ISSN: 2664-9640

An Optimization of dynamic load balancing algorithm

Xin Han[1, a], Lele Ren[2] and Xiangsheng An[1]

[1] School of Xi'an Jiaotong University, Xi'an, China;

[2] School of Xi'an University of Posts and Telecommunications, Xi'an, China.

[a]1043116928@qq.com

Abstract

The high traffic of a large number of Internet users has put enormous pressure on Internet software. In order to solve the load pressure on servers, enterprises choose to build a cluster system to handle user requests. Cluster systems need to use load balancing strategy to select one server in the cluster to handle requests. Hardware load balancing strategy has not been widely used because of its high cost and poor iteration.Software load balancing strategies are divided into two main methods: static load balancing and dynamic load balancing. Static load balancing strategies do not consider the state information of the server, and assign tasks with a fixed probability. Dynamic load balancing strategies are selected according to the real-time performance of the server. This paper proposes an optimized dynamic load balancing strategy, which considers the performance of the server and the needs of users. The experiment shows that on the basis of meeting the needs of users, the strategy has also been improved in other aspects. In terms of throughput and response time, it achieves better results than polling algorithm and minimum number of connections algorithm.

Keywords

Load balancing, user demand, server performance, web server cluster.

1. Introduction

With the popularization and development of the Internet, the number of Internet users and Internet penetration rate are increasing day by day. The large number of Internet users undoubtedly brings huge network access to Internet software, but the increase of access also brings huge load pressure to the back-end server. In the face of such high concurrent access demand, a single server device is difficult to bear. At present, the common method is to build a server cluster to improve the system performance to deal with the huge demand of access. Server cluster will form a cluster system with multiple servers to provide users with unified network services, which can increase the concurrent processing capacity of the system and the error redundancy capacity of the system in case of failure of a single machine, effectively avoid that the cluster system cannot provide services normally when a single server is not running normally, and realize the high reliability of the system and the high availability of resources[1]. How to distribute user requests to subordinate servers in a server cluster, load balancing strategy will solve this problem.

The early method was to use DNS as a load balancing policy, and then make the client's traffic directly reach each server by resolving different IP addresses to the client. However, one big disadvantage of this method is the delay problem. The scheduling policy of DNS load is relatively simple,after the change of the scheduling policy, the caches of DNS nodes at all levels will not take effect on the clients in time and can not meet the business needs.Load balancing can solve this problem well,load balancing strategy can scientifically select the appropriate server to process the task. Static load balancing strategies include polling and IP_ hash、url_

ISSN: 2664-9640

hash and other methods, these strategies do not take into account the real-time state of the server, and the server may be overloaded but still required to process requests. Dynamic load balancing strategies often only consider one or serval performances of servers in the cluster, and both strategies do not consider the needs of users.

Literature [2] adopts a hybrid load balancing algorithm, which combines polling algorithm and dynamic method. It maintains two waiting queues for short requests and long requests respectively and executes them separately. At the same time, it balances the waiting time of requests in the queue, which is more friendly to long requests and conducive to scheduling fairness. However, the efficiency and response time of the algorithm need to be improved.

Literature [3] studies load balancing and Web caching solutions in some typical cases, focusing on how to design solutions combined with nginx and its modules.

Reference [4] proposed an efficient load balancing and performance optimization scheme (LBS) to solve the problems of throughput, long-term connectivity, end-to-end delay and energy efficiency in resource constrained WSNs.

The optimization strategy proposed in this paper will select the server to process the request combined with various performance indicators of the server, the comprehensive performance of the server and the demand of users.The main work is as follows.

Considering the demand degree of different users, the ratio of user demand degree to the response time of the request is taken as the processing result value.

CPU surplus rate, memory surplus rate, disk I/O surplus rate and network bandwidth surplus rate are selected as performance indicators. Calculate the comprehensive weight of all servers through the weight of four indicators and the value of four indicators, and then select the appropriate server according to the comprehensive weight of each server.

If only the comprehensive weight is considered, the residual rate of some performance may be very low, so set a threshold for each of the four indicators. If the four performance values of the server are more than their thresholds, add the server to the light load queue, and then select the server with the highest comprehensive performance from the light load queue.If a server's partial performance value is less than or equal to the performance threshold, the server is added to the heavy load queue. When the light load queue is empty, select the server with the highest comprehensive performance from the heavy load queue.

2. Related work

2.1. User demand

On the basis of considering the performance of the server, a user requirement indicator is added. The system will give priority to users with high demand, the value of requirement will be from low to high 1~5, and then the ratio of requirement value to response time will be used as the result value of processing requests.That is:

$$value = priority / (endtime - starttime) \quad (1)$$

2.2. Server Performance Indicators

Dynamic load balancing strategy needs to consider real-time server performance. The server has many parameters. When selecting indicators, we should follow two principles: 1) the selected parameters can truly reflect the performance of the server. 2) There should be no excessive additional loss when obtaining server parameter data[5]. Based on these two criteria, the method selects CPU surplus rate, memory surplus rate, network bandwidth surplus rate and disk IO surplus rate as the server load evaluation indexes. Firstly, consider whether the four indicators exceed the corresponding threshold. Secondly, combine the weight of each indicator and the value of each indicator to calculate the server's comprehensive weight,

ISSN: 2664-9640

comprehensively measure the load status of cluster servers when they are running[6]. The formula for calculating the weight of server C_i is as follows:

$$WS(Ci) = W(cpu) * R(cpu) + W(meo) * R(meo) + W(net) * R(net) + W(io) * R(io) \quad (2)$$

In addition, the configuration of servers in the cluster is also different. Compared with servers with high configuration and low performance surplus rate, servers with low configuration and high performance surplus rate may have relatively low ability to process requests[7]. Therefore, each server needs to have its own weight ($W(C_i)$) according to the proportion of its own configuration in the cluster configuration. The final weight of server C_i is:

$$Q(Ci) = WS(Ci) * W(Ci) \quad (3)$$

2.3. Analytic Hierarchy Process

Analytic hierarchy process (AHP) is a systematic method that takes a complex multi-objective decision-making problem as a system, decomposes the objective into multiple objectives or criteria, then decomposes it into multiple indicators (or criteria and constraints) at multiple levels, and calculates the single ranking of levels. Calculates the hierarchical single ranking (weight) and total ranking through the qualitative index fuzzy quantitative method, so as to be used as the objective (multiple indicators) and multi scheme optimization decision-making[8]. Analytic hierarchy process decomposes the decision-making problem into different hierarchical structures according to the order of general objectives, sub objectives of each level, evaluation criteria and specific standby investment scheme, and then obtains the priority weight of each element of each level to an element of the upper level by solving the eigenvector of the judgment matrix, Finally, the weighted sum method is used to merge the final weight of each alternative scheme to the overall goal, and the best scheme is the one with the largest final weight[9]. The analytic hierarchy process is suitable for decision-making problems with staggered evaluation index layers, and it is difficult to quantitatively describe the target value.

The weight coefficient of the index is calculated in three steps.

First, you need to build a hierarchical analysis model. Considering the load evaluation as the target layer and the CPU surplus rate, memory surplus rate, network bandwidth surplus rate, and disk IO surplus rate indicators as the criterion layer. The hierarchical analysis model structure of this paper is shown in Figure 1.

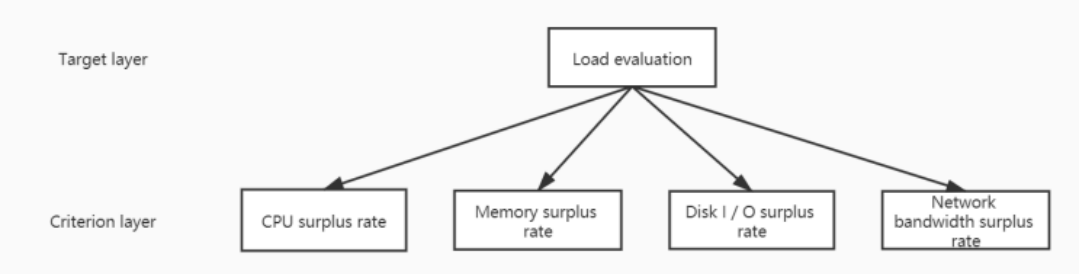

Fig.1 Hierarchical Analysis Model

Using the "1-9" scale method, a judgment matrix is constructed based on the server test situation. The 1-9 scale table is shown in Table 1.

TALBE 1: "1-9" scale table

Scale	Meaning
1	Indicates that the two factors have the same importance compared to each other
3	Indicates that the former is slightly more important than the latter
5	Indicates that the former is obviously more important than the latter

ISSN: 2664-9640

7	Indicates that the former is strongly more important than the latter
9	Indicates that the former is extreme more important than the latter
2,4,6,8	Represents the median of the above adjacent judgements

A judgment matrix is constructed based on the server test. The importance ratio of performance indicators is shown in Table 2. During the actual server test, it is found that the importance of CPU surplus rate and memory surplus rate is high, while the importance of network bandwidth and disk IO surplus rate is relatively low. Therefore, the importance of CPU surplus rate, memory surplus rate is set to 3, network bandwidth and disk IO surplus rate is set to 1.

TALBE 2: Importance ratio table of performance indicators

	CPU	Mem	DiskIO	Net
CPU	1	1	3	3
Mem	1	1	3	3
diskIO	1/3	1/3	1	1
Net	1/3	1/3	1	1

Comparison table 3 can be obtained by adding matrix columns and normalizing them according to formula (4).

$$q_{ij} = \frac{q_{ij}}{\sum_{i=1}^{n} q_{ij}} (i, j = 1,2,\ldots,n) \tag{4}$$

TALBE 3: Importance ratio table after column addition and normalization

	CPU	Mem	DiskIO	Net
CPU	0.375	0.375	0.375	0.375
Mem	0.375	0.375	0.375	0.375
diskIO	0.125	0.125	0.125	0.125
Net	0.125	0.125	0.125	0.125

After AHP calculation, the weight formula of server C_i is as follows:

$$WS(Ci) = 0.375 * R(cpu) + 0.375 * R(meo) + 0.125 * R(net) + 0.125 * R(io) \tag{5}$$

2.4. Server weight strategy design

As shown in Figure 2, four modules constitute the strategy. This load balancing strategy is divided into four modules: information collection module, weight calculation module, demand ranking module and balanced scheduling module. The information collection module is responsible for collecting the performance of each server regularly. The weight calculation module is responsible for calculating the weight of each server. The demand sorting module sorts the demand according to the product of the demand degree of the demand and the waiting time of the demand. The balanced scheduling module selects the server in the light load queue or heavy load queue according to the comprehensive weight of the server.

ISSN: 2664-9640

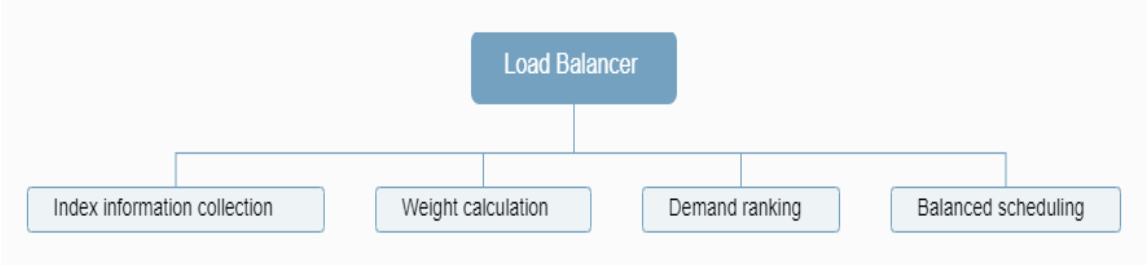

Fig.2 Frame diagram of algorithm

There are several ways to collect server performance data: 1. Real time data collection: synchronize the performance data of the server at all times, with high accuracy, but excessive loss. 2. Select some servers in the cluster for collection: the loss is very small, but not all servers are fully utilized. 3. Regular collection: determine a fixed period, and then collect the performance data of each server regularly. The strategy mentioned in the article will collect information modules on the basis of regular collection.

Because the server may have high comprehensive weight, but some performance is very low, this method first considers the residual performance of the four indicators of the server, second considers the total weight of the server. Initially, a threshold is set for each of the four indicators, then two queues are created. If the four indicator data of the server are lower than the threshold, enter the light load queue, and select the server with the largest weight in the light load queue to process the request. When the data of some indicators of the server is higher than the threshold, the server enters the overload queue. When the request arrives, the load balancing strategy will give priority to the server with the largest comprehensive weight in the light load queue. If the light load queue is empty, select the heavy load queue, and select the server with the largest weight in heavy load queue. Referring to the practical application in the project, the thresholds of the four indicators are obtained according to the specific situation of the server running. Set the threshold of CPU surplus rate to 25%, the threshold of memory surplus rate to 20%, the threshold of disk I / O surplus rate to 10%, and the threshold of network bandwidth surplus rate to 20%.

3. Experiment

The algorithm mentioned in this paper is optimized on the basis of weighted polling algorithm, and a dynamic load balancing algorithm is obtained. This experiment is based on nginx. In this experiment, we select the polling algorithm, the minimum number of connections algorithm and the dynamic optimization algorithm mentioned in this paper to experiment. Requests are distributed to the back-end server cluster, and response time and throughput of the three scheduling algorithms are calculated for the cluster system under different concurrent requests. The larger the throughput, the shorter the response time, which indicates that the algorithm has stronger processing concurrency ability. Five experiments were conducted and the average values of the experimental data were recorded.

The load information collection module and the weight calculation module are implemented based on Java language, the user demand ranking module and the balanced scheduling module are written based on C language, and the equalizer scheduling module is an improvement of the third-party upstream module in the nginx source code and compiled into nginx. The above four modules work together to complete the dynamic adaptive load balancing scheduling of client requests.

3.1. Test Tools

Apache JMeter is a Java based stress testing tool developed by Apache organization. It was originally designed for web application testing, but later extended to other testing fields. It can

ISSN: 2664-9640

be used to test static and dynamic resources, such as static files, Java applets, Java objects, databases, and so on. JMeter can be used to simulate huge loads on servers, networks or objects, test their strength and analyze the overall performance under different stress categories.

JMeter is a stress testing and evaluation tool written for web developers and system administrators to test program and system performance under pressure. It is often used to test the tolerance of applications under high concurrency pressure. It can simulate multi-user concurrent access through configuration, allow users to customize the number of users and the number of users' cyclic access, simulate users' stress test on URLs, and simulate users' concurrent access to web servers.

3.2. Experimental Environment

Deploy nginx machines in the windows environment and select three machines as background servers to process user requests. The configuration of the four machines is shown in table 4.

Table 4: Three Scheme comparing

Server	Server1	Server2	Server3	Nginx Server
CPU	1 core	1 core	2 core	4 core
Memory	2 GB	2 GB	4 GB	4 GB
Network rate	1000 Mbps			
Disk capacity	20 GB			
Operating System	Ubuntu 18.04			
JDK	v1.8			
Nginx	v1.16.0			
Tomcat	V7.0.39			

3.3. Experimental design

The information collection module can collect real-time performance data of all back-end servers.

CPU surplus rate: obtain CPU information through “cat/ proc/stat” command and obtain CPU information in a short period of time. The total time obtained for the first time is defined as t1 and idle time is idle1. The total time obtained for the second time is defined as t2 and idle time is idle2. CPU utilization is (idle2-idle1) / (t2-t1).

Memory surplus rate: obtain memory related data through “cat / proc / meminfo” command.

(3) Network bandwidth surplus rate: obtain the total size of network packets received and sent by the network port through “cat/proc/net/dev” command.

(4) Disk IO surplus rate: “iostat -x -d -k 1" command can get disk real-time data.

Weight calculation module can select the server to be weighted by calculating and analyzing the results of information collection. User demand ranking module can estimate the benefits of processing user needs, and give priority to the requests with high benefits. The load scheduling module can weight the selected server according to the previous module results and process the corresponding requests.

3.4. Experimental results and analysis

This test is a stress test on the polling algorithm, the minimum number of connections algorithm and the dynamic optimization algorithm considering user demand mentioned in this paper, and

ISSN: 2664-9640

compares and analyzes the request response time and throughput under different concurrent requests. For the convenience of writing, the polling algorithm is abbreviated as RR, the minimum number of connections algorithm is abbreviated as LC, and the load balancing algorithm mentioned in this paper is named dynamic.

3.4.1. Request response time

According to the response time of the server cluster to the client request processing, take the response time of all requests under each test to record, and the performance difference of the scheduling algorithm can be most intuitively reflected. The shorter the request response time, the stronger the concurrency of the system. Through the stress test tool, simulate and test the concurrency from 500 to 5000 in steps of 500, analyze the response time of requests under different concurrent requests, and the request response time of different load balancing algorithms under different concurrent requests is shown in the table 5.

TALBE 5: Response time comparison table of three algorithms

concurrent connections	response time(ms)		
	Rr	Lc	Dynamic
500	374	409	382
1000	720	683	701
1500	1029	1045	1013
2000	1354	1350	1320
2500	1866	1986	1765
3000	2113	2031	1931
3500	2315	2251	2354
4000	2836	2960	2720
4500	3503	3316	3267
5000	4866	4574	4176

According to the request response time data of different algorithms under different concurrent requests, the response time comparison line chart is drawn as figure 3.

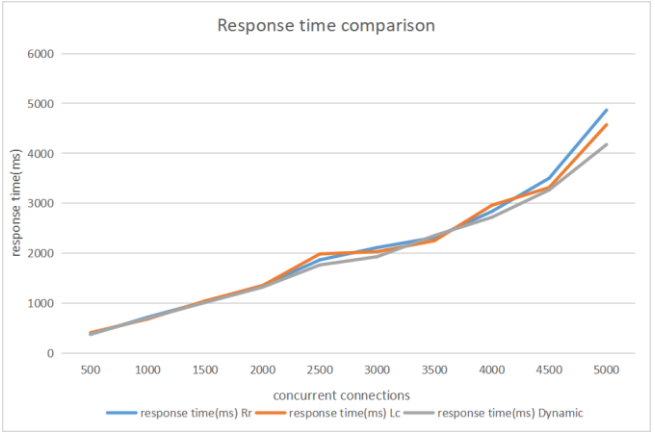

Fig.3 Response time comparison chart of three algorithms

According to the experimental data, the load balancing algorithm optimized in this paper is 6.4% better than polling algorithm and 4.7% better than minimum number of connections algorithm in terms of average response time of requests.

ISSN: 2664-9640

3.4.2. Throughput

The higher the throughput, the higher the efficiency of the server in processing requests, indicating that the server has stronger processing capacity in dealing with high concurrent requests. The throughput of different load balancing algorithms under different concurrent requests is shown in table 6.

TALBE 6: Throughput comparison table of three algorithms

concurrent connections	Throughput(per second)		
	Rr	Lc	Dynamic
500	138.8	193.7	149.4
1000	129.4	136.6	137.8
1500	116	129.2	124.5
2000	136.6	143.5	136.4
2500	139.4	147.3	152.4
3000	147.8	163	167.8
3500	118.1	172.4	189.3
4000	187.6	193.7	204.7
4500	152.4	219.9	232.6
5000	167.5	231.6	249.7

According to the throughput data of different algorithms under different concurrent requests, the throughput comparison line chart is drawn as figure 4.

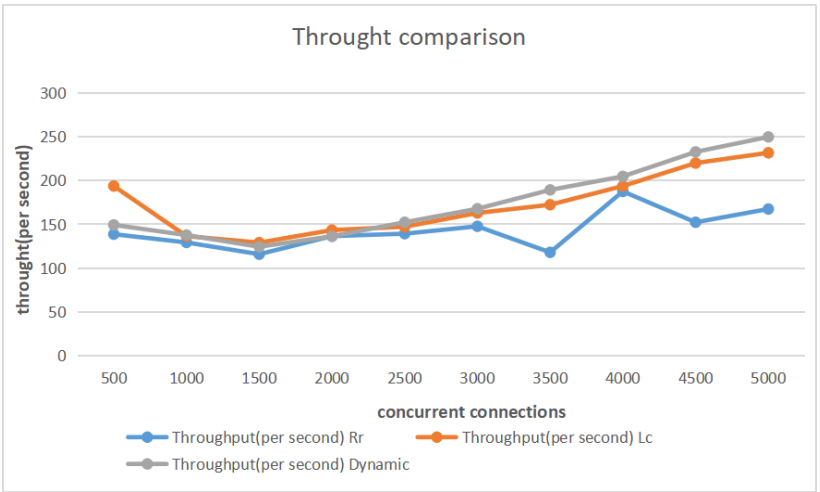

Fig.4 Throughput comparison chart of three algorithms

According to the experimental data, the optimized load balancing algorithm in this paper improves and stabilizes throughput compared with other algorithms, 21.6% higher than the polling algorithm, and 0.79% higher than the minimum number of connections algorithm.

3.4.3. Results of user demand

Experiments are carried out to compare the result values of demand handling under different number of concurrent requirements. The result value reflects whether the algorithm meets the needs of the overall user. If the server responds to the task request, while considering the server load, it can also consider the characteristics of the request itself and the different needs of different users to initiate requests, it is conducive to better improve user satisfaction and improve the overall satisfaction of the service.

ISSN: 2664-9640

A user's request carries the requirement that the user fills in, which ranges from 1 to 5 from low to high. The ratio of the requirement to the response time of the request is used as the result value of the user's requirement.

The result values of user demand degree of different algorithms are shown in table 7.

TALBE 7: Comparison table of user demand results of three algorithms

concurrent connections	Results of user demand		
	Rr	Lc	Dynamic
500	1365	772	1447
1000	2634	2550	3036
1500	3847	3901	4451
2000	4944	5430	5743
2500	5411	5692	6913
3000	6107	6507	7742
3500	6695	6828	8325
4000	6773	7575	8600
4500	6640	6885	8412
5000	6691	6965	8837

According to the result value data of user satisfaction of different algorithms under different number of concurrent requests, the comparison line chart of user satisfaction result values is drawn as figure 5.

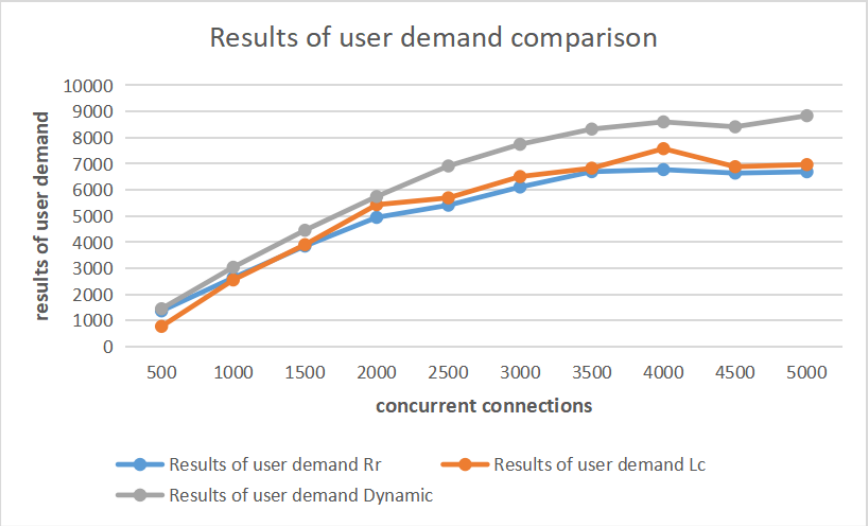

Fig.5 Comparison chart of user demand results of three algorithms

According to the experimental data, the optimized load balancing algorithm in this paper meets the needs of the whole user better than other algorithms, 24.2% higher than the polling algorithm, and 16.3% higher than the minimum number of connections algorithm.

For the dynamic load balancing algorithm considering user demand proposed in this paper, the stress test tool JMeter is used for stress test, and the stress test results are compared with those of polling algorithm and minimum connection algorithm. The test experimental results are analyzed from three aspects: request response time, throughput and the result value of user demand. The test data show that the load balancing algorithm proposed in this paper is effective and feasible.

ISSN: 2664-9640

4. Conclusion

With the growth of the number of Internet users and Internet penetration, the requirements for Internet software are also increasing. How to effectively use cluster resources to provide services for a large number of users has become the focus.

This paper proposes a dynamic load balancing algorithm based on user demand and weighted polling algorithm. Periodically collect the information of CPU surplus rate, memory surplus rate, disk I / O surplus rate and network bandwidth surplus rate of each back-end server. The weight coefficients of each load index of the server are obtained by analytic hierarchy process, and the comprehensive weight of each back-end server is calculated by using the server weight formula, and finally the real-time weight of each back-end server is determined. Estimate the result value of user demand degree, then give priority to the demand with high result value in order to obtain high overall satisfaction. Build a performance test environment, use JMeter to simulate the client to send high concurrent access requests, and the load balancer receives and forwards the access requests to the back-end server. Test the dynamic load balancing algorithm proposed in this paper, polling algorithm and minimum connection algorithm, and count the test results, The test results show that the dynamic load balancing algorithm proposed in this paper is feasible and effective.

References

[1] RAMANA K, PONNAVAIKKO M, SUBRAMANYAM A. A global dispatcher load balancing approach for a web server cluster［M］. Singapore：Springer Singapore, 2019.

[2] Samir Elmougy,Shahenda Sarhan,Manar Joundy. A novel hybrid of Shortest job first and round Robin with dynamic variable quantum time task scheduling technique[J]. Journal of Cloud Computing,2017,6(1):

[3] Xiaoni Chi.Web Load Balance and Cache Optimization Design based Nginx Under High-concurrency Environment[C].Proceedings of the 3nd International Conference on Digital Manufacturing & Automation,2012:2447-2450.

[4] Rahim Khan. An efficient load balancing and performance optimization scheme for constraint oriented networks[J]. Simulation Modelling Practice and Theory,2019,96:

[5] TEEGALA S K, SINGAL S K. Optimal costing of overhead power transmission lines using genetic algorithms［J］. International Journal of Electrical Power ＆ Energy Systems, 2016, 83：298-308.

[6] YAN Y, HE L, MENG Q. Exploration and improvement in keyword extraction for new s based on TFIDF［J］. Energy Procedia, 2011(13) : 3551—3556.

[7] BISWAS S K, BO R DOLOI M, SH R EYA J. A graph based keyword extraction model using collective node weight［J］. Expert Systems with Applications, 2018, 97: 51—59.

[8] YAN Y, TAN Q, XIE Q, et al. A graph—based approach of automatic keyphrase extraction［J］. Procedia Computer Science, 2017, 107: 248—255.

[9] CHI X, LIU B, NIU Q, et al. Web load balance and cache optimization design based nginx under high—concurrency environment［C］//2012 third international conference on digital manufacturing and automation (ICDM A) . ［s. l. ］: ［s. n. ］, 2012: 62—67.