Scholar Publishing Group
Distributed Processing System
https://doi.org/10.38007/DPS.2022.030308
ISSN 2790-0916 Vol. 3, Issue 3: 61-71

Distributed System Load Balancing Strategy for Converged Workstations

Rashid Gullun[*]

University of Sulaimani, Iraq

[*]*corresponding author*

Keywords: Converged Workstations, Distributed Systems, Load Balancing Strategies, Load Balancing Algorithms

Abstract: In the era of cloud computing and big data, distributed block storage system has become more and more important with its unique advantages. Load balancing (LB) is an important feature of distributed block storage systems, and it is also one of the hotspots of current distributed block storage research. Based on the existing LB algorithms, researching new and more effective LB techniques is very effective in improving the stability of the system. The main purpose of this paper is to study the LB strategy in the distributed system (DS) of converged workstations. Based on the traditional LB strategy, this paper selects multiple evaluation indicators and establishes a corresponding evaluation function to improve the accuracy of judging the actual load of the node server. According to the situation of different hardware parts, the hierarchical standard of the load state is formulated, and the instantaneous stress test of the improved LB algorithm and the weighted minimum connection algorithm is carried out. The complexity of a LB algorithm. Experiments show that with the increase of the number of request connections, the response time of the system under the two algorithms is longer, but the response time of the improved LB scheduling algorithm has a smaller increase. That is to say, the improved LB algorithm is better than the weighted least connection algorithm in handling the instantaneous pressure of the system.

1. Introduction

Many DSs mainly rely on algorithms related to data distribution to evenly distribute data on each node to achieve a certain LB effect. The performance of the DS LB algorithm depends to a large extent on the selection of evaluation indicators. A good evaluation index can enable each node to work under normal load status, without making some nodes overloaded or too low, so that the system's operating status and file processing performance can be optimized. Improper selection of evaluation indicators will cause system load imbalance [1-2].

In their research on LB, Alam et al. proposed a Resource-Aware LB (REAL) model [3] for a batch of independent tasks with a centralized load balancer to make the solution suitable for practical applications with migration costs. Heterogeneous distributed environment to maximize load level balancing considering bandwidth requirements of migration tasks. To facilitate LB, Aktas et al. store data in a redundant manner in DSs [4]. We evaluate the LB performance of the storage scheme, where the loads served to objects are uniformly sampled at random from all load vectors with fixed cumulative values. The LB of the system is represented by the maximum spacing between ordered statistics of uniform random variables or the maximum value of consecutive spacings.

Traditional dynamic LB algorithms use a single load evaluation index [5-6]. The advantage of a single evaluation index system is that it is easy to code and implement, but there are certain defects. For example, when there are data blocks on a node server that are allocated for hotspot access, but its own disk space usage is relatively low, the load situation obtained by using only the disk space usage will be very different from the actual situation [7-8]. Through the above analysis, a single evaluation index cannot truly reflect the real load of each node in the DS. Based on the traditional LB strategy, this system will select multiple evaluation indicators and establish corresponding evaluation functions to improve the accuracy of judging the actual load of the node server.

2. Design Research

2.1. Defects of Traditional Algorithms

Traditional algorithms have the following problems:

(1) The evaluation index of node load is single. Different DSs deal with different scenarios [9-10]. Therefore, the factors affecting the load of distributed nodes in the cluster are also different, and it is difficult to directly evaluate the load of nodes through a certain index. Therefore, the single evaluation index system of the traditional LB algorithm has certain problems when applied to the actual DS.

(2) The LB threshold is single, which causes the state of each distributed node in the cluster to change in a short time, resulting in system instability.

(3) The load situation of all nodes in the DS is completely collected by the scheduling node at regular intervals. In addition, it is responsible for the scheduling and distribution of tasks, and the burden is heavy.

2.2. Algorithm Improvement Content

A well-designed LB scheduling algorithm should satisfy the following three conditions:

(1) Establish a reasonable node load evaluation index and evaluation method, which can truly reflect the real-time load of each distributed node in the cluster;

(2) Each distributed node actively calculates its own real-time load, and reports the load information to the scheduling node at an appropriate time to reduce the processing burden of the scheduling node.

(3) Minimize the complexity of the algorithm.

Aiming at the shortcomings of the traditional LB scheduling algorithm, combined with the characteristics of the management system of transmission and transformation projects [11-12], the algorithm is improved from the following aspects.

(1) Improve the calculation method of distributed node load. Different from the single evaluation index of the traditional algorithm, the evaluation function of the distributed node load is established;

(2) The LB threshold strategy is improved, and two thresholds are set to avoid the problem of frequent node state changes under the original single threshold strategy, so as to improve the system stability;

(3) Design the corresponding load migration strategy [13-14].

2.3. Selection of Load Evaluation Indicators

The performance of the DS LB algorithm depends to a large extent on the selection of evaluation indicators [15-16]. A good evaluation index can enable each node to work under normal load status, without making some nodes overloaded or too low, so that the system's operating status and file processing performance can be optimized. Improper selection of evaluation indicators will cause system load imbalance.

Through the research of multiple traditional LB algorithms, the evaluation indicators of each algorithm are summarized and analyzed, and it is found that the following indicators can be used as candidates [17-18].

(1) CPU utilization
(2) Memory usage
(3) CPU temperature
(4) Network speed
(5) System response time
(6) Number of running tasks
(7) Disk I/O rate

2.4. Quantitative Description of Load

It can be determined that the file access load is positively related to the number of accesses and the access size, and for the chronological order of file access, the load of the more recent access is higher. That is, the load of the file at a certain time is related to the previous access, so introduce:

$$L_0(f)=0 \tag{1}$$

$$L_i(f)=\lambda\times S_i+(1-\lambda)\times L_{i-1}(f),\lambda=\frac{\Delta t_i}{\Delta t_i-\Delta t_{i-1}} \tag{2}$$

where f represents the accessed file; Li(f) represents the load value of the file after the ith access, and the 0th access is naturally 0; Si represents the file access size; Δti represents the file load from zero to the ith access interval time; and its calculated λ is used to represent the impact of file access time sequence.

Considering the time the file is occupied, and considering the system performance problem, it is impossible to update the load value all the time, so a certain detection time is selected to update the load value, so the formula is revised as:

$$L_i(f)=\lambda\times\sum\nolimits_{m=1}^{n}(S_m\times U_m)/T+(1-\lambda)\times L_{i-1}(f),\lambda=\frac{\Delta t_i}{\Delta t_i-\Delta t_{i-1}} \tag{3}$$

The corresponding meanings of some variables also need to be changed. At this time, i represents the ith monitoring rather than the ith visit. T represents the monitoring time, Sm and Um represent the m-th access size and access time respectively within the monitoring time, and n represents the number of file accesses within the monitoring time. The average value of the loads obtained during this period is the load at the end of time T.

3. Experimental Study

3.1. Selection of load evaluation function

After the evaluation index is determined, the corresponding evaluation function needs to be constructed. The function of the evaluation function is to calculate the load of each node in the distributed cluster. The idea of determining the evaluation function is to first select several indicators for evaluation, then assign a certain weight coefficient to each indicator, and finally combine the indicators and their corresponding weights in a certain form. The evaluation function usually has the following forms.

(1) Index addition

Each evaluation index has a certain influence on the total evaluation value, but there is no essential difference in the degree of influence. Because the value ranges of each indicator are independent of each other, the effect of mutual compensation can be achieved. The general form of the index addition evaluation function is as follows.

$$F(X_j) = \sum \omega_{ji} X_{ji} \tag{4}$$

(2) Multiplication of indicators

Similar to the addition of indicators, in the evaluation function form of multiplication of indicators, the evaluation indicators can also compensate each other, but the compensation effect is generally weak. Its general form is as follows.

$$F(X_j) = \prod_{i=1}^{n} (X_{ji})^{\omega_{ji}} \tag{5}$$

(3) Index substitution

It is applicable to the situation where each evaluation index compensates each other. Its general form is as follows.

$$F(X_j) = 1 - \prod_{i=1}^{n} (1 - X_{ji}) \tag{6}$$

In the previous research on the load calculation model of DSs, the linear weighting method has been widely studied and has been widely used in the actual file system. It can be seen that the calculation model established by the linear weighting method can be more Accurately describe the load on the file server. Therefore, on the basis of the linear weighting method, it is improved to establish the calculation model of the load.

The basic idea of linear weighting is that a set of non-negative numbers ωi (i=1,...,m) corresponding to each influencing factor is given to represent the importance of the corresponding influencing factor to the final goal. First multiply each weight coefficient with each corresponding target to get ωifi(i=1,...,m), and then sum it up to get the following expression, which is the required evaluation function:

$$u(f) = \sum_{i=1}^{m} \omega_i f_i \tag{7}$$

From this, it can be concluded that the load evaluation function of each distributed node in the distributed cluster is:

$$L = K_1 * L(cpu) + K_2 * L(mem) + K_3 * L(io) \qquad (8)$$

L(cpu) is the CPU usage, L(mem) is the memory usage, and L(io) is the disk I/O rate. K1 is the weight coefficient of CPU utilization, K2 is the weight coefficient of memory utilization, and K3 is the weight coefficient of disk I/O rate.

In the system, memory usage, disk I/O rate, and CPU utilization have different effects on the load of each distributed node in the cluster. According to the degree of influence of each factor on the load, a certain weight coefficient is respectively assigned, and the evaluation function of the node load is obtained according to formula 8.

3.2. LB Subsystem

The cluster of the system includes three types of nodes, one is the scheduling node, which is responsible for the collection and task scheduling of the load status of other nodes in the cluster, the other is the processing node, which is specifically responsible for the processing of files, and the other is the storage node. Node, responsible for the storage of project data and files. The structure of the LB subsystem is shown in Figure 1, which mainly includes load migration, location query, node communication, and load information storage modules.

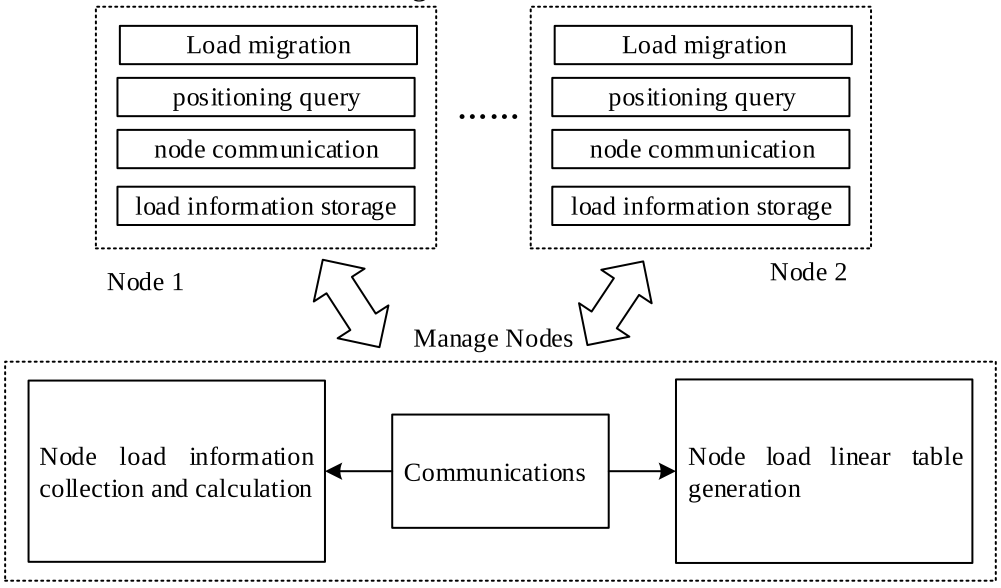

Figure 1. LB subsystem block diagram

Load collection module: responsible for collecting real-time load reported from cluster distributed processing nodes.

Positioning query module: When the load needs to be migrated, the linear load table is searched by the polling strategy to find the migration target node.

Task migration module: When the load of a node is not good, it is responsible for the migration of the node load.

Load information storage module: establish a load linear table to store the load information of each distributed node in the cluster.

Communication module: responsible for issuing instructions, receiving instructions, and parsing instructions for each distributed node in the cluster.

3.3. Overview of the Integrated Program

In view of the problems faced by the system, combined with the design process of several stages and comprehensively considering the advantages of each stage, the relationship between each module in the comprehensive scheme as shown in Figure 2 can be obtained.

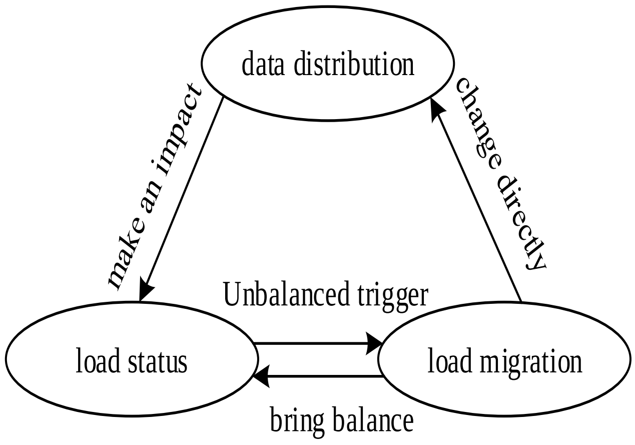

Figure 2. Interrelationships between modules

According to the above-mentioned scheme design, the data is firstly distributed among the nodes as evenly as possible, then according to the real-time load determination scheme, the real-time access load of the nodes is counted, the load status of the nodes is determined, and corresponding records are made for future use. stage is used. Finally, according to the load status and data recorded by statistics, the load is adjusted according to the load migration scheme, so that the system can reach a state of load balance. The previous scheme chose to describe the data migration scheme first, and then the LB scheme, in order to highlight the criticality of load determination.

The relationship between the three key modules is shown in Figure 2, which clearly shows the interaction and influence between the various modules. The data distribution scheme will affect the load state (the load is mainly related to the access to the stored data), and the unbalanced load state will trigger the load migration process. The load migration process in turn makes the load more balanced. At the same time, the load migration process actually affects the storage of data on the node. This is the case with the interaction between the entire scheme and the internal modules.

Finally, the redundant storage copy can be used for further LB. When the read request arrives, the read request selects the replica based on the node with the lowest load among the three replicas as the node that responds to the read request, which can avoid further increase in the load of the high-load node and further balance the system load. Because the data consistency required by different systems is different, in order to avoid data consistency problems, write requests are not differentiated.

4. Experiment Analysis

4.1. LB Threshold Policy Design

If at a certain moment, a node in the cluster is overloaded, the scheduling node migrates the tasks on the node to other lightly loaded nodes, so that the state of the node changes and is divided into lightly loaded nodes. point queue. However, if a new task is assigned to the node at this moment, the state of the node may change again and become overloaded. The task needs to be migrated again according to the scheduling strategy, which will affect the stability of the system.

According to the situation of different hardware parts, the hierarchical standard of load state is formulated, as shown in Table 1. The data in the table are all measured in percentage.

Table 1. Load status tiering criteria

STATE	CPU	MEM	I/O
Light	0~14	0~16	0~19
Moderate	14~37	16~40	19~38
Heavy	37~100	40~100	38~100

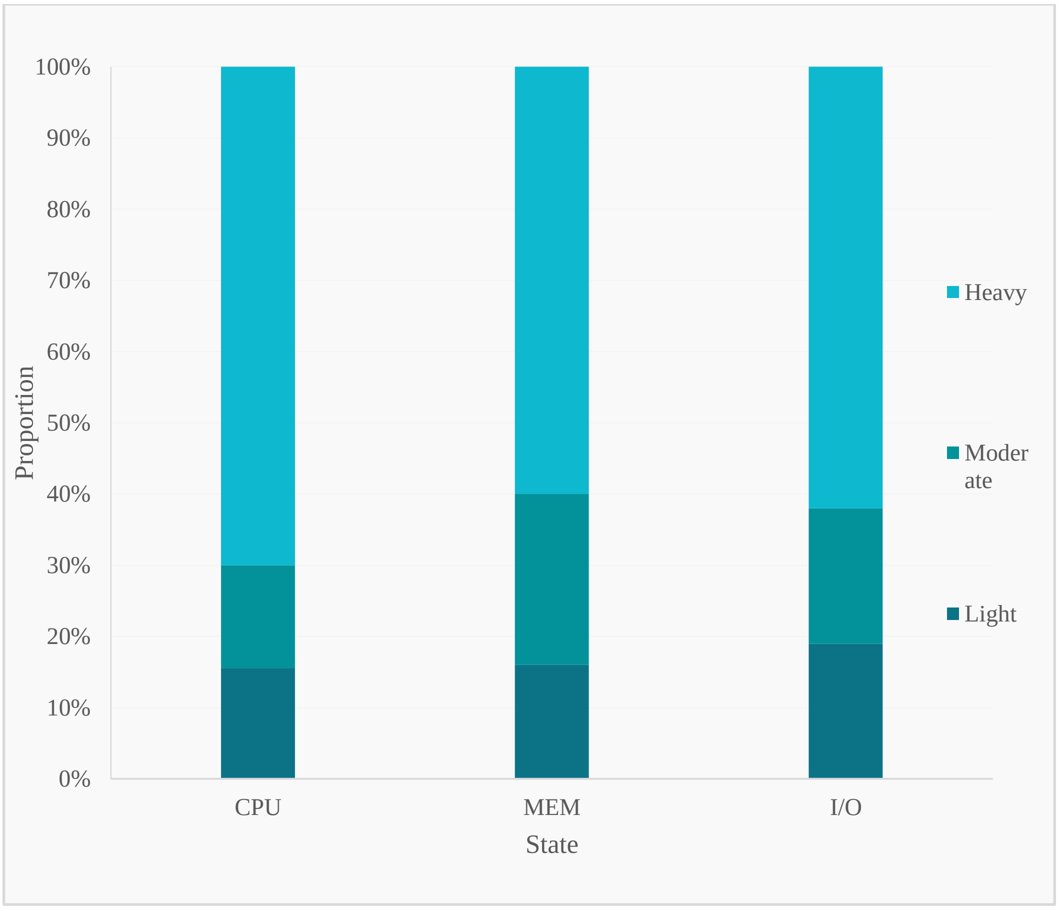

Figure 3. Load state hierarchical standard analysis

The upper bound of CPU utilization is 37%, and the lower bound is 14%; the upper bound of memory utilization is 40%, and the lower bound is 16%; the upper bound of hard disk I/O rate is 38%, and the lower bound is 19%. Then, according to the previously calculated weight coefficients of CPU utilization, memory utilization, and disk I/O rate, the upper and lower boundary values of the node server load can be obtained. Since the data in the table is a unified standard formulated by the industry, the LB threshold derived from this should have sufficient accuracy, and the load status of the distributed nodes in the cluster can be reasonably judged and divided.

4.2. Test Results and Analysis

(1) Test results

The instantaneous stress test results of the improved LB algorithm and the weighted least connection algorithm are shown in the table, and the instantaneous stress test results under the two scheduling algorithms are also shown.

Table 2. Test results

	40	80	120	160	200	240	280	320	360	400	440	480
Improved scheduling algorithm	245	436	694	812	1026	1264	1485	1692	1814	2102	2368	2500
Weighted Least Connections Algorithm	226	419	694	825	1086	1314	1583	1786	2174	2500	2766	3345

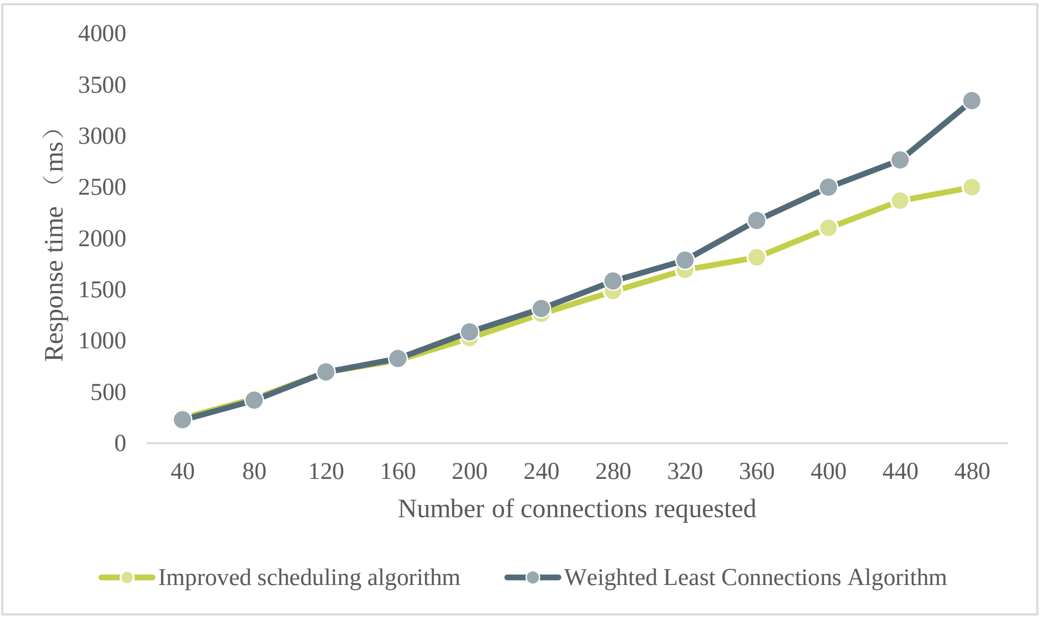

Figure 4. Analysis of test results

From the instantaneous stress test results, when the number of connections requested by the client is small, the response time of the weighted least connection algorithm is shorter than that of the improved LB scheduling algorithm. This is because the improved LB algorithm combines the characteristics of power transmission and transformation project management and takes into account many practical situations, so it is logically more complex and requires more system resources. When the number of requested connections is small, the improved LB algorithm cannot give full play to its advantages, so the response time is longer than the traditional algorithm.

However, as the number of subsequent request connections increases, the improved algorithm gradually shows its own advantages. It can be seen from the test result graph that when the number of requested connections is 160, the response time of the improved algorithm begins to be lower than the weighted minimum connection method. With the increase of the number of requested connections, the response time of the system under the two algorithms is longer, but the response time of the improved LB scheduling algorithm has a smaller increase.

Through the comparative test, the improved LB algorithm is better than the weighted minimum connection algorithm in handling the instantaneous pressure of the system.

(2) Analysis of simulation results

The complexity of the LB scheme based on round-robin, the LB scheme based on the minimum active number and the LB scheme based on resource prediction proposed in this paper under different number of tasks is compared. The total time taken to execute indicates the complexity of each LB algorithm. The comparison results are shown in Table 3.

Table 3. Complexity analysis of different LB algorithms

Number of tasks (pieces) / LB scheme	500	1000	1500	2000
Round-robin based LB scheme	7.1	13.6	20.1	26.4
LB Scheme Based on Minimum Active Number	10.3	20.3	29.0	38.6
LB Scheme Based on Resource Prediction	15.8	23.9	28.4	32.4

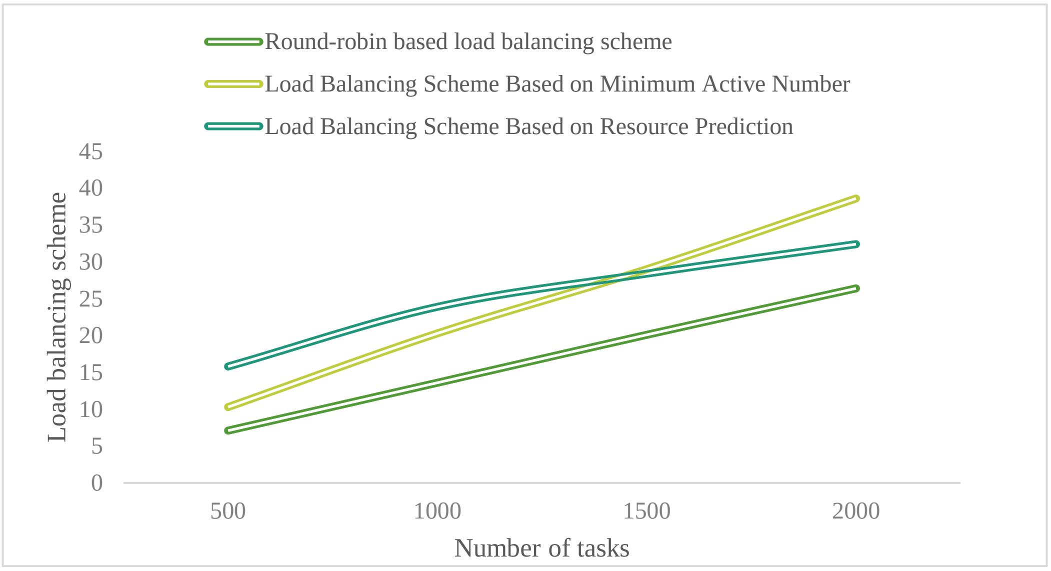

Figure 5. Complexity analysis diagram of different LB algorithms

As can be seen from the figure, as the number of tasks increases, the complexity of all three schemes increases. The LB scheme based on round-robin has a simple algorithm and a low degree of sensitivity to the number of tasks, so the complexity is linearly related to the number. The LB scheme based on the minimum active number needs to update the active number list of each fog server for the start and end of each task, so it is time-consuming. The scheme proposed in this paper is based on resource prediction. With the increase of the number of tasks, the enrichment of samples improves the accuracy of prediction, which greatly reduces the total execution time of the algorithm. Generally speaking, the complexity of the algorithm proposed in this paper is between the LB scheme based on round-robin and the LB scheme based on the minimum active number.

5. Conclusion

In the era of big data and the tide of cloud storage, many directions related to data are worthy of continuous research, and distributed storage still has a huge space for development. Research on various issues related to distributed storage will continue to deepen, and the LB problem of distributed storage still has great theoretical and research significance. Many distributed storage systems mainly rely on algorithms related to data distribution to evenly distribute data on each node for storage, so as to achieve a certain LB effect. Dynamic adjustments are made based on the specific load results of the nodes collected after running, and the load tends to be balanced eventually. The dynamic LB method can flexibly deal with the unbalanced problems that may occur in the system, but the implementation is more complicated, and sometimes it may affect the system performance. This paper combines the strategies and comprehensively considers the LB methods used to further improve the balancing ability of the system.

Funding

This article is not supported by any foundation.

Data Availability

Data sharing is not applicable to this article as no new data were created or analysed in this study.

Conflict of Interest

The author states that this article has no conflict of interest.

References

[1] Handur E. Particle Swarm Optimization for LB in Distributed Computing Systems – A Survey. Turkish Journal of Computer and Mathematics Education (TURCOMAT), 2021, 12(1S):257-265.

[2] Augustine J, Cohen A, Peleg D, et al. Distributed Graph Realizations. IEEE Transactions on Parallel and DSs, 2021, PP(99):1-1.

[3] Alam M, Haidri R A, Shahid M. Resource Aware LB Model for Batch of Tasks (BoT) with Best Fit Migration Policy on Heterogeneous Distributed Computing Systems. International Journal of Pervasive Computing and Communications, 2020, 16(2):113-141.

[4] Aktas M F, Behrouzi-Far A, Soljanin E, et al. Evaluating LB Performance in Distributed Storage With Redundancy. IEEE Transactions on Information Theory, 2021, PP(99):1-1.

[5] Rathan K, Roslin S. Q-Learning and MADMM Optimization Algorithm Based Interference Aware Channel Assignment Strategy for LB in WMNS. International Journal of Intelligent Engineering and Systems, 2021, 14(1):32-41.

[6] Korndrfer J, Eleliemy A, Mohammed A, et al. LB4OMP: A Dynamic LB Library for Multithreaded Applications. IEEE Transactions on Parallel and DSs, 2021, PP(99):1-1.

[7] Azzouzi S, Hsaini S, Charaf M. A Synchronized Test Control Execution Model of DSs. International Journal of Grid and High Performance Computing, 2020, 12(1):1-17.

[8] Khochare A D, Krishnan A, Simmhan Y. A Scalable Platform for Distributed Object Tracking

across a Many-camera Network. IEEE Transactions on Parallel and DSs, 2021, PP(99):1-1.
[9] Giordano A, Rango A D, Rongo R, et al. Dynamic LB in Parallel Execution of Cellular Automata. IEEE Transactions on Parallel and DSs, 2021, 32(2):470-484.
[10] Veen D J V, Kudesia R S, Heinimann H R. An Agent-Based Model of Collective Decision-Making: How Information Sharing Strategies Scale With Information Overload. IEEE Transactions on Computational Social Systems, 2020, PP(99):1-17.
[11] Sim H, Khan A, Vazhkudai S S, et al. An Integrated Indexing and Search Service for Distributed File Systems. IEEE Transactions on Parallel and DSs, 2020, PP(99):1-1.
[12] Cebrian J M, Balem T, Barredo A, et al. Compiler-Assisted Compaction/Restoration of SIMD Instructions. IEEE Transactions on Parallel and DSs, 2021, PP(99):1-1.
[13] Gupta N, Jati A, Chauhan A K, et al. PQC Acceleration Using GPUs: FrodoKEM, NewHope, and Kyber. IEEE Transactions on Parallel and DSs, 2021, 32(3):575-586.
[14] Giordano A, Rango A D, Rongo R, et al. Dynamic LB in Parallel Execution of Cellular Automata. IEEE Transactions on Parallel and DSs, 2021, 32(2):470-484.
[15] Feltus C. Reinforcement Learning's Contribution to the Cyber Security of DSs: Systematization of Knowledge. International Journal of Distributed Artificial Intelligence, 2020, 12(2):35-55.
[16] D'Amico M, Gonzalez J C. Energy hardware and workload aware job scheduling towards interconnected HPC environments. IEEE Transactions on Parallel and DSs, 2021, PP(99):1-1.
[17] Hassanzadeh-Nazarabadi Y, Kupcu A, Ozkasap O. LightChain: Scalable DHT-based Blockchain. IEEE Transactions on Parallel and DSs, 2021, PP(99):1-1.
[18] Schug A K, Werner H. Spatio-Temporal Loop Shaping for Distributed Control of PDE Systems. IFAC-PapersOnLine, 2020, 53(2):4014-4019.