Big data cloud platform server load balancing algorithm based on improved chaotic partition algorithm

Hao Xiang¹,³, Ting Zhang¹,²,*, Zhewen Li¹

¹School of Artificial Intelligence, Wuhan Technology and Business University, Wuhan, China
²Institute of Information and Intelligent Engineering Applications, Wuhan Technology and Business University, Wuhan, China
³School of Information Science and Technology, Tibet University, Tibet Autonomous Region China, China

*Corresponding author: zhangting@163.com

Abstract. Due to the heterogeneity of big data cloud platform and the uneven distribution of data between cloud servers, when cloud server cluster processes a large number of tasks, the node load is often uneven. To solve this problem, a load balancing algorithm based on improved chaotic partition algorithm for big data cloud platform server is proposed. According to the statistics of the average resource consumption of various services provided by the cluster, combined with the running time and resource occupation of tasks on the server, the total remaining task load of the server at a certain time point is predicted, so as to obtain the actual task load status of the node, and correct the task load in time. Experiments show that the load balancing algorithm of big data cloud platform server based on improved chaotic partition algorithm can effectively balance the load of multi task heterogeneous cloud servers, and has high feasibility. Then, based on this improved load balancing algorithm, we can also extend it to the application of multi-objective algorithm, such as robot path planning, target allocation scheduling algorithm and so on. There will be similar and reasonable performance compared with the same period last year.

Keywords: Chaotic partition algorithm; Big data; cloud platform; Load balancing; robot; Multi objective path planning

1. Introduction

Cloud computing, as the core technology of big data, combines traditional computer technologies such as grid computing, distributed computing, parallel computing, network storage, virtualization, load balancing and other traditional computer technologies with network technology. Cloud service adopts distributed storage mode, and data is no longer stored on a single machine or server, but stored on multiple machines on a group of cloud servers[1]. Through the cloud computing platform, the hardware equipment of cluster server is integrated together to provide external data storage and business access functions, which greatly improves the data processing ability of the system, with good scalability and easy management and maintenance. In cloud server cluster, due to the heterogeneity of
software environment and the imbalance of data between servers, a large number of tasks are often needed to process, resulting in unbalanced load of nodes and affecting the quality of service[2]. Generally, both the wheeled load balancing algorithm and the weighted wheeled load balancing algorithm allocate the task with fixed probability without considering the network and server state information[3]. The dynamic load balancing algorithm is used to allocate tasks and obtain the load status information of network or server in real time, including minimum connection, minimum weight, minimum location connection, repeated connection, etc. Based on the improved chaotic partition algorithm, the number of current connections of each server is calculated when each request is allocated, and the request is assigned to the server with the least number of links. Long term operation, the node load can not be adjusted in time, resulting in system tilt, affecting the balance effect. Through the comprehensive analysis of node performance, a node selection method based on load weighting is proposed, and the load distribution probability is introduced on this basis. The disadvantage is that the resources required by computing tasks are inconsistent, and it is difficult for nodes to use a single resource.

In the multi-objective chaotic partition algorithm, Multi mobile robot cooperative detection of multiple target positions is a common form of multi mobile robot cooperative detection environment, such as multiple military robots cooperative reconnaissance of multiple military sites, multiple space robots collecting spatial feature data of multiple observation points and other applications will involve this form of task planning. Similar problems are multi traveling salesman problem (MTSP) and multi vehicle routing problem (VRP), but the algorithms of these two kinds of problems cannot be directly used to solve multi robot systems.

The main reason is that the dynamic uncertainty of the robot working environment requires the system to adjust the task planning in real time, and the computational complexity of the planning algorithm should be as low as possible. In addition, the vehicle computing capacity of robot system is usually very limited, which is not enough to quickly execute complex planning program to achieve real-time task planning. Therefore, it is necessary to study the task planning method which can meet the needs of multi mobile robot system. Cooperative detection of multiple target positions by multiple mobile robots.

2. Big data cloud platform server load balancing algorithm

2.1. Data scheduling and processing method of big data cloud platform server

![Cloud platform server data scheduling model](image)

For the dynamic load balancing algorithm of big data cloud platform server, it is generally necessary to dynamically feed back the server's load information to the front-end task scheduler as the basis for task scheduling[4]. However, a complete dynamic load balancing algorithm needs to solve the following problems: how to receive user requests, how to dynamically feed back load information to the front-end task scheduler, and how to judge the load status, How to select the target server, how to deal with node overload and so on. On this basis, a centralized task scheduling strategy is proposed, that is, nodes monitor their own load information, regularly feed back information to front-end dispatchers, use the improved chaotic partition algorithm to judge the node load, realize effective dynamic alarm when the node load is too large, and classify the task requests, match different types of requests with specific server groups[5]. The weighted ratio of the working capacity and the real-time
load value of the node determines the forwarding target of the task request, and regularly maintains the node information in the server group. This chart shows a dynamic load balancing algorithm for an improved application model.

In terms of load information collection, distributed load information collection is more challenging than centralized load information collection. Due to the centralized collection mode, the load scheduler often actively queries the load information, and the workload of these load schedulers has become a bottleneck of system performance[6]. Among them, the distributed method entrusts the collection of load information to the server nodes, so that they can monitor their own load information in the cluster in real time, and only periodically feed back the information to the front-end task scheduler. The real-time performance of the system is ensured, and the overload information is timely fed back to the front-end dispatcher in case of node overload. In addition, the workload of overload scheduler can be reduced. This paper classifies service requests to match specific types of requests with server groups, so as to reduce the repeated caching of the same content on multiple servers, improve the cache hit rate of servers, shorten the response time of requests, and improve the system throughput[7]. According to the collection strategy of load information and the judgment of load status, how to select the target server is discussed. According to the selection strategy of the target server, after receiving the new task request, the front-end scheduler first determines the type of the task request, then matches with the corresponding server group, and then determines the forwarding probability according to the current load weighting ratio of each node in the corresponding server group. When the load proportion value of wi node is 0, the expression of weight value Pi is calculated according to the load proportion value of wi,

\[
p_i = (1 - \omega_i) / \sum_{i=0}^{n-1} (1 - \omega_i)
\]

that is, the probability value of the current task request being forwarded to I

\[
p_i = (1 - \omega_i) / \sum_{i=0}^{n-1} (1 - \omega_i)
\]

In this way, we can reduce the number of requests in a server, reduce the number of requests in a server, and reduce the number of requests in the same server[8]. Because the system can process more user requests per unit time, the overall performance of the server cluster has been improved. Compared with the traditional greedy strategy or other strategies of tracking server load information, this balancing strategy based on probability weighted forwarding obtains better load balancing degree based on the advantages of tracking server load information strategy, and improves the server cache hit rate by classifying requests. In this way, we can expect to achieve a certain balance in the degree of load balancing and system performance.

2.2. Network load balancing algorithm based on improved chaotic partition

According to resource consumption, such as CPU, memory and network bandwidth, tasks can be divided into three types: computing task preference, network task preference, computing task preference and network task preference. Obviously, in the software and hardware environment, the same task request, the same CPU time, the same memory consumption, the same network transmission capacity [9]. When the memory resources are enough to meet the task execution, the time required to complete the task is usually only related to CPU and network resources.

Therefore, this paper assumes that the server has sufficient memory resources, LCPU represents the total traffic generated by the CPU in the process of task completion, LUL and LDL represent the traffic generated in the process of CPU resource upload and download respectively. Based on the above parameter setting, the system flow generated after task J starts is calculated according to Formula (2):
In Formula (2), \( t_{\text{begin}} \) and \( t_{\text{end}} \) respectively represent task start time and end time, \( CPU_P(j) \) represents CPU traffic consumption per unit time under task \( j \), \( UL_P(j) \) and \( DL_P(j) \) represent traffic consumption per unit time during CPU resource upload and download under task \( j \), and \( p \) represents CPU traffic control coefficient.

In order to ensure that the new tasks assigned to the server can get enough system resources, the algorithm aims to achieve the maximum load value between the servers. Cloud server cluster contains load servers, denoted as \( S \), i.e., \( S = \{S_1, S_2, \ldots, S_n\} \), each node of cloud server is denoted as \( C = \{C(S_1), C(S_2), \ldots, C(S_n)\} \). Taking the \( j \)-th server as an example, the load balance recorded at time \( T \) is calculated as follows:

\[
S(j) \in S \\
G_{CPU}^*(S(j), t) = \max \left( \sum_{C(S_i)} G_{CPU}^* \right. \\
\left. \left( S_i \in S, t \right) \right)
\]

2.3. Implementation of server load balancing in big data cloud platform

The big data cloud platform server load balancing algorithm realizes the load balancing when the cluster system is transparent to users. In addition to focusing on how to handle the request, users also focus on whether they can respond to the request as soon as possible and when they can receive the response [12]. When processing task requests, server-side requirements will not lead to server overload or affect server performance. Generally speaking, the factors that affect server operation include CPU utilization, task queue length, processing speed, network utilization, number of processes, etc.

Set the CPU task queue length as \( L \), and calculate the sum of server utilization \( Q \) of all tasks by using the network server utilization \( q_i \) corresponding to the current task \( J \). The formula is as follows:

\[
Q = \sum_{j=1}^{n} q_j \cdot L, \quad 0 \leq q_j \leq Q
\]

Using the server task scheduling strategy, when the node server processing tasks and load capacity reach the maximum match, the load evaluation module mainly completes two functions: comprehensive load ratio calculation and judge the load state of the server. The load query module can obtain the maximum working capacity of CPU memory, disk I/O utilization, and current network bandwidth. The 10,000 steps are load state assessment. In the load assessment module, load calculation and load state are unified. When NodeServer allocated \( n \) tasks to \( M \) NodeServer, the resource demand and energy consumption were separated. The specific algorithm was as follows:

\[
G(N) = \frac{\sum_{C(S)} B(S) + \sum_{C(S)} F(S)}{Q}
\]

\[
E(N) = \frac{\sum_{C(S)} E(S) + \sum_{C(S)} F(S)}{Q}
\]
Where: G (N) represents the total amount of resources required when N tasks are allocated to M node servers; E (N) is the total energy of the server of the big data cloud platform; B (S) refers to the computing capacity of S servers on the big data cloud platform; E(S) Energy consumption of S servers as a big data cloud platform; F(S) refers to the abnormal loss of S servers on the big data cloud platform.

3. Analysis of experimental results
In order to verify the rationality of the big data cloud platform server load balancing algorithm, experiments were carried out with two clients and three servers. Load balancing system environment is Windows XP system and MRTG traffic monitoring software installed in the load balancer and SNMP collection server running in the server load distribution program. Each server runs a program that communicates with the load balancer. The CPU utilization rate, network utilization rate, memory utilization rate and other parameters are collected as the test performance indicators. The software environment for this experiment is shown in the table below.

| Name                      | Parameter          |
|---------------------------|--------------------|
| Operating system          | -                  |
| Development tool          | Win XP             |
| Development language      | Visual Studio.NET.3.5 |
| System operating environment | C++/JAVA          |
| Browser                   | 360 browser /EI browser |
| Performance monitoring software | MARTG         |

From the idea of minimum connection scheduling algorithm, when a task is assigned to a server, it needs to meet a condition, that is, the connection established by the server keeps a certain ratio with the current connection of the server. However, through analysis, we can find that the connection number of a server can only be used to approximate the load of the server, but can not accurately represent the current load value of the server. If the connection number of a server is very large at this time, it will lead to the accumulation of connection number due to the decrease of processing capacity. Therefore, reassigning new tasks to the server will increase the server's load This is detrimental to task processing and system balance. At this time, if you can monitor the status of the server and allocate the load to the server according to its actual state, or suspend the allocation of new load to it, the processing capacity of the server can be buffered to a certain extent, so as to ensure the performance of the server. Based on this, the comparative test results of the two methods are recorded, as shown in the following figure:

![Figure 2 Comparison test results](image)

The research shows that compared with single server, cluster system can improve the server
environment to a great extent, and has strong practicability and robustness. The two algorithms are obviously different. The method does not distinguish task types, and the proportion of task allocation is close to the calculation result. The algorithm is basically suitable for the network tasks assigned by nodes, and the calculation amount is proportional to the CPU computing power of nodes. This method can balance the load of nodes, solve the problem of uneven load distribution, improve the efficiency of load balancing, and increase the stability and flexibility of cluster system.

4. Conclusion
In the environment of big data cloud platform, server performance has gradually become a bottleneck problem. This paper analyzes the server cluster technology and its characteristics in detail. According to the characteristics of cluster, using multi server technology can not only effectively solve the problem of increasing the number of users and network traffic, but also effectively reduce the cost input and cost. The principle of load balancing technology, the existing load balancing technology and algorithm are analyzed and studied in depth. Based on the analysis of server network path load balancing technology and the minimum connection number weighted scheduling algorithm, this paper proposes a big data cloud platform server load balancing algorithm based on improved chaotic partition algorithm, and gives its implementation process. Combined with the server load change, it realizes reasonable resource allocation and load balancing. Chaotic partition algorithm is also an application of multi-objective algorithm, such as robot path planning, load balancing algorithm can provide specific ideas.

Acknowledgements
The work is supported by College Student Innovation and Entrepreneurship Training Program of China under Grant No.201813242004, and University-level scientific research project of Wuhan Technology and Business University under Grant No.A2016009.

References
[1] Kang X.B., Lin G.F., Chen Y.J., et al. “Robust and secure zero-watermarking algorithm for color images based on majority voting pattern and hyper-chaotic encryption”. Multimedia Tools and Applications, vol.79, no.1, pp.1169-1202, 2020.
[2] Saxena A., “A comprehensive study of chaos embedded bridging mechanisms and crossover operators for grasshopper optimisation algorithm”. Expert Systems with Applications, vol.132, no.10, pp.166-188, 2019.
[3] Lloyd D., Murray D.B., Aon M.A., et al. “Temporal metabolic partitioning of the yeast and protist cellular networks: the cell is a global scale-invariant (fractal or self-similar) multioscillator”. Journal of biomedical optics, vol.24, no.5, pp.051404.1-051404.17, 2019.
[4] Kaur M., Singh D., Sun K., et al. “Color image encryption using non-dominated sorting genetic algorithm with local chaotic search based 5 D chaotic map”. Future Generation Computer Systems, vol.107, no.5, pp.333-350, 2020.
[5] Ibrahim R A., Abd Elaziz M., Lu S., “Chaotic Opposition-Based Grey-Wolf Optimization Algorithm based on Differential Evolution and Disruption Operator for Global Optimization”. Expert Systems with Applications, vol.108, no.10, pp.1-27, 2018.
[6] Wang X., Wang Y., Zhu X., et al. “A novel chaotic algorithm for image encryption utilizing one-time pad based on pixel level and DNA level”. Optics and Lasers in Engineering, vol.125, no.2, pp.105851.1-105851.12, 2020.
[7] Rajagopalan A., Kasinathan P., Nagarajan K., et al. “Chaotic self daptive interior search algorithm to solve combined economic emission dispatch problems with security constraints”. International Transactions on Electrical Energy Systems, vol.29, no.2, pp.2026, 2019.
[8] Yousri D., Babu T. S., Allam D., et al. “A Novel Chaotic Flower Pollination Algorithm for Global Maximum Power Point Tracking for Photovoltaic System Under Partial Shading Conditions”. IEEE Access, vol.7, no.6, pp.121432-121445, 2019.
[9] Hamdy M., Shalaby R., Sallam M., “Experimental verification of a hybrid control scheme with chaotic whale optimization algorithm for nonlinear gantry crane: A comparative study”. ISA Transactions, vol.98, no.4, pp.418-433, 2020.

[10] Tayal A., Kose U., Solanki A., et al. “Efficiency analysis for stochastic dynamic facility layout problem using meta-heuristic, data envelopment analysis and machine learning”. Computational Intelligence, vol.36, no.1, pp.172-202, 2020.

[11] Wang H., Wu Q. M. J., Wang J., et al. “Optimizing simple deterministically constructed cycle reservoir network with a Redundant Unit Pruning Auto-Encoder algorithm”. Neurocomputing, vol.356, no.9, pp.184-194, 2019.

[12] Liu C., Li K., Liang J., et al. “Service Reliability in an HC: Considering From the Perspective of Scheduling With Load-Dependent Machine Reliability”. IEEE Transactions on Reliability, vol.68, no.2, pp.476-495, 2019.