Scholar Publishing Group
Distributed Processing System
https://doi.org/10.38007/DPS.2022.030206
ISSN 2790-0916 Vol. 3, Issue 2: 78-86

Development Process of Distributed System Based on Load Balancing Fault Tolerant Scheduling Algorithm

Saravan Sridevi*

University of Balochistan, Pakistan

**corresponding author*

Keywords: Load Balancing, Fault-Tolerant Scheduling Algorithm, Distributed System, Load Weight

Abstract: With the rapid development of the Internet and the advancement of information technology, the server storage model is difficult to cope with the storage and management of data by companies or departments, and the file system becomes more and more complex. The purpose of this paper is to study the distributed system development process of fault-tolerant scheduling algorithm based on load balancing. This paper takes the distributed file system as the research object, points out the load skew problem in the process of load balancing, and analyzes the reason according to the file system architecture and design concept. Based on the concepts of static load balancing and dynamic load balancing, an optimal load balancing algorithm of FastDFS is proposed. The algorithm can not only ensure the load sharing ability of the system, improve the stability of the system, but also prevent the load imbalance in the process of linear expansion of the system, and overcome the original load balancing problem. Experiments show that the distributed system constructed in this paper has better resource scheduling performance, and the real-time load is also optimized to some extent.

1. Introduction

With the rapid development of the Internet and the increasing popularity of information technology, the amount of data processed by people is increasing by tens of thousands, and the requirements for data computing and storage capabilities are constantly increasing. The era of massive data has come. According to current trends, distributed file systems have become an increasingly important technology as part of the enterprise storage space, and have been rapidly developed. The computer stores and manages data and files through the file system. It connects multiple originally unrelated file systems on nodes distributed in different locations through network communication and data transmission to form a huge number of nodes. File system network. Which node the data is stored on and from which node the data is obtained are all

transparent to the user. According to no need to care, the user only needs to manage the files and data as if they were operating the local file system [1] -2].

In the research of the distributed system development process based on the load balancing fault-tolerant scheduling algorithm, many scholars have studied it and achieved good results, for example : Chatterjee M proposed: For large-scale software development, the concept of software architecture is more The definition is much more complicated. Software architecture depends on the architect's culture and their understanding of software engineering. Some of this information is shared with development teams in trivial form, while others in the form of meaningful tools that support software development and evolution are standard [3]. Kada B et al. define: "The software definition of a computer software or program is one or more parts of a system consisting of software components, the externally visible properties of these components, and the relationships between these components" [4].

This paper takes the improvement of FastDFS load balancing algorithm as the innovation point, and completes the following work: First, it analyzes the architecture of FastDFS distributed file system and the design concept of lightweight system, cluster and peer-to-peer system. And back up and synchronize data. Changes in the load balancing process in the system can cause load stress during line extension, and in-depth analysis of the key factors for load slowing down. The full algorithm takes into account the actual load of the storage server and the remaining storage space in operation, and conducts appropriate experiments on the advanced algorithm to ensure the accuracy and high quality of the advanced algorithm.

2. Research on Distributed System Development Process Based on Load Balancing Fault-Tolerant Scheduling Algorithm

2.1. Load Balancer and Load Balancing Algorithm

The use of load balancers is to prevent service interruption and failure due to uneven server load. It is generally used when the server continues to upgrade or system maintenance. When a server is selected to launch services, new users will not be allocated to the server. This can ensure that uninterrupted services are provided for users. However, due to the large number of cloud computing platform resources, various types, large scale, and on-demand supply, if a load balancer is used to ensure that the load of the server group increases on the one hand, the cloud computing platform On the other hand, since the load balancer integrates the load balancing algorithm, and the hardware integration algorithm is limited, it cannot be applied to various resource scheduling of the cloud platform. Therefore, in the design of the cloud platform, it must be designed in line with the cloud computing service-oriented The software method of load balancing resource scheduling, that is, it is necessary to reasonably adopt the load balancing scheduling algorithm mechanism to ensure the dynamic scheduling of cloud resources [5-6].

Because load balancing is widely used in computer clusters, there is a lot of research in this area, and many balancing algorithms to solve the load problem have also been generated. The basic ones include round-robin, global random selection, global and local diffusion, and proxy-based algorithms. method and so on. The earliest use was static equalization, and it has been developed to the relatively complex but easier to implement dynamic equalization. Static load balancing is relatively simple and easy to implement, but its adaptability is relatively weak. Since performance such as execution time and task size cannot be accurately predicted before running, it can only be roughly estimated based on experience, so there is a large error in static division. In contrast, although dynamic load is easy to implement, it is more difficult than static load. Dynamic balancing

can dynamically adjust the number of tasks of each node during operation according to the node load, and can also dynamically adjust according to the response time and load conditions of each node[15-16]. Through load balancing, cluster resources can be maximized, so in practical applications, load The balancing implementation adopts dynamic load balancing [7-8].

2.2. Using Model Checking for Property Verification

Using DIScid for formal description can well describe the properties of component interactions in distributed systems, and at the same time, we hope to further verify the properties. The idea of formal property verification is to use logical reasoning (mainly theorems) for deductive verification. For complex systems, manual reasoning is cumbersome and inefficient, making it difficult to popularize in practice on a large scale. The other is the model checking method (modelcheckins) that searches the finite state space, which automatically analyzes and verifies whether the given system description L satisfies a specific property P. The principle is[9-10]

$$L \mathrm{I}\ \neg P = \phi \tag{1}$$

Whether it is true or not, and when the nature of violation or loophole occurs, its support tool can give its problem sequence, thus providing convenience for loophole location and system improvement of the system. Its detection strategy is shown in Figure 1. The core is to use Disc to describe the interaction style and nature respectively, and then turn it into the state diagram of LTS, and then perform a synchronous product of the two state diagrams to analyze and detect the interaction style constructed by Dise. Correctness [11-12].

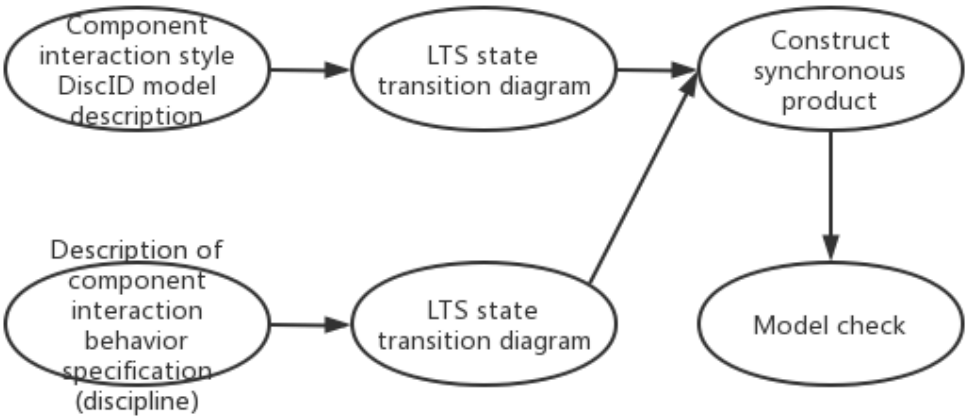

Figure 1. Discid's model detection idea

2.3. Algorithm Selection

The fitness function reflects the degree of adaptation of chromosomes to the environment. The higher the fitness function, the more likely the chromosome should be selected. In the algorithm described in this paper, the r value of the scheduling corresponding to a chromosome should be closely related to the fitness of the chromosome, because we finally use the r value to evaluate the load balancing performance of the heterogeneous system. In the worst case of system load

balancing, r takes the maximum value. Obviously, when all tasks are allocated to the node with the worst computing power in the heterogeneous system, the load balancing of the system is the worst. Then, we can determine the value range of v [13-14]. make

$$\gamma = \sqrt{\frac{1}{n}((n-1)(\frac{\sum_{j=1}^{m}(w_j * \lambda_j)}{\sum_{k=1}^{n} c_k})^2 + \left| \frac{\sum_{j=1}^{m}(w_j * \lambda_j)}{\sum_{k=1}^{n} c_k} - \frac{\sum_{j=1}^{m}(w_j * \lambda_j)}{c_{\min}} \right|} \quad (2)$$

The value range can be determined.

3. Research and Design Experiment of Distributed System Development Process Based on Load Balancing Fault-Tolerant Scheduling Algorithm

3.1. Overall Design Framework

In complex large-scale distributed systems, unified management software is used to manage resources, plan cloud resources, integrate all resources, and divide them into common high-productivity resources. High computing density and critical resources. Resources, if the separation of different resources is not enough, in the actual work in the future, you can also add new resource classes to group resources, perform capacity monitoring and real-time statistics, and also solve the problem of preset cloud source resource performance. Low can not satisfy users demand. Policy pools are designed between cloud resource pools and cloud platform services[15-16]. All policies are centrally managed in a policy pool, where policies are divided into function priority policies and set a special value. Due to the large number of cloud resources, the scheduling algorithm of cloud resources is mainly based on algorithms, and other algorithms are added according to user needs. Therefore, in the proposed planning group, the request is first transferred to the corresponding process. The resource selection is set according to the service type, and then through the load The balancing algorithm balances the system load. Finally, since cloud computing is a service, this paper mainly classifies cloud services according to the convenience they provide, and examines the performance-based characteristics of cloud computing. For the above three types, each type corresponds to a different strategy selection strategy [17-19].

3.2. Experimental Design

In this paper, two tests are carried out for the distributed system constructed in this paper. The first is the resource scheduling performance test. The resource scheduling test is carried out on each part of the distributed system to analyze its scheduling performance. The second is the change of system real-time load before and after improvement.

4. Experimental Analysis of Distributed System Development Process Based on Load Balancing Fault-Tolerant Scheduling Algorithm

4.1. Resource Scheduling Performance Test

In this paper, the resource scheduling performance of the distributed system constructed in this paper is tested, and four groups of experiments are used for longitudinal comparison, mainly for the

processing time of each task. The experimental data are shown in Table 1.

Table 1. Summary of resource scheduling performance test results

	Experiment 1	Experiment 2	Experiment 3	Experiment 4
Overall response time	433.32	450.08	449.90	450.08
Data center processing time	0.40	0.42	0.42	0.42
Virtual machine overhead	539.59	539.59	539.59	539.59
Data transmission overhead	1.91	2.53	2.53	2.53

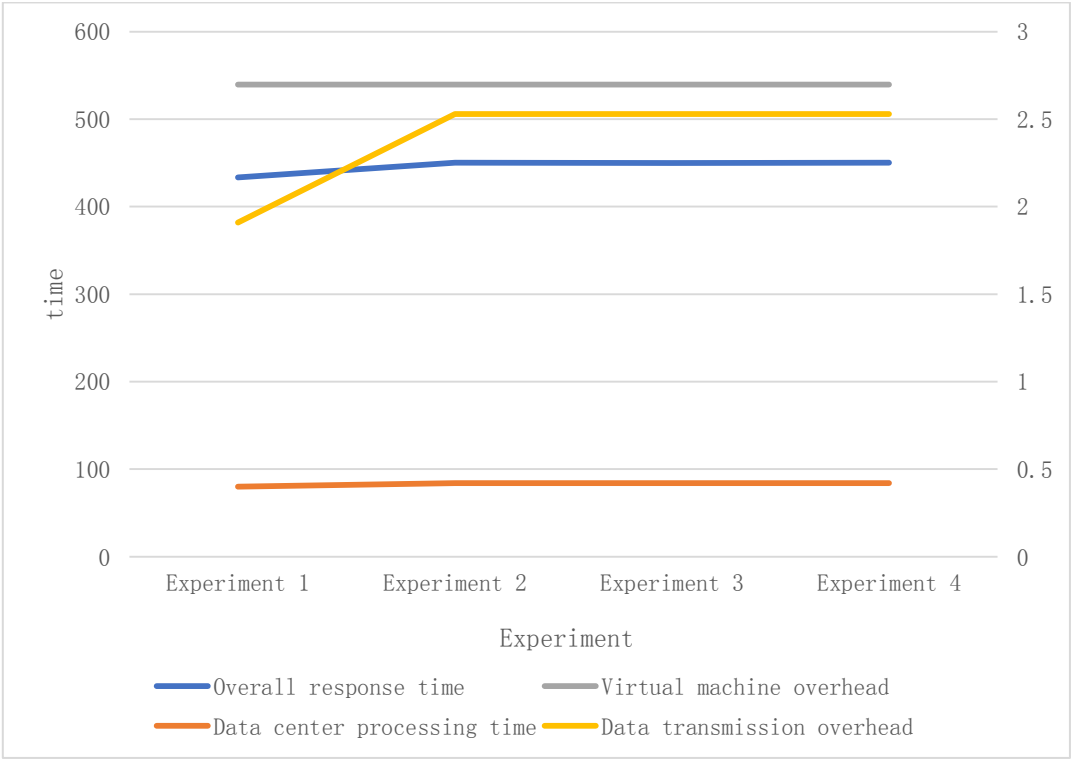

Figure 2. Processing time of the distributed system

Figure 2 Results Comprehensive analysis of six groups of experimental data from four directions: overall response time, data processing time, virtual machine overhead and data transmission overhead. The central processing time is slightly higher than the time of using the load balancing algorithm alone. From the perspective of system overhead, the data transmission overhead is also higher than the time of the independent balancing algorithm, but the data transmission overhead of the superposition algorithm is the same, and the virtual machine overhead is the same as the scheduling used. The algorithm is related to the stacking algorithm, because the different stacking algorithms lead to slight differences in the virtual machine overhead. From the data center load graph in the experimental data, it can be seen that the data center load in a separate load balancing algorithm is lower than the system load after the algorithm is stacked. But the load of the three data centers is relatively balanced. The experimental results show that the use of the scheduling algorithm and the load balancing algorithm will have a certain impact on the overall response time and system overhead, but the impact is not very large, and the overall system load tends to be stable in the system resource allocation, and at the same time it can meet the The different needs of users for different distributed system services.

4.2. Real-Time Load Rate Change

In this paper, 10 threads are used to simulate the number of concurrent users, the number of uploads per thread is 1024, and the size of a single file is 1M. At the same time, the real-time load value is obtained by reading the data in the shared memory. The statistical results of the experiments on the three load balancing strategies are shown in Table 2 below.

Table 2. Real-time load value change before and after the improvement

	Group1	Group2	Group3
Improvement 2	26	32	60
Improvement 1	26	32	70
Before the improvement	14	26	50

As can be seen from Figure 3, according to the above three load value change curves, among the load changes before the improvement, only the load of group 3 has been changing, and the other two have not changed, showing a serious load imbalance phenomenon. In the load change of Improvement 1, it can be seen that when the load value of group3 exceeds 75, group2 will help to share some of the load. In the load change of improvement 2, the load values among the three groups all change, and the change of group3 is more severe, which means that group3 bears more load.

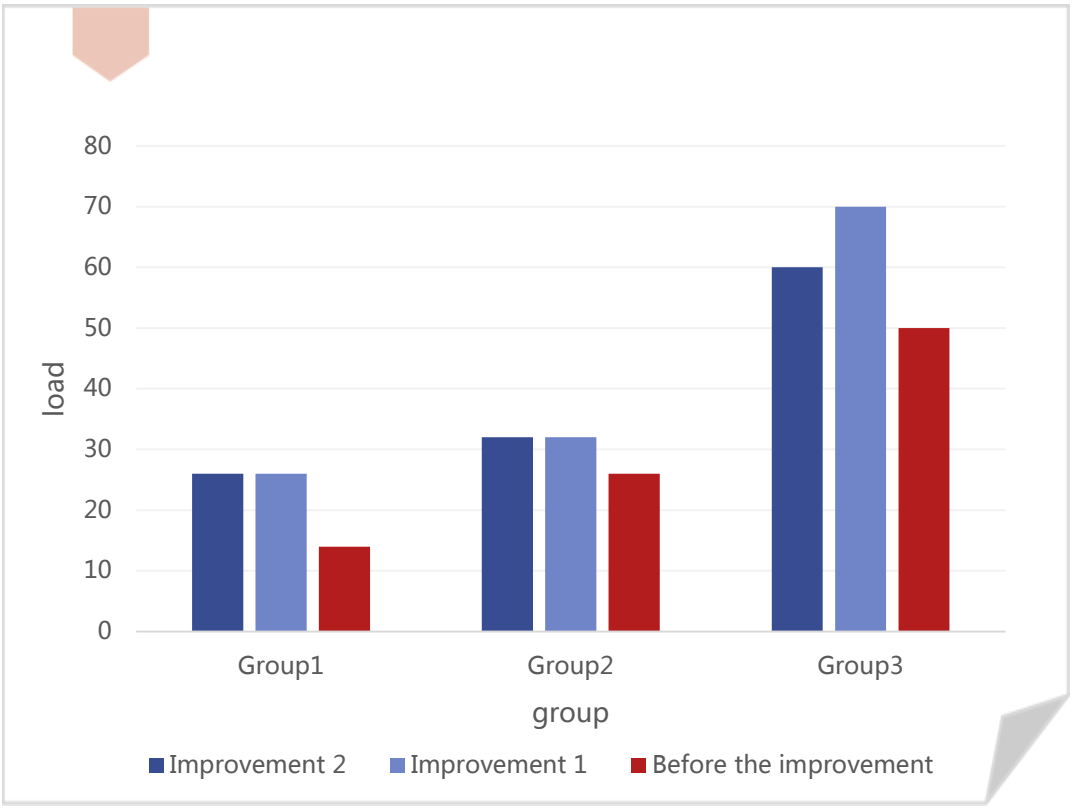

Figure 3. Real-time load change diagram

5. Conclusion

This article takes the FastDFS distributed file system as the research object, so it is necessary to understand the architecture and design concept of the system. Then, the load balancing technology is deeply studied, and it is pointed out that the load sloping phenomenon will occur for a period of time when the algorithm is horizontally expanded, and the main reason for this is that the real load of the server node is not considered. For this reason, this paper uses the idea of dynamic load balancing algorithm to regularly collect the real-time load of server nodes. In order to reduce the system overhead caused by dynamic load, and considering the balance of storage space, the static load balancing algorithm is simple, low cost, and The characteristics of ensuring load balancing between nodes with the same load in a short time, using the remaining space to calculate the weights, adopting the idea of weighted round-robin, combining dynamic and static algorithms, and taking into account the adoption of different load strategies for different services, This improvement is reflected in that the entire life cycle of the distributed system is unified in the control of the software architecture. ADisDTool supports the implementation of the above methods proposed in this paper in the development of distributed systems, and provides syntax editing environments such as DISADL and Discid. Semantic conversion tools, property verification tools, graphical interactive environment, etc., have good convenience and versatility.

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] Friedemann S, Raffin B. An elastic framework for ensemble-based large-scale data assimilation:. The International Journal of High Performance Computing Applications, 2022, 36(4):543-563.

[2] Zjavka L. Power quality statistical predictions based on differential, deep and probabilistic learning using off‐grid and meteo data in 24‐hour horizon. International Journal of Energy Research, 2022, 46(8):10182-10196. https://doi.org/10.1002/er.7431

[3] Chatterjee M, Mitra A, Setua S K, et al. Gossip-based fault-tolerant load balancing algorithm with low communication overhead. Computers & Electrical Engineering, 2020, 81(1):106517.

[4] Kada B, Kalla H. A Fault-Tolerant Scheduling Algorithm Based on Checkpointing and Redundancy for Distributed Real-Time Systems. International journal of distributed systems and technologies, 2019, 10(3):58-75.

[5] Zhou P, Huang J, Qin X, et al. PaRS: A Popularity-Aware Redundancy Scheme for In-Memory Stores. IEEE Transactions on Computers, 2019, 68(4):556-569. https://doi.org/10.1109/TC.2018.2876827

[6] Ahmad N, Elhassan H M, Rehman M B, et al. Comparative Study on Load Balancing Algorithm for Multiprocessor Interconnection Networks. International Journal of Advanced Trends in Computer Science and Engineering, 2019, 8(3):410-414.

[7] Komalavalli D, Padma T. Swarm intelligence-based task scheduling algorithm for load balancing in cloud system. International Journal of Enterprise Network Management, 2021, 12(1):1. https://doi.org/10.1504/IJENM.2021.112669

[8] Garg V, Tiwari R, Shukla A, et al. A Distributed Cooperative Approach for Dynamic Target Search Using Particle Swarm Optimization with Limited Intercommunication. Arabian Journal for Science and Engineering, 2022, 47(8):10623-10637.

[9] Ray S, Kasturi K, Patnaik S, et al. Optimal allocation of DGs for non-linear objective function modeling in a three-phase unbalanced distribution system using crow search optimization algorithm. Journal of Interdisciplinary Mathematics, 2022, 25(3):681-701.

[10] Kumar C, Marston S, Sen R, et al. Greening the Cloud: A Load Balancing Mechanism to Optimize Cloud Computing Networks. Journal of Management Information Systems, 2022, 39(2):513-541.

[11] Manoharan J S. Double attribute based node deployment in wireless sensor networks using novel weight based clustering approach. Sādhanā, 2022, 47(3):1-11.

[12] Ziyath S P M, Subramaniyan S. An Improved Q-Learning-Based Scheduling Strategy with Load Balancing for Infrastructure-Based Cloud Services. Arabian Journal for Science and Engineering, 2021, 47(8):9547-9555. https://doi.org/10.1007/s13369-021-06279-y

[13] Bouyahia O, Abdallah A, Yazidi A, et al. Fault Tolerant Fuzzy Logic Control of a 6-Phase Induction Generator for Wind Turbine Energy Production. Electric Power Components and

Systems, 2021, 49(8):756-766.

[14] Huo X, Wu K, Miao W, et al. Research on Network Traffic Anomaly Detection of Source-Network-Load Industrial Control System Based on GRU-OCSVM. IOP Conference Series: Earth and Environmental Science, 2019, 300(4):042043 (7pp).

[15] Wang Y J, Ai B B, Qin C Z, et al. A load-balancing strategy for data domain decomposition in parallel programming libraries of raster-based geocomputation. International Journal of Geographical Information Science, 2022, 36(5):968-991.

[16] Rao K P, Ramamurthy D V, Reddy N S, et al. Integrated simultaneous scheduling of machines, automated guided vehicles and tools in multi machine flexible manufacturing system using symbiotic organisms search algorithm. Journal of Industrial and Production Engineering, 2022, 39(4):317-339. https://doi.org/10.1080/21681015.2021.1991014

[17] Tsang K T, Liu J J, Deng Y H. A variant RSA acceleration with parallelisation. International Journal of Parallel, Emergent and Distributed Systems, 2022, 37(3):318-332.

[18] Huang B, Ma P, Lin H, et al. Distributed scatterer interferometry for forested and hilly areas using a topographical homogeneous filtering. Remote Sensing Letters, 2022, 13(5):460-469.

[19] Melo T T, Ferreira M, Bezerra L, et al. Effect of replacing soybean meal by a blend of ground corn and urea-ammonium sulphate on milk production and composition, digestibility and N balance of dairy Murrah buffaloes. Journal of Dairy Research, 2022, 89(2):134-140. https://doi.org/10.1017/S002202992200036X