithms should

not be too complex, the basic idea is shown in Figure 2. Dynamic feedback refers to a cyclic process of collection - feedback - scheduling (utilization) - collection again.

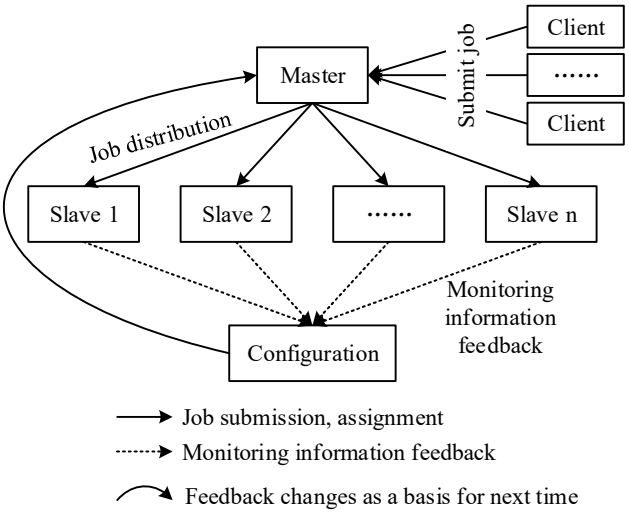


Fig. 2. Dynamic feedback process

After a period of cluster service, the load of each node will be different, in order to equalize the cluster load, it is necessary to periodically feedback the cluster load distribution, in order to provide a basis for judging whether the task allocation is reasonable or not. The feedback and judgment process also needs to consume resources, so it is not possible to make each node load the same at all times. T as a period of time, each T time interval to query, feedback, modify the load situation can be. Feedback content is mainly the load information of each node in the cluster, such as node $i(i=1,2...n)$'s CPU usage (represented by $L(CPU_i)$), I/O usage (represented by $L(I/O_i)$), memory usage (represented by $L(M_i)$), and network usage (represented by $L(N_i)$), which individually cannot completely represent the load situation of node i, but their synthesis can represent the real load situation, and using a function to synthesize them. Then the load situation of node N_i $L(N_i)$ can be expressed as shown in equation (4):

$$L(N_i)=\begin{bmatrix} r_1 & r_2 & r_3 & r_4 \end{bmatrix}\cdot\begin{bmatrix} L(CPU_i) \\ L(I/O_i) \\ L(M_i) \\ L(N_i) \end{bmatrix}, i=1,2,...n, \text{ and } \sum r=1 \tag{4}$$

Comparison of the differences between the processing power of cluster nodes makes the load per unit of processing power essentially semi-balanced. The processing capacity of a node can be measured from several aspects such as CPU type and number, memory capacity, disk I/O, network rate, etc. Again individual metrics cannot replace the processing capacity of a node, but as a whole, the processing capacity $C(N_i)$ of a i node (N_i) can be expressed as equation (5):

$$C(N_i)=\begin{bmatrix} r_1' & r_2' & r_3' & r_4' \end{bmatrix}\cdot\begin{bmatrix} \mathrm{m}C(CPU_i) \\ C(I/O_i) \\ C(M_i) \\ C(N_i) \end{bmatrix} \text{among } i=1,2,...n \text{ and } \sum r=1 \tag{5}$$

In order to represent the difference between the impact of the various parts of the node on the overall load of the node introduced parameters r', m represents the number of CPUs (N_i),

Equation (5) takes into account the differences in the indicators within each node, such as the FTP service on the CPU and the network and memory on the node is more prominent, in the parameter setting can be appropriately increased with the corresponding r, for example, when the $m=2$ when the $r=\{0.15,0.2,0.3,0.35\}$, the process of execution is still to be based on the feedback load distribution of the parameters to be modified, so that the system load as much as possible balanced distribution among the nodes, this idea we call dynamic feedback load balancing.

3.2. Relevant definitions

Definition 1: Considering that the processing capacity of nodes in the cluster may vary with the introduction of the fluctuation factor X, the node fluctuation factor of each node in the cluster is represented by X_i, $C(N_i)$ size can be calculated according to the previous section of Equation (5), X the larger represents the node's comprehensive processing capacity, of course, the actual scheduling can be used to deal with the relative size of the processing capacity:

$$X_i = \frac{C(N_i)}{\min\{C(N_1), C(N_2), ..., C(N_n)\}} \tag{6}$$

where $\min\{C(N_1), C(N_2), ..., C(N_n)\}$ denotes the processing power of the node with the least processing power among the n nodes, i.e., the X of the node with the least processing power is 1.

Definition 2: The total cluster load case is $L(T)$ and the unit load of the cluster is $L_{avg}(T)$ then:

$$L_{avg}(T) = \frac{L(T)}{\sum X_i} \tag{7}$$

$$L(T) = \sum L_i(T) \tag{8}$$

Definition 3: Dynamic prioritization is assigned between jobs with the same priority under a specific priority level, expressed as shown in Equation (9):

$$Priority = \frac{T_{execute} + T_{wait}}{T_{execute}} \tag{9}$$

After assigning dynamic priority to a job, the dynamic priority of the job will increase with the increase of the waiting time, which can avoid small jobs waiting for a long time and reduce the average response time of the job.

Definition 4: The cluster load balancing degree is judged as LB, LB is calculated as in equation (10), and the cluster load is considered to be balanced when $LB < 5\%$:

$$LB = \frac{\sigma}{L_{avg}(T)} < 5\% \tag{10}$$

$$\sigma^2 = \sum \left\{ \frac{L_i(T)}{X_i} - L_{avg}(T) \right\}^2 \tag{11}$$

Equation σ is the standard deviation of the load of each node, measure the cluster load balancing situation can be calculated on the LB, if it meets the judging criteria, the load balancing can schedule the task to the current node.

4. Numerical Validation of DFLB Load Balancing Technique

4.1. Experimental environment construction

In order to complete the algorithm validation, a distributed public network digital cluster system test environment is built, and the Floodlight controllers are installed on different physical devices, and the operating system is selected to be Ubantu16.04.This experiment is deployed on Mininet and Floodlight controllers, and a customized topology is built by running the customized topology on Mininet and connecting it to the controller Floodlight to set up the experimental environment and simulate the communication between the client and the data center. The path forwarding policy for new data flows is computed in Floodlight and delivered to each switch in the form of a flow table. Open vSwitch switches supporting the OpenFlow protocol are used, Iperf is used to simulate network traffic, and Ubuntu 16.04 is used as the operating system, installed on a Linux computer with an Intel i7-9600KF 3.7GHz CPU and 16GB RAM.

Mininet has three types of command parameters, namely, network construction startup parameters, internal interaction commands, and external runtime parameters. Network construction startup parameters can be used to set topology, switch type, and link attributes, etc.; internal interaction commands can be used to interact with the virtual nodes, such as joining or removing nodes, and the external runtime parameters are mainly used to control Mininet's runtime environment, for example, to Setting log output, etc. These command parameters can set the topology structure, switch type and link attributes, etc., which is convenient for users to modify and configure the topology.

Floodlight is a Java-based SDN controller developed with the main function of providing programmable control and management of SDN networks using the OpenFlow protocol. In practice, users can build and deploy SDN network applications on Floodlight to flexibly customize the network topology, rules and policies to meet their specific goals and needs, such as load balancing and flow control.

After running the Floodlight controller, enter sudo mn --controller=remote at the command line to create a simple SDN topology network consisting of one switch and two hosts connected to the Floodlight default address and port 127.0.0.1:6653.

Iperf is a TCP/IP and UDP/IP based network performance testing tool that measures network bandwidth and network quality by using command line mode.

Cbench is a commonly used tool for performance testing of SDN controllers, which simulates a certain number of hosts and switches connected to the controller and tests the performance of the controller by sending Packet-in messages to the controller. The test is performed by connecting to each controller using the Cbench software in latency mode, which sends a specified number of Packet-in messages and waits for the controller to return a matching Packet-out message. In latency mode Cbench can calculate the response delay for each Packet-in message.

After several tests, the rate of Packet-in messages is constantly changed, the change of response delay under different rates is compared, and the rate that makes the response delay increase significantly is the desired rate. After testing the Floodlight controller used in this paper, the maximum processing capacity of Packet-in message per unit time is about 7000.

4.2. Distributed system experimental topology description

A controller cluster consisting of four FloodLight controllers is deployed in the control layer, and a total of 16 OpenvSwitch switches are used in the data layer, with each controller managing four switches, i.e., the controller's role is Master, and its role relative to the remaining switches is Slave.

The properties of each controller in the cluster are set in the configuration file server.properties as shown in Table 1.

Table 1. Properties of cluster controllers

Name	The controller C1	The controller C2	The controller C3	The controller C4
NodeId	1	2	3	4
IP address	192.168.58.4	192.168.58.5	192.168.58.6	192.168.58.7
Port number	55551	55552	55552	55554

One host is connected behind each switch, and the hping3 utility is run on the host to send a specified number of ARP packets to hosts connected to other switches. Since there is no flow table entry on the local switch indicating the flow to the other switch, there is no flow table matching the ARP packet after it arrives at the switch, so the switch encapsulates the ARP packet as a Packet-in request and sends it to the controller, which replies to the switch with a Packet-out message after route recalculation to be used to match the ARP packet. Thus the effect of generating Packet-in messages is realized.

4.3. Load Balancing System Performance Testing

4.3.1. Response Latency Testing. The size of the response delay is an important indicator for evaluating the processing performance of the controller. The smaller the response delay, the faster the processing speed of the controller, proving its better performance.

The response latency test is shown in Fig. 3. When the Packet-in rate is lower than 9000packet/s (the maximum Packet-in message processing capacity of the controller), there is basically no difference in the response latency between the DFLB algorithm of this paper and the static deployment method, which is due to the fact that there is not yet an overloaded controller, and the load balancing of the control plane. When the rate of Packet-in messages is continuously increased, the response latency of the static deployment method starts to be gradually higher than that of the DFLB algorithm due to the lack of load balancing mechanism. For DFLB, the switch migration operation is performed so that the load of the overloaded controllers is transferred to the lightly loaded controllers, which balances the load of the controller cluster. Compared with the static deployment method, the DFLB algorithm can reduce the response latency by about 20%. However, when the rate of packet-in messages is greater than 18000 packets/s, the response latency of the controllers for both static deployment and DFLB algorithms increases rapidly because it is no longer possible to equalize the load of the severely overloaded control plane through switch migration.

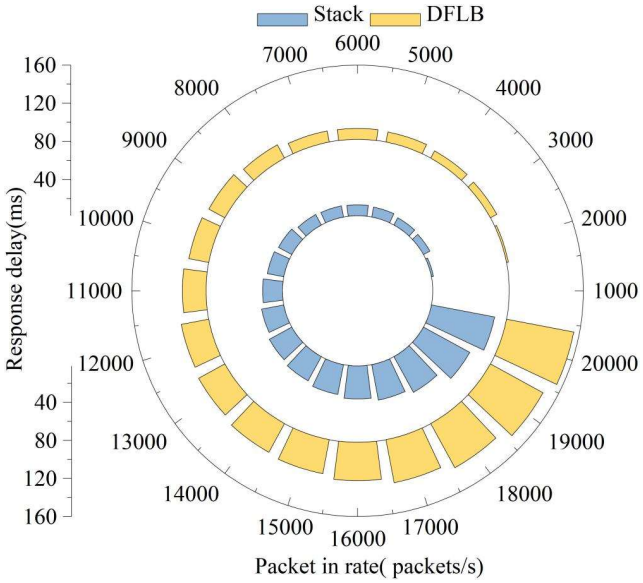


Fig. 3. Comparison of response delay

4.3.2. Load Balancing Effectiveness Test. In order to test the load balancing effectiveness of the designed DFLB algorithm, the following three algorithms are run on the controllers in the system respectively:

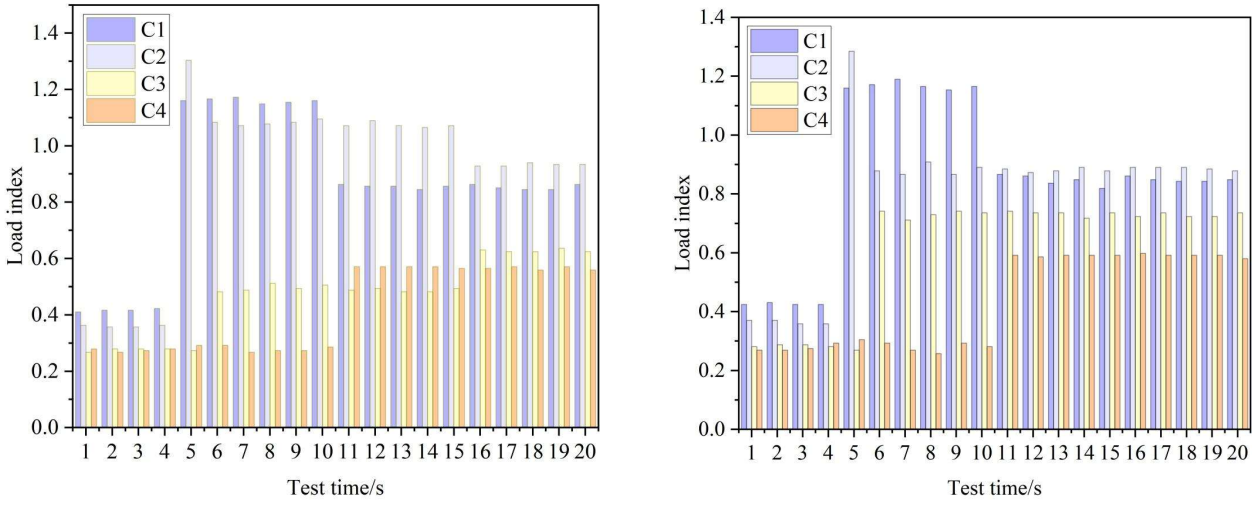
1) DALB algorithm
2) PALB algorithm
3) DFLB algorithm designed in this paper.

The controller of the DFLB algorithm designed in this paper can migrate multiple switches at the same time for multiple overloaded controllers in one cycle with high migration efficiency, while the DALB and PALB algorithms can only deal with one overloaded controller at a time, which achieves a long cycle of load balancing and low migration efficiency.

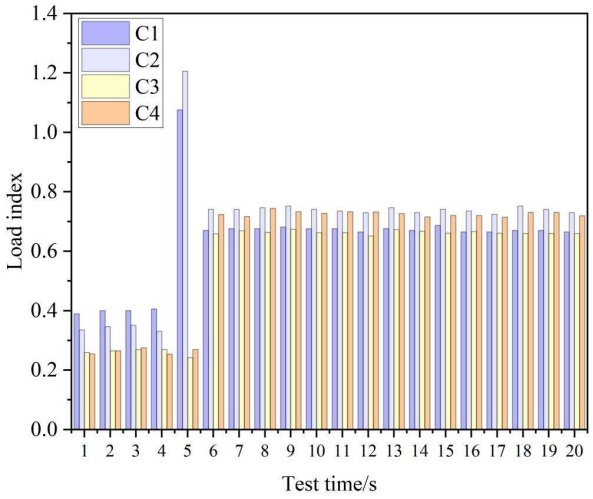
When the experiment ends, the load variance of the controller clusters under the three algorithms is calculated, and the DFLB algorithm in this paper makes the load more balanced among the controllers.

The migration overhead in the load balancing process is mainly reflected in the number of migrated switches.The DFLB algorithm selects the switch that reduces the cluster variance the most under the overload controller as the migrated switch, so it can quickly reduce the load variance of the cluster to a reasonable interval and reduce the number of migrated switches. Therefore, it can be concluded that the DFLB algorithm designed in this paper meets the requirements of efficiency, effectiveness and stability, and has a relatively good load balancing effect.

The throughput changes of the system under the three algorithms are shown in Fig. 4, the sudden increase of Packet-in messages in the 5ths leads to a serious overload of the controllers C1 and C2, and the processing performance of the controllers decreases, which causes packet loss, thus resulting in a decrease in the average throughput.The DALB algorithm has a slower growth of throughput due to the average load-balancing effect.The PALB algorithm consumes a lot of time due to its measurement and computation of round-trip delays. The PALB algorithm consumes more processing resources for the measurement and computation of round-trip delay, and the performance of the control plane is affected, which ultimately leads to a reduction in throughput. The DFLB algorithm designed in this paper can quickly complete the migration of switches, so that the load between the controllers is more balanced, and at the same time, because the system adopts the controller selection strategy to save the computational resources of the controller.



(a)DALB (b)PALB



(c)DFLB

Fig. 4. Controllers load of three algorithm

The load balancing degree statistics of all algorithms are shown in Table 2. Within 1 min, the average load balancing degree of the simple polling algorithm and the minimum number of connections algorithm is more than 5, and the maximum load balancing degree is 7.22. The load balancing degree of the DALB and PALB algorithms is lower than that of the traditional algorithms. Deviation is smaller and the load balancing degree is more stable, which reflects that the real-time scheduling and allocation of resources are more reasonable, and the more reasonable the resource allocation, the higher the overall resource utilization of the server.

The user simulator sends a login request every 60ms and 1000 login requests to the load balanced server in 1min. During the period of 1000 user logins, the logged in user initiates the call randomly.

The DFLB based algorithm is less than the traditional algorithm, DALB and PALB algorithms in terms of longest login delay and average login delay. Since the actual load of the server is most balanced when load balancing with the DFLB algorithm, the resource allocation is reasonable, and the server response is more timely, which makes the user login delay under this algorithm lower and the experience better with the same server resources.

Table 2. Load balancing and delay of three kinds of algorithms

Algorithm	Maximum equilibrium	Mean equilibrium	Mean delay	Shortest delay	Maximum delay
Simple polling	7.22	5.14	6.154	0.180	17.022
Minimum connection number	6.31	4.21	6.235	0.180	16.425
DALB	5.62	4.23	5.552	0.180	15.826
PALB	5.12	3.82	5.846	0.180	14.321
DFLB	3.74	2.26	5.232	0.180	12.028

4.3.3. Throughput testing. The throughput changes of the system under the three algorithms are shown in Fig. 5, and the average throughput decreases in the 6ths due to the sudden increase of Packet-in messages, which causes the controllers C1 and C2 to be severely overloaded, and the controller's processing performance decreases, causing packet loss, thus resulting in a decrease in the average throughput.The DALB algorithm has a slower increase in throughput due to the average load balancing effect.The PALB algorithm consumes more processing resources due to its measurement and computation of round-trip delay, which ultimately leads to a decrease in throughput. The PALB algorithm consumes more processing resources for the measurement and

computation of round-trip delay, and the performance of the control plane is affected, which ultimately leads to a reduction in throughput. The DFLB algorithm designed in this paper can quickly complete the migration of switches and make the load between controllers more balanced, and at the same time, the system adopts the controller selection policy to save the computational resources of controllers. Eventually, during throughput recovery, the average throughput of the system is improved by about 10% and 6% compared to the other two algorithms, and the throughput grows faster.

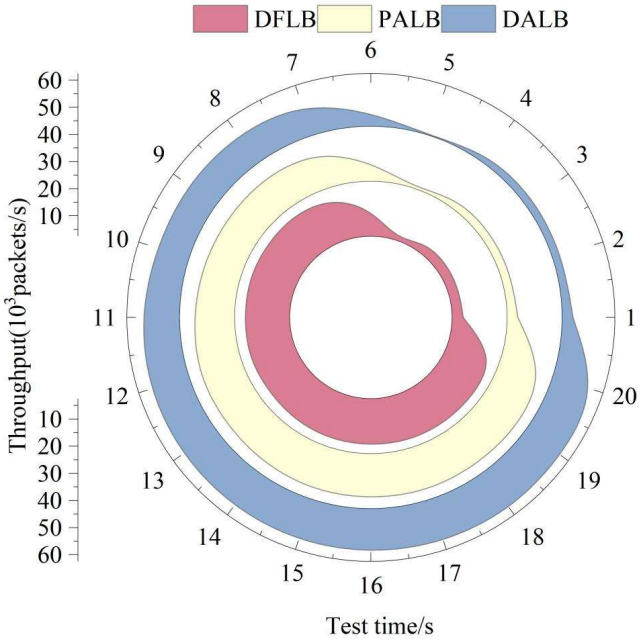


Fig. 5. Comparison of throughput

4.3.4. Number of concurrent connections. When evaluating load balancing strategies, we mainly examine the actual number of concurrent connections to analyze the performance of the dynamic feedback-based load balancing algorithm, and use the IP Hash algorithm that comes with Nginx as a comparison. Among them, the core idea of IP Hash algorithm is to perform hash mapping based on the IP address of the source of the service request, and the result of the hash operation is used as the basis for selecting the server node that actually responds to the service request, and then assigns the service request to the corresponding server node, and the number of concurrent connections is shown in Table 3. As the number of concurrent connections rises, the access rate of the DFLB algorithm far exceeds that of the IP Hash algorithm.

Table 3. Actual concurrent connection number comparison

Concurrent number	IP Hash The actual number of concurrent connections	Proportion	IP Hash The actual number of concurrent connections	Proportion
200	198	99.00%	198	99.00%
400	399	99.75%	399	99.75%
600	598	99.67%	598	99.67%
800	800	100.00%	800	100.00%
1000	687	68.70%	994	99.40%
1200	564	47.00%	758	63.17%
1400	524	37.43%	673	48.07%

The actual concurrent connections test results are shown in Figure 6, when the number of concurrent connections is low (800 or less), the actual number of concurrent connections is basically the same as the number of concurrent request connections, with a small amount of occasional loss (IP Hash algorithm's loss rate is less than 1%, which is able to maintain the basic normal request access performance. When the number of concurrent connections continues to increase and reaches 1000, the IP Hash algorithm shows significant concurrent request connection loss, with a loss rate of more than 30%, while the loss rate of the dynamic feedback-based load balancing algorithm is only 0.6%. In comparison, the latter has better service request access performance.

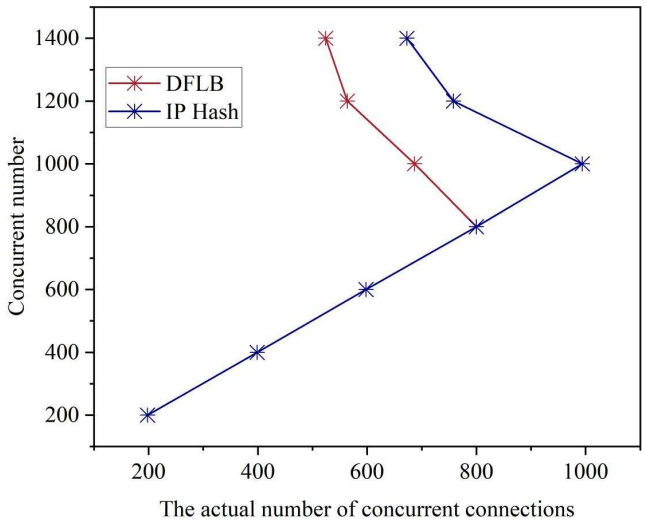


Fig. 6. The actual concurrent connection number test results

5. Conclusion

In this paper, Dynamic Feedback Load Balancing (DFLB) algorithm is proposed based on the concept of numerical analysis. The dynamic feedback loop process of collection-feedback-utilization-collection is utilized to equalize the load distribution among the nodes of the system. Through numerical validation experiments, it can be seen that: the response delay of the system with the application of the DFLB algorithm, is smaller, and the processing speed of the controller is faster. The DFLB algorithm can quickly complete the migration of switches, reducing the waste of the controller's computational resources, so that the system load is more balanced, and the goal of high efficiency, effectiveness, and stability is realized. Compared with the traditional algorithms, DALB and PALB algorithms, the DFLB algorithm is better in both the longest login delay and the average login delay. The dynamic feedback load balancing algorithm based on numerical analysis is better than the IP Hash algorithm in solving high data concurrency.

References

[1] Wu, J. (2017). Distributed system design. CRC press.

[2] Assiri, B., & Sheneamer, A. (2025). Fault tolerance in distributed systems using deep learning approaches. PloS one, 20(1), e0310657.

[3] Van Steen, M., & Tanenbaum, A. S. (2016). A brief introduction to distributed systems. Computing, 98, 967-1009.

[4] Hanmer, R., Jagadeesan, L., Mendiratta, V., & Zhang, H. (2018, October). Friend or foe: Strong consistency vs. overload in high-availability distributed systems and SDN. In 2018 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW) (pp. 59-64). IEEE.

[5] John, N. P., & Bindu, V. R. (2020). Prediction Mechanism–A Novel Approach for OverLoad Management in a Distributed Computing System. Procedia Computer Science, 171, 2097-2104.

[6] Ghomi, E. J., Rahmani, A. M., & Qader, N. N. (2017). Load-balancing algorithms in cloud computing: A survey. Journal of Network and Computer Applications, 88, 50-71.

[7] Kumar, P., & Kumar, R. (2019). Issues and challenges of load balancing techniques in cloud computing: A survey. ACM computing surveys (CSUR), 51(6), 1-35.

[8] Mishra, S. K., Sahoo, B., & Parida, P. P. (2020). Load balancing in cloud computing: a big picture. Journal of King Saud University-Computer and Information Sciences, 32(2), 149-158.

[9] Gupta, Y. (2021). Novel distributed load balancing algorithms in cloud storage. Expert Systems with Applications, 186, 115713.

[10] Zakutynskyi, I., Rabodzei, I., Burmakin, S., Kalishuk, O., & Nebylytsia, V. (2023). IMPROVING A PROCEDURE OF LOAD BALANCING IN DISTRIBUTED IOT SYSTEMS. Eastern-European Journal of Enterprise Technologies, 125(2).

[11] El-Sayed, H., Sankar, S., Prasad, M., Puthal, D., Gupta, A., Mohanty, M., & Lin, C. T. (2017). Edge of things: The big picture on the integration of edge, IoT and the cloud in a distributed computing environment. ieee access, 6, 1706-1717.

[12] Mallikarjuna, B., & Reddy, D. A. K. (2019). The role of load balancing algorithms in next generation of cloud computing. Control Syst, 11, 20.

[13] Pourghebleh, B., & Hayyolalam, V. (2020). A comprehensive and systematic review of the load balancing mechanisms in the Internet of Things. Cluster Computing, 23(2), 641-661.

[14] Puthal, D., Ranjan, R., Nanda, A., Nanda, P., Jayaraman, P. P., & Zomaya, A. Y. (2019). Secure authentication and load balancing of distributed edge datacenters. Journal of Parallel and Distributed Computing, 124, 60-69.

[15] Kashani, M. H., & Mahdipour, E. (2022). Load balancing algorithms in fog computing. IEEE Transactions on Services Computing, 16(2), 1505-1521.

[16] Chen, S. L., Chen, Y. Y., & Kuo, S. H. (2017). CLB: A novel load balancing architecture and algorithm for cloud services. Computers & Electrical Engineering, 58, 154-160.

[17] Mishra, K., & Majhi, S. (2020). A state-of-art on cloud load balancing algorithms. International Journal of computing and digital systems, 9(2), 201-220.

[18] Xu, M., Tian, W., & Buyya, R. (2017). A survey on load balancing algorithms for virtual machines placement in cloud computing. Concurrency and Computation: Practice and Experience, 29(12), e4123.

[19] Ivanisenko, I., & Volk, M. (2017, September). Simulation methods for load balancing in distributed computing. In 2017 IEEE East-West Design & Test Symposium (EWDTS) (pp. 1-6). IEEE.

[20] Zhou, G. (2022). Implementation of Dynamic Load Balancing in Distributed System Based on Improved Algorithm. Mobile Information Systems, 2022(1), 1735098.

[21] Ding, S., Chen, C., Xin, B., & Pardalos, P. M. (2018). A bi-objective load balancing model in a distributed simulation system using NSGA-II and MOPSO approaches. Applied soft computing, 63, 249-267.

[22] Mirtaheri, S. L., & Grandinetti, L. (2017). Dynamic load balancing in distributed exascale computing systems. Cluster Computing, 20, 3677-3689.

[23] Kishor, A., Niyogi, R., & Veeravalli, B. (2020). Fairness-aware mechanism for load balancing in distributed systems. IEEE Transactions on Services Computing, 15(4), 2275-2288.

[24] Jiang, Y. (2015). A survey of task allocation and load balancing in distributed systems. IEEE Transactions on Parallel and Distributed Systems, 27(2), 585-599.

[25] Gopi, A. P., Narayana, V. L., & Kumar, N. A. (2018). Dynamic load balancing for client server assignment in distributed system using genetical gorithm. Ingénierie des systèmes d'information, 23(6).

[26] Shafiq, D. A., Jhanjhi, N. Z., & Abdullah, A. (2022). Load balancing techniques in cloud computing environment: A review. Journal of King Saud University-Computer and Information Sciences, 34(7), 3910-3933.

[27] Mirtaheri, S. L., Fatemi, S. A., & Grandinetti, L. (2017). Adaptive load balancing dashboard in dynamic distributed systems. Supercomputing Frontiers and Innovations, 4(4), 34-49.

[28] Kirichenko, L., Ivanisenko, I., & Radivilova, T. (2016, February). Dynamic load balancing algorithm of distributed systems. In 2016 13th International Conference on Modern Problems of Radio Engineering, Telecommunications and Computer Science (TCSET) (pp. 515-518). IEEE.

[29] Ivanisenko, I. N., & Radivilova, T. A. (2015, October). Survey of major load balancing algorithms in distributed system. In 2015 information technologies in innovation business conference (ITIB) (pp. 89-92). IEEE.