Scholar Publishing Group
Distributed Processing System
https://doi.org/10.38007/DPS.2021.020406
ISSN 2790-0916 Vol. 2, Issue 4: 44-51

Distributed System Application Strategy based on Dynamic Load Balancing Model of Cloud Computing

Mazinin Anuen*

Griffith University, Australia

**corresponding author*

Keywords: Cloud Computing Technology, Dynamic Load Balancing Model, Distributed System, Application Strategy Research

Abstract: As a new distributed computing technology, more and more applications are deployed and developed in the cloud system. At the same time, load balancing(LB) is the key technology to solve the high-performance computing in the distributed system(DS). Therefore, this paper studies the application strategy of dynamic LB model based on cloud computing(CC) in DS. CC technology and LB algorithm are briefly analyzed; The dynamic LB model based on CC is discussed, and the dynamic LB strategy is designed; Finally, the effectiveness of the application strategy of the dynamic LB model based on CC in the DS is analyzed through simulation experiments. The experimental results show that the dynamic LB algorithm designed in this paper has a good effect on the LB of each node in the DS.

1. Introduction

CC represents the development direction of computer and communication technology in the future. Cloud storage represented by DS is an important part of CC, and LB technology is one of the core technologies of cloud storage. In recent years, distributed file system has become a promising application architecture of distributed file system with its own advantages, and it is the research hotspot of commercial cluster system. So it is very important to study the practical application of cluster. Therefore, the research on how to design a LB strategy suitable for this specific application scenario has become the focus. In this paper, the application strategy of dynamic LB model based on CC in DSs is studied and analyzed.

As the LB technology is very important to the overall performance of the distributed cluster system, and is the core technology to meet the high availability needs of users for the distributed cluster system, many colleges, research institutions and enterprises at home and abroad have conducted long-term research and development in this field. Wallden m et al analyzed and

compared several dynamic LB algorithms, such as bee foraging, preference based random sampling and active clustering algorithm. Their analysis shows that with the increase of system diversity, the bee foraging algorithm can give full play to the throughput compared with the other two algorithms [1]. Alam m proposed two static algorithms in cloud environment. One of them is the opportunistic LB algorithm. In this algorithm, the service manager calculates the nodes with the smallest task execution time, and these nodes accept and process the upcoming tasks. The second strategy is the minimum LB strategy, which improves the resource utilization of the cloud data center by maintaining load balance [2].

For the cloud system, how to reasonably schedule tasks and allocate resources to ensure the load balance of the DS is very important. Firstly, this paper expounds the development status of CC at home and abroad. According to the analysis of CC technology, the LB problem of cloud system is pointed out; Secondly, the characteristics and classification of LB technology are analyzed; The application strategy of dynamic LB model based on CC in DS is studied and analyzed, and the dynamic LB algorithm based on CC is proposed. In this algorithm, the task master agent and task slave agent with master-slave structure are designed in the DS, and efficient task scheduling and resource allocation are realized. Through evaluation and comparative simulation experiments, the results show that the strategy is excellent in task completion time, average response time and load balance [3-4].

2. Dynamic LB Model based on CC

2.1. CC

As a new distributed computing technology, more and more applications are deployed and developed in the cloud system. Through cloud service providers such as Amazon, Google, Microsoft and Alibaba, we can quickly configure and publish applications with minimal management. CC provides a service-oriented architecture and network service application, including fault tolerance, feasibility, flexibility and scalability, and reduces the cost of information technology [5].

CC technology architecture: from the technical level, CC can be roughly divided into four parts. The main function of the physical resource layer is the integration of physical hardware resources and related network equipment. The resource pool layer is mainly used to virtualize physical resources to form a virtual resource pool for the upper layer to call. The main function of the resource middleware layer is to manage the resources, security, users and tasks of the system, monitor the system, detect and adjust the resource load. The SOA layer provides users with encapsulated CC web applications, and users can obtain corresponding services and resources according to their own needs [6-7].

2.2. Introduction to LB Algorithm

LB means to increase the computing capacity, reduce the computing time and improve the comprehensive performance of the DS through the allocation and scheduling of load tasks. Therefore, LB is the key technology to solve high-performance computing in DSs. Generally speaking, the LB algorithm will complete the task allocation and scheduling according to the state information of the terminals in the DS. The specific state information includes the CPU utilization rate, memory utilization rate, communication capacity and other performance parameters of each terminal. Therefore, LB problem can also be called task scheduling problem [8-9].

LB algorithm plays an important role in DS. This kind of algorithm can not only solve the problem of balanced allocation of terminal simulation tasks in DS, but also reduce the communication overhead between terminals. It is a key technology to improve the running speed of DS [10].

Generally, LB algorithms are divided into static LB algorithms and dynamic LB algorithms.

Static LB algorithm refers to the situation that the task allocation has been completed before the system runs, and the task will not migrate during the system operation.

Dynamic LB refers to the imbalance between the terminals caused by the configuration of the running terminals and the size of the overhead of the tasks during the system operation. At this time, the tasks between the terminals need to be migrated to achieve dynamic balancing.

Static LB algorithm is generally applicable to DSs with small fluctuation of load tasks. However, due to the particularity of distributed traffic network simulation system, the fluctuation of terminal load in the system is determined. Therefore, dynamic LB is more commonly used to solve the load distribution problem of distributed traffic network simulation system [11-12].

2.3. Dynamic LB Model based on CC

At the beginning, the LB technology only distributes user requests and task requests to the server in the form of static algorithms, Improve its resource utilization rate [13-14]. However, with the increase of requests, the tasks assigned to nodes with weak computing capacity will take a long time to be processed, with long waiting time, increased processing delay, and reduced overall system efficiency. It greatly improves the response speed of the cloud data center and makes more rational use of resources, thus avoiding unnecessary power consumption and waste of idle resources [15-16].

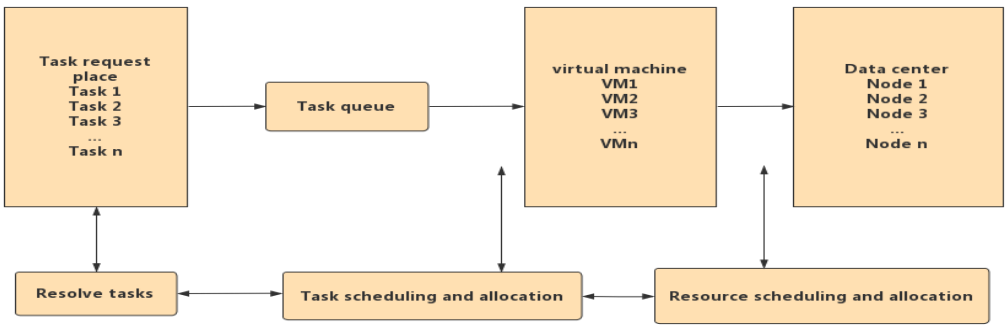

Figure 1. Two stages of implementing LB

In the CC environment, the LB process can be completed in two different stages, as shown in Figure 1. The different implementation stages of LB mechanism can be summed up as primary distribution and redistribution. The initial assignment is to assign tasks to each physical node before the tasks enter the node, so that the system can maintain an efficient running state. Reallocation is completed during node operation [17-18].

3. Design of Dynamic LB Strategy

The dynamic LB strategy designed in this paper is a centralized strategy. A manager is set up in the system to collect the load information of other execution nodes periodically and provide the

target node query service for the nodes that need to balance the load. This manager is a node with relatively high performance in the system and does not participate in the execution of tasks. Its relationship with other nodes and its own functional modules are shown in Figure 2.

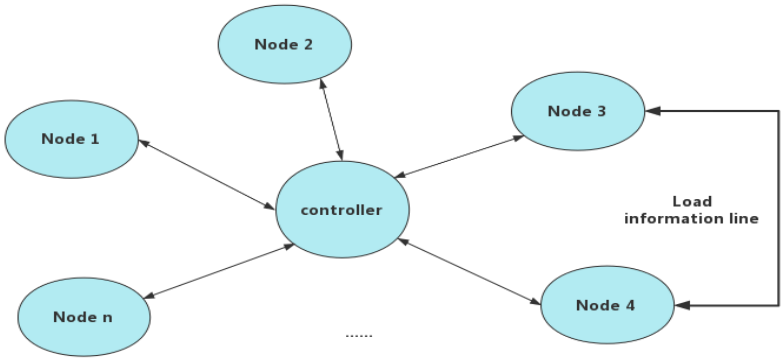

Figure 2. Node relationship diagram in the system

On the task execution nodes 1 and 2, the node needs to periodically collect its own load information, perform load calculation according to the preset load index, and evaluate its status at a certain time (sender, receiver, or neither sending nor receiving). If LB is required, it needs to communicate with the manager, find the target node with which it can balance the load, send or receive a part of the tasks, The node itself also sequentially executes the local task queues in the ready state.

3.1. Selection of Load Index

The load index Li is the quantification of the node load evaluation standard, and it is an important parameter of the dynamic LB strategy. The evaluation results of the current load condition of a node are different depending on the selected load index. In any case, the selected load index must accurately reflect the load status of the node. Only when the system can accurately evaluate the current load status of each node can the node timely determine its current status (i.e., whether it is overloaded, lightly loaded or properly loaded), whether it needs to transfer tasks, and its current role in the system (i.e., sender, receiver or not participating in task transfer activities); However, if the load information of the node is not accurately evaluated, the node will not be able to carry out LB activities in a timely manner, resulting in poor performance of the entire system.

3.2. Data Structure Table and Relevant Calculation of Nodes

Firstly, the relevant definitions and calculation methods are introduced as follows:

When the execution node of the system is put into use for the first time, the system administrator will set an ideal initial load value for it. This value is obtained according to the hardware configuration of each node and is recorded as HLi (I = 1,2,..., n). Generally, the higher the hardware configuration, the larger the default value.

The dynamic load value is calculated according to the parameters of various aspects during the operation of the node, and is recorded as kli (I = 1,2,..., n). The calculation formula is expressed as:

$$KL_i = V_i \times L_{cpu}(i) + V_2 \times L_{io}(i) + V_3 \times L_{memory}(i) + V_4 \times L_{process}(i) \quad (1)$$

Where VI is the scale parameter set by the system for the node, VI ∈ [0,1], and Σ VI = 1. For different types of system applications, the settings of various parameters are different. In the environment of web applications, the weight of memory resources and response time can be set to be larger. LCPU (I), Lio (I), lmemory (I), lprocess (I) and lresponse (I) are the available load values of a certain parameter of node i, which are CPU utilization, I / O utilization, memory utilization, total number of processes and response time in turn. The purpose of dynamic load value is to accurately reflect the load condition of the node and serve as the basis for predicting the future load condition of the node.

According to the initial load value and dynamic load value of the node, the current load value li of the node can be calculated. The calculation formula is as follows:

$$L_i = HL_i + E \times (KL_i - HL_i) \tag{2}$$

The current load value li of a node is a measure of the load status of a node in the system, which is used as a basis for locating a light load node when a heavy load node transfers tasks, and it is also the basis for the system manager to generate the sending node table and the receiving node table.

The specific division method is as follows:

(1) When Li ⩽ LT, the node is in light load state;

(2) When LT < Li <ht, the node is in the load state;

(3) When Li ⩾ HT, the node is in the overload state.

The specific types and corresponding representation methods of nodes are shown in Table 1.

Table 1. Node status correspondence table

Current load status of node	Current role (type) of the node	Representation in data table
Heavy load	Sender	S
Light load	Recipient	R
Cargo-worthiness	OK Node (neither send nor receive)	OK

The sender node needs to transfer a part of its own load and can no longer receive new tasks. The receiver node can receive tasks transferred from other nodes, while the OK node has sufficient resources to complete its current tasks, but can no longer receive new tasks.

4. Experimental Analysis of Application Strategy of Dynamic LB Model based on CC in DS

The decision support system implemented in this paper has eight ordinary nodes that perform business. The hardware configuration of these eight nodes is not the same. The author selects 2, 4 and 7 nodes as the representative nodes of each performance for analysis, and intercepts the operation data of these three nodes in the same period of time from the respective load result tables saved during the operation of these three nodes, The load time and CPU utilization time curves of these three nodes are drawn.

The load time diagram corresponding to these three nodes is shown in Figure 3. Where, the horizontal axis represents the time, the vertical axis represents the number of loads (that is, the number of tasks), the horizontal axis sets the start time to 0, and the interval time to 5 × 2 seconds,

the interception time length is 360 seconds, and the vertical axis is set with the initial load number of 0 and the interval number of 100.

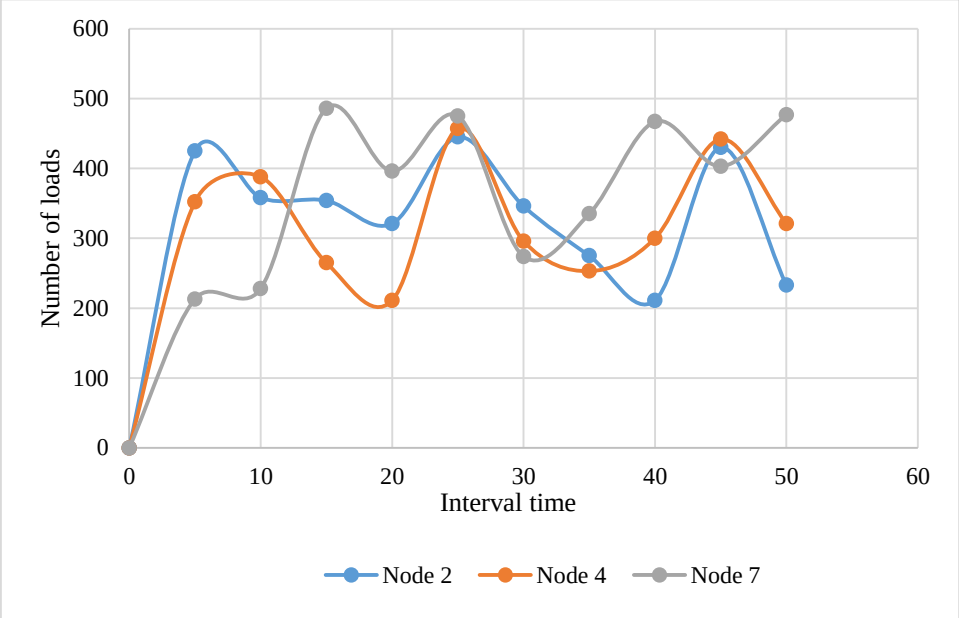

Figure 3. Load time diagram of different nodes

It can be seen from Fig. 3 that the loads of these three nodes are neither too heavy nor too light at the same time, which indicates that by using the dynamic LB strategy, the loads of each node can maintain a good balance effect at the same time. The CPU utilization time curve corresponding to these three nodes is shown in Figure 4. Where, the horizontal axis represents time, and the vertical axis represents CPU utilization. The setting of the horizontal axis is the same as that of the load time graph, and the same time data is intercepted. The vertical axis sets the initial utilization rate as 0 and the interval number as 0.2.

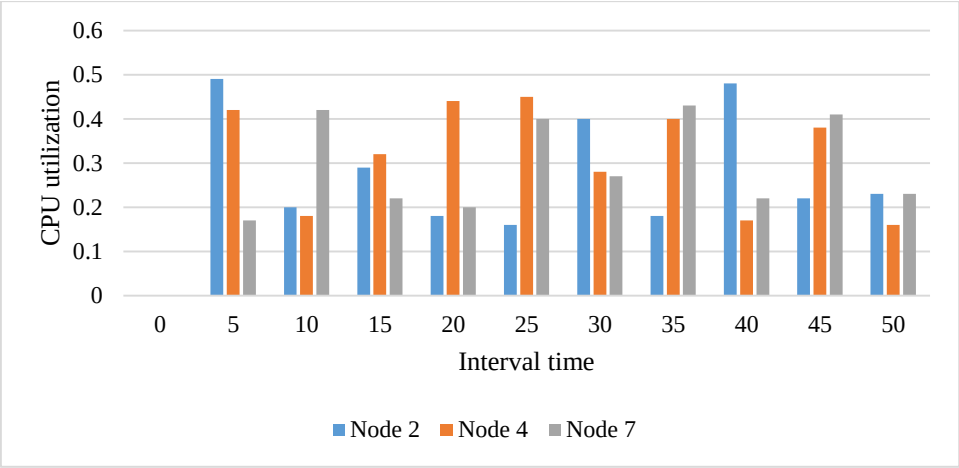

Figure 4. CPU utilization of different nodes - time chart

It can be seen from Fig. 4 that the CPU utilization of these three nodes is also relatively average, and there is no case that one node is too busy and the other node is too busy at the same time. This shows that all nodes perform tasks well and the task allocation is reasonable. In conclusion, the

dynamic LB algorithm designed in this paper has a good effect on the LB of each node in the system.

5. Conclusion

In this paper, the application strategy of dynamic LB model based on CC in DS is studied. Through the analysis of the operation results of each node in the system, it can be seen that the improved dynamic LB strategy designed in this paper can adapt to the change of the load state of the system, adjust the startup strategy in time, and reduce the system overhead to a certain extent. The implementation effect of LB is good; The designed system is implemented, and the running results of the system are analyzed to verify the stability and effectiveness of the strategy. This paper adopts a centralized strategy. Although the workload of the manager has been reduced to a certain extent, if there are many nodes, the workload of the manager will increase correspondingly, and it needs to be further optimized; Although there are many researches on LB, there are still many problems: there is no effective LB algorithm that can be applied to all DSs. Therefore, a lot of research is needed on the application strategy of dynamic LB model in DSs.

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] Wallden M, Markidis S, Okita M, et al. Memory Efficient Load Balancing for Distributed Large-Scale Volume Rendering Using a Two-Layered Group Structure. IEICE Transactions on Information and Systems, 2019, E102.D(12):2306-2316.

[2] Alam M, Haidri R A, Shahid M. Resource Aware Load Balancing 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.

[3] Cabrera A, Acosta A, Almeida F, et al. A heuristic technique to improve energy efficiency with dynamic load balancing. Journal of Supercomputing, 2019, 75(3):1610-1624.

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

[5] Nakajo Y, Athavale J, Yoda M, et al. Dynamic Load Balancing Using Actual Workload Traces Based on Central Processing Unit Temperatures. Journal of Electronic Packaging, 2019, 141(3):031014.1-031014.20.

[6] Al-Sayegh A, Sotelino E D. A New Row-Wise Parallel Finite Element Analysis Algorithm with

Dynamic Load Balancing. International Journal of Earthquake and Impact Engineering, 2020, 3(2):120-142.

[7] Giordano A, Rango A D, Rongo R, et al. Dynamic Load Balancing in Parallel Execution of Cellular Automata. IEEE Transactions on Parallel and Distributed Systems, 2021, 32(2):470-484.

[8] Huang J, Liu Y, Li R, et al. Optimal power allocation and load balancing for non-dedicated heterogeneous distributed embedded computing systems. Journal of Parallel and Distributed Computing, 2019, 130(AUG.):24-36.

[9] Perez A C, Acosta A, Almeida F, et al. A Dynamic Multi–Objective Approach for Dynamic Load Balancing in Heterogeneous Systems. IEEE Transactions on Parallel and Distributed Systems, 2020, PP(99):1-1.

[10] Stephen B, Telford R, Galloway S. Non-Gaussian Residual based Short Term Load Forecast Adjustment for Distribution Feeders. IEEE Access, 2020, PP(99):1-1.

[11] Khalid Y N, Aleem M, Ahmed U, et al. Troodon A machine-learning based load-balancing application scheduler for CPU–GPU system. Journal of Parallel and Distributed Computing, 2019, 132(OCT.):79-94.

[12] Daraghmi E, Daraghmi Y A. Advanced Diffusion Approach To Dynamic Load-Balancing For Cloud Storage. International Journal of Parallel and Distributed Systems and Networks, 2019, 10(2/3):01-13.

[13] Korndrfer J, Eleliemy A, Mohammed A, et al. LB4OMP: A Dynamic Load Balancing Library for Multithreaded Applications. IEEE Transactions on Parallel and Distributed Systems, 2021, PP(99):1-1.

[14] Kitsuwan N, Pavarangkoon P, Widiyanto H M, et al. Dynamic load balancing with learning model for Sudoku solving system. Digital Communications and Networks, 2020, 6(1):108-114.

[15] Pei J, Hong P, Xue K, et al. Efficiently Embedding Service Function Chains with Dynamic Virtual Network Function Placement in Geo-distributed Cloud System. IEEE Transactions on Parallel and Distributed Systems, 2019, 30(99):2179-2192.

[16] Liu K Z, Teel A R, Sun X M, et al. Model-Based Dynamic Event-Triggered Control for Systems With Uncertainty: A Hybrid System Approach. IEEE Transactions on Automatic Control, 2020, PP(99):1-1.

[17] Haji L M, Zeebaree S, Ahmed O M, et al. Dynamic Resource Allocation for Distributed Systems and Cloud Computing. Test Engineering and Management, 2020, 83(May-June 2020):22417 – 22426.

[18] Dwivedi K M, Osuch T, Trivedi G. High sensitive and large dynamic range quasi-distributed sensing system based on slow-light -phase-shifted fiber Bragg gratings. Opto-Electronics Review, 2019, 27(3):233-240.