International Journal for Housing Science and Its Applications
Publish August 3, 2025. Volume 46, Issue 3 Pages 944-952

https://doi.org/10.70517/ijhsa46371

Optimization Design of Cloud Computing Network Based on Load Balancing

Zhihuang Jiang[1,*]
[1] College of Data and Computer Science, Guangdong Peizheng College, Guangzhou, Guangdong, 510830, China
Corresponding authors: (e-mail: j88289876@hotmail.com).

Abstract Cloud computing is a kind of service system mainly controlled by cloud data centers, and the increasing scale of information transmission puts forward higher requirements on its scheduling ability. This paper takes rationalization of resource scheduling as the research objective and launches the research on cloud computing network load imbalance problem. By analyzing the network resource load based on processing time, a cloud computing network load balancing model is constructed. It also proposes a dynamic load balancing strategy for cloud computing network. The strategy utilizes the distributed computing and storage capabilities of the cloud computing platform to reasonably migrate virtual machines online. In this way, it ensures that the load of each server tends to be balanced, so as to realize the dynamic balance and control of cloud computing network load. Under a variety of experimental environments, the model proposed in this paper not only optimizes the average system response time by more than 30%, but also tends to be smoother. It shows that the model can improve the effectiveness of the data analysis problem in the communication process, and then optimize the cloud computing network load balancing.

Index Terms cloud computing network, virtual machine, load balancing, resource scheduling, dynamic balancing

I. Introduction

With the wide application of cloud computing technology, cloud computing network architecture has become a hot spot for research. Cloud computing network architecture is one of the key technologies to realize cloud computing services, which can provide efficient resource management and data transmission [1]. However, existing cloud computing network architectures have some problems, such as performance bottlenecks, security issues, scalability and reliability issues, and efficiency and cost issues [2], [3].

Load balancing is an important technique to optimize cloud computing network architecture by spreading network traffic across multiple servers to fully utilize the computing power of the servers and improve the overall performance of the network [4], [5]. Kumar, C et al. proposed a load balancing mechanism based on dynamic pricing and task allocation for and optimization of cloud computing networks [6]. Shafiq, D. A et al. optimized the resource utilization of cloud computing based on load balancing algorithm using factors such as QoS task parameters, VM priority and resource allocation as constraint variables to improve the load balancing of cloud computing applications [7]. Komathi, A et al. used Support Vector Machine (SVM) to classify the data and load balancing using Ant Colony Algorithm based on the performance metrics such as execution time, throughput, overhead, optimization time and migration counts [8]. Hasan, R. A and Mohammed, M. N. Load balancing across virtual machines using Krill Load Balancer (Krill LB) algorithm introduces an improved dynamic energy-aware model that incorporates energy costs into the load balancing process for optimizing resource allocation and energy efficiency in cloud computing environments [9]. Gao, R and Wu, J constructed a new method for dynamic load balancing in cloud computing based on ant colony optimization technique which includes forward-backward ant mechanism, max-min rule and task execution prediction to accelerate the search process [10]. Alghamdi, M. I incorporates artificial neural networks into two-particle swarm optimization algorithms for optimizing load balancing and task scheduling in cloud computing environments [11]. Gamal, M et al. constructed a new hybrid meta-heuristic load balancing algorithm for cloud computing which combines percolation computing with ant colony optimization in order to improve load balancing and reduce energy consumption for network optimization [12]. Khan, A. R combines deep learning, reinforcement learning and hybrid optimization techniques to construct a new dynamic load balancing approach for cloud computing to optimize task scheduling and resource utilization in cloud computing environments in order to improve network performance in cloud computing [13].

As the amount of data handled by cloud computing platforms continues to increase, the network congestion problem becomes more and more serious. Network congestion can lead to data transmission delays, affecting the responsiveness and overall performance of applications [14]. However, existing load balancing algorithms often fail to provide differentiated quality of service guarantees, resulting in certain applications failing to meet their performance requirements [15]. In cloud computing network architecture, traffic scheduling is one of the key factors affecting performance [16]. Since different applications have different requirements and loads, efficient traffic scheduling algorithms are needed to improve the overall performance of traditional load balancing algorithms.

In this paper, we first analyze the cloud computing network resource load and elaborate the calculation of network resource load characteristics to provide a data foundation for load task scheduling. Based on this data preparation, a cloud computing network load balancing model is established. Subsequently, it proposes a dynamic load balancing task scheduling strategy for cloud computing network in view of the security problems such as data leakage that may occur in the communication process of cloud computing network. It also explores the way of migrating running virtual machines and the selection of target servers to realize the cloud computing network load balancing. Finally, the performance, throughput and convergence experiments of the cloud computing network load balancing model under different experimental conditions are carried out successively to test the effectiveness of the model and algorithm.

II. Cloud Computing Network Load Balancing Model

II. A.Network Resource Load Analysis Based on Processing Time

Assume that the communication network environment contains a total of n physical nodes and that all nodes are involved in the transmission of resources, corresponding to a total number of transmission tasks of R, and that there are any of these nodes $x_i\left(i \le n\right)$ with an average processing time of t_{ct} for the transmission tasks, and that the running time required to complete a transmission task is t_{yt}. Then within n nodes, there exists a physical node whose presence processing is greater than the critical time for communication transmission, i.e., the upper limit of timeout customized by the network, and the computation for this class of nodes can be expressed as equation (1):

$$\frac{f(t_{ct})+f(t_{yt})}{n} \ge T \tag{1}$$

where $f(*)$ denotes the critical function, and T denotes the upper limit of network customized transmission timeout, it is not difficult to see that the function of determining the timeout node is a monotonically decreasing function by observing equation (1). On this basis, in order to fully utilize the transmission space of network resources and effectively improve its transmission capacity for parallel tasks, this paper takes the node whose judgment result is not timeout as the target node for load scheduling, and its corresponding transmission channel is also the load target scheduling channel, while the load to be scheduled comes from the node with timeout and the corresponding channel.

At this point there is a load balancing problem, as the node load scheduling occurs, the node's processing time and running time will change dynamically accordingly, which makes the scheduled node may be transformed into a scheduling node, for this reason, this paper sets a temporary variable a for each physical node, and there is an initial value of a as $f(t_{ct})+f(t_{yt})$, which serves to provide real-time feedback on the ability of this target physical node to handle network resource transfer tasks as well as load balancing. Then there are, when the unfinished transmission tasks at the node are transmitted at the current processing speed, the time to complete the amount of r_i tasks is equation (2):

$$T_i = \frac{r_e}{ar_i} \tag{2}$$

where T_i denotes the time for x_i node to complete r_i amount of tasks and r_e denotes the total amount of tasks it has transmitted. And with the scheduling of the load, the change in the value of a corresponding to the node can be expressed as equation (3):

$$a = \frac{r_i - e - \dfrac{r_e \Delta T}{a}}{D} \tag{3}$$

where e denotes the load size of the scheduling, i.e., the amount of transmission tasks, and e takes a positive value when the scheduling form is outgoing, and a negative value when the scheduling method is incoming. ΔT denotes the time overhead of the scheduling process and D denotes the bandwidth of the channel where the node is located. In this way, the dynamic characteristics of the network resource load are calculated to provide a data base for subsequent scheduling.

II. B.Network Load Balancing Modeling

In the communication network link layer, link load imbalance will produce link congestion and packet loss. At the cloud data center end, collect and apply cloud network security server and client terminal data, combine with security cloud computing, and build a network load balancing model. Setting link load balancing policy preferences, according to different bandwidth usage rate mapping preferences, routing control of the load volume, to complete the communication network load balancing in the cloud computing service platform. For a network topology of number j, there are $\frac{j}{2}$ networks in parallel, which makes a large number of redundant links during data transmission. Packets are extracted during data stream exchange, they are reasonably matched, and the data streams are operated according to the corresponding commands, and the packets are operated accordingly until all tables have completed the corresponding matching. Load dynamics are detected, and load detection requests are sent to the switches in the data center network according to the flow dynamics. In response to the received load information, the load rate is calculated and the collected information is sent to the network load consistency. Based on the stream rate and the number of ports, the bandwidth utilization is calculated and then the bandwidth utilization is converted into network preference. When the functionally distributed network receives the stream request message sent by the data stream, the functionally distributed network gets the forwarding line for the load volume according to the network preference for the line setting. Transmitting network data information makes the functional distributed network console collect information about the network data and then passes that information to the data stream center. The functional distributed network console uses active transmission to transmit periodic data and sends default messages to get the data flow connection status and generate a real-time network topology map. Since the load in the network is constantly changing, the global topology of the network is updated slowly when the rate of the data streams changes if the load accumulates over a long period of time. When the rate of data flow is basically unchanged, too short accumulation time may consume a lot of resources. Therefore, for the problem of detecting dynamic changes in the amount of load, the degree of network state change is constantly monitored based on the cumulative time size. The cumulative time size Δy is related to the rate change Δx of the data stream and also to the size of the previous cumulative time $\Delta y'$. It is assumed that the size of the first accumulation time is the minimum accumulation time $\min \Delta y$ and the maximum accumulation time is expressed as $\max \Delta y$. The formula for calculating the load accumulation time is equation (4):

$$\Delta y = \begin{cases} \min\left(\Delta y' + 1, \max \Delta y\right) \\ \Delta y' \times \Delta x \\ \max\left(\Delta y' - 1, \min \Delta y\right) \end{cases} \tag{4}$$

Two threads are opened for load load volume monitoring. In the design of the load balancing strategy, the application uses the Softmax function, when the network bandwidth utilization is small, for the network link greatly selected probability. When the network bandwidth utilization is large, very small selection probability for the link. Through the establishment of load balancing model can realize the communication network resource node utilization rate gradually converges to the average value, to achieve the goal of completing the communication network load balancing.

III. Dynamic Load Balancing for Cloud Computing Networks

The computing unit of cloud computing technology is virtual machine, by seamlessly migrating the running virtual machine between 2 servers, it can realize the dynamic balancing of network load. This chapter focuses on the principles of the strategy and the implementation of dynamic load balancing in cloud computing.

III. A. Dynamic Load Balancing Task Scheduling Policy

Dynamic load balancing task scheduling for networks using cloud computing platforms. As an on-demand service platform, cloud computing platform needs to have good service quality. The task deadline violation rate is selected as an important index for the performance service evaluation of cloud computing platform. The priority of the pending load balancing task is denoted by P, and its calculation formula is shown in equation (5):

$$P = T_d - T_a \tag{5}$$

where T_d and T_a are load balancing task deadline and load balancing task arrival time, respectively. The smaller the calculation result of Eq. (5), the higher the priority of the corresponding load balancing task.

The computing power required by the cloud computing platform to perform the load balancing task is expressed in equation (6):

$$s_i = s_l / P \tag{6}$$

where: S_i is the cloud computing platform execution load function, and S_l is the load balancing task length.

Calculate the average CPU resource, average memory resource, and average bandwidth resource of all load balancing tasks within the load balancing task set, and compare the information of each load balancing task. When the CPU resource required by a load balancing task is greater than or equal to the average CPU resource, it means that the task is a CPU-intensive type. When the CPU resources required for the load balancing task are less than the average CPU resources and the load balancing task memory resources are greater than or equal to the average memory resources, the task is memory intensive. When the load balancing task memory is less than the average memory and the bandwidth resource is greater than the average bandwidth resource, the task is network bandwidth intensive.

The current load of the cloud computing platform node is denoted by L_j, which is calculated as in equation (7):

$$L_j = w_1 r_1 + w_2 r_2 + w_3 r_3 + w_4 r_4 + w_5 r_5 \tag{7}$$

where: r_1 and r_2 are the available CPU and bandwidth resources, r_3 and r_4 are the available memory resources and the task queue, r_5 is the response time of executing the task, and w_i is the corresponding weight of each task.

In order to ensure that the quality of service of network load balancing tasks performed by the cloud computing platform meets user requirements, the load balancing tasks should be completed within the specified time. When creating the virtual machine of the cloud computing platform corresponding to the load balancing task, the CPU resources of the cloud computing node are considered first, and then the memory and bandwidth resources of the cloud computing node are considered.

The VM resources set for network load balancing task i need to satisfy the constraints as in equation (8):

$$\begin{cases} s_l / (P - t) \le v_1 \\ s_f \le v_2 \\ s_n \le v_3 \end{cases} \tag{8}$$

where: t is the waiting time of the task in the network load balancing task queue. v_1 and v_2 are the CPU resources and memory resources of the virtual machines in the cloud computing, respectively. v_3 is the bandwidth resource of the virtual machine in the cloud computing platform. The type of the virtual machine created by the cloud computing platform needs to be the same as the type of the load balancing task.

III. B. Cloud Network Migration Virtual Machine Selection

The upper and lower thresholds of the servers within the cloud computing network are $\delta_{\max}$, $\delta_{\min}$. VM migration is triggered when the predicted server load value appears to be greater than $\delta_{\max}$ or less than $\delta_{\min}$ for 2 consecutive cycles. Considering the shortest VM migration time and the minimum migration amount together, the migration target VM set selection model is established, expressed as equation (9):

$$R = \begin{cases} S \in V & \hat{x}_{j'} - \sum u_a(v) < \delta_{\max} \\ S \to \min, T_s \to \min & \hat{x}_{j'} > \delta_{\max} \\ V & \hat{x}_{j'} < \delta_{\min} \end{cases} \tag{9}$$

where: $\hat{x}$ is the predicted value of the j' th server load within the cloud computing network. v is the amount of resources occupied by virtual machine v. V is the set of virtual machines within the cloud computing network. S is a subset of VMs to be migrated within V. T_s is the VM migration time, and T_v is the migration time of v in equation (10):

$$T_v = \frac{Z_v}{N_v} \tag{10}$$

where: Z_v is the memory occupied by v. N_v is the available bandwidth of v, and T_s is the sum of all T_v.

The set of target VMs R to be migrated can be determined by solving Eq. (9) using genetic algorithm as follows:

Step 1 Generate an initial population M while solving for the fitness of each chromosome, if the fitness f of a chromosome is the optimal solution, then return the chromosome for which the optimal solution exists and continue to Step 6.

Step 2 Select any 2 chromosomes within M and perform crossover.

Step 3 Solve for the f value of the 2 chromosomes after the crossover operation, if 1 of the chromosomes is already the optimal solution, then return to the chromosome where the optimal solution exists, and continue to step 6. Conversely, select the 2 chromosomes with the maximal f corresponding to the chromosomes before and after the crossover, and perform a mutation operation on them.

Step 4 Solve for the f value of the mutated chromosomes, and if 1 of the chromosomes is already the optimal solution, then return to the chromosomes before and after the mutation crossover and select the 2 chromosomes corresponding to the largest f within the chromosomes and store them into the new population.

Step 5 Repeat steps 2-step 4 until the chromosome corresponding to the largest f is obtained.

Step 6 Output the solution result of Eq. (10), i.e., the set of target VMs R to be migrated.

III. C. Target Server Selection for Cloud Computing Networks

After determining the target set of VMs R to be migrated, it is necessary to select the target server. Using the maximum remaining capacity as the selection strategy, the target server is selected and the conditions it needs to fulfill are:

(1) The target server can provide sufficient resources for R.

(2) Avoid secondary migration as much as possible to reduce the number of VM migrations.

Based on the above selection strategy, the target server can be selected to minimize the number of VM migrations with energy effect and improve the resource utilization.

IV. Testing and Evaluation of Network Load Balancing Models

In order to validate the effectiveness of the modeling algorithms proposed above, this section launches experiments on runtime performance, throughput rate and average waiting time, and convergence performance.

IV. A. Operational performance

In order to verify the performance of the algorithm, this section utilizes open-source cloud computing simulation software to simulate a distributed cloud computing network environment, and conducts data sampling on the software's own PlanetLab platform to test and verify the overall performance of the algorithm in this paper and compare it with a representative algorithm for load balancing in cloud computing networks (K). First, no less than 25 heterogeneous physical host nodes (CPU processing power (MIPS) {1000,1800,2600,3000}, memory (G) {1,2,4,8}, bandwidth (Mb/s) {500,700,1000}) are selected for the experiments, and each physical node is configured with 3-5 virtual computing nodes respectively (CPU processing power (MIPS) {200, 500,1000,1500,2500}, memory (G) {0.5,1,2,3}, bandwidth (Mb/s) {100,200,500}), simulate to build a distributed cloud computing network environment and simulate the operation of the cloud computing system driven by different types of cloud tasks and different sizes of service data, and utilize the algorithm in the paper to perform the system DLB operation. Statistics of the average response time of the system, analysis of virtual machine migration and node utilization, and algorithm comparison results are shown in Fig. 1 and Fig. 2.

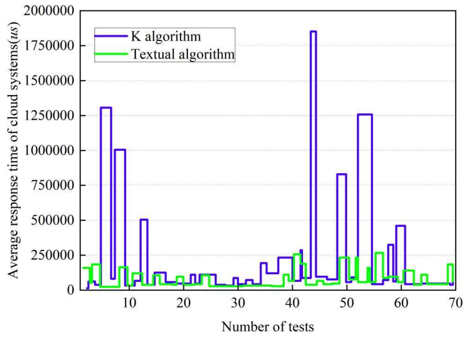

Figure 1: Average system response time under different cloud task conditions

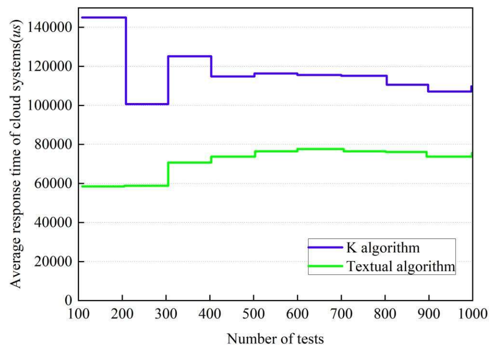

Figure 2: The average system response time under different test times

As can be seen from the figure, on the one hand, this paper's algorithm has better adaptability to different cloud computing task types, the average response time of the system has been significantly reduced, the algorithm is running in a stable state, and the system has shown good load balancing performance in the initial stage of testing. On the other hand, as the number of tests increases, the feedback adjustment optimization effectiveness of this paper's algorithm is more obvious, compared with the K algorithm, the average response time of the system is not only optimized by more than 30%, but also tends to be smoother, and the overall performance of the cloud computing load balancing system has been greatly improved.

IV. B. Throughput rate and average waiting time

The experimental results of throughput rate of the designed dynamic load balancing algorithm are shown in Fig. 3, and the experimental results of the average customer waiting time are shown in Fig. 4. From the figure, we learn that the throughput rate of this paper's algorithm fluctuates in the range of 1000~3500, with the mean value of about 2644 and the mean variance of about 369. The fluctuation of the average customer waiting time of this paper's algorithm fluctuates in the range of 271~701s, with the mean value of about 387 and the mean variance of 60.

Combining data and images: the throughput rate and average customer waiting time fluctuation range up and down is small. The overall mean square deviation of the performance of the three web servers is around 369, with only one pulse fluctuation, which indicates that the performance of the three servers did not differ too much throughout the experiment, so the results received by the clients show the same float, and the method maintains a mean square deviation of around 369 with good stability, without the occurrence of a certain server being overloaded at a certain point of time, while the other one server is idle. The mean squared deviation of both the throughput rate and the average client waiting time is slightly lower than that of the static load balancing algorithm, indicating that the minimum response time algorithm has higher stability.

Overall, the method proposed in this paper has better stability and can distribute load to back-end servers better, but its accuracy is affected by the back-end performance index, so it is necessary to monitor the back-end index in real time.

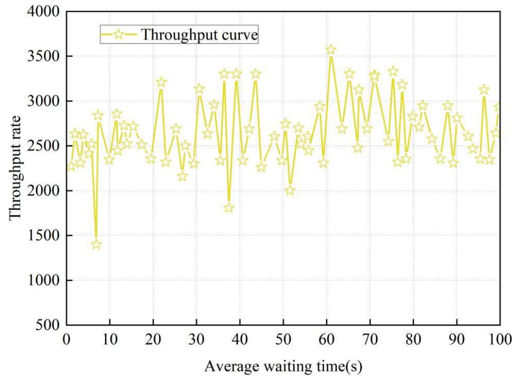

Figure 3: Throughput change

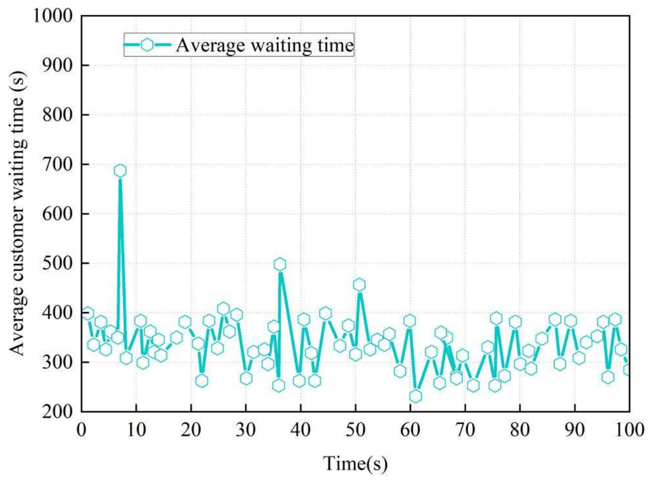

Figure 4: Average customer waiting time changes

IV. C. Convergence performance

This section evaluates the convergence performance of this paper's model algorithm and the DTORS algorithm. Figure 5 represents the variation of load migration with the number of samples V for this paper's model algorithm for different iteration stopping criterion ϵ. When ϵ is larger, the load migration can converge with fewer number of samples, but converges to a higher load migration. If a smaller ϵ is used, the algorithm needs to sample more random variables from the multivariate Gaussian distribution to recover the computational migration decision. For example, when $\epsilon = 10^{-1}$, load migration can converge to a stable value after 108 samples. When $\epsilon = 10^{-2}$, at least 140 samples are needed to converge to a stable load migration. However, too small an ϵ increases the number of times the algorithm samples and the iteration time for each sample, so the ϵ of the modeling algorithm in this paper is set to 10^{-2}.

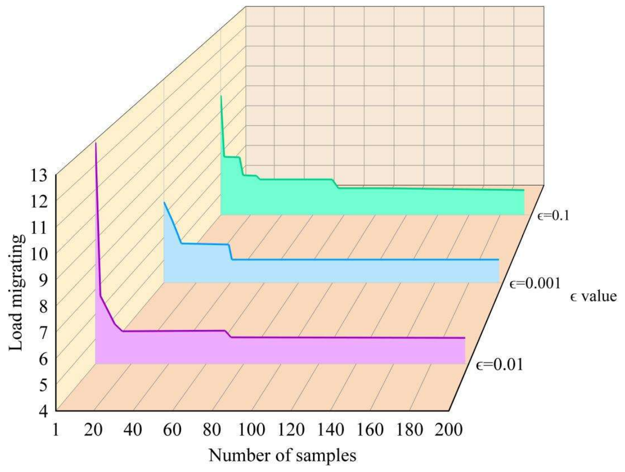

Figure 5: The load migrating v.s.stopping criterion ϵ in textual algorithm

Figures 6 and 7 simulate the load migration convergence of the DTORS algorithm for different learning rates ϵ and batch training sizes M, respectively. It can be seen that the DTORS algorithm can converge to a stable value with different learning rates and batch training sizes. However, a learning rate that is too large or too small leads to higher volatility and convergence to higher load migration. Therefore, the learning rate is set to 10^{-5}. Fig. 7 shows the convergence performance at three batch training sizes. When the batch training size is 128, the DTORS algorithm converges to higher load migration. When the batch training size is 512, the DTORS algorithm converges at a lower rate than when M = 256. Therefore, the batch training size of the DRORS algorithm is set to 256.

In summary, it can be seen that the DRORS algorithm learning rate (10^{-5}) is lower than the model algorithm (10^{-2}) in this paper.

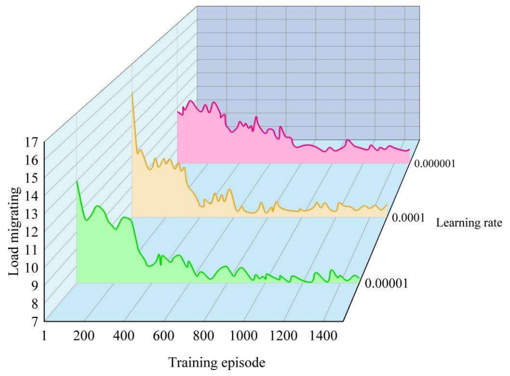

Figure 6: The load migrating v.s. stopping criterion ϵ in DTORS algorithm

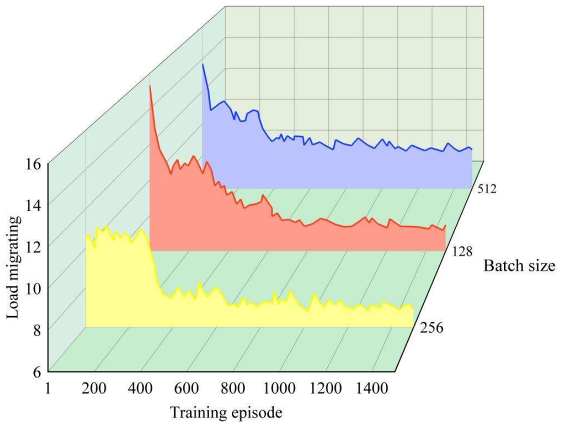

Figure 7: The load migrating v.s.stopping criterion ϵ in DTORS algorithm

V. Conclusion

Aiming at the problem of cloud computing network load imbalance, this paper proposes a cloud computing network load balancing model. The model optimizes network load balancing by real-time mining and exploring and identifying transmission data to reduce the probability of communication channel node queue stacking problems during transmission. It also utilizes cloud computing technology to migrate virtual machines online to achieve dynamic load balancing and control.

The designed model algorithm optimizes the average response time of the system by more than 30% and tends to be smooth, the throughput rate fluctuates in the range of 1000~3500 with an average variance of about 369, and the average waiting time of the customers fluctuates in the range of 271~701s with an average variance of 60. Compared with similar algorithms, it not only converges under different coefficient parameters but also achieves the near-optimal load migration Cost. By optimizing the dynamic balancing control of cloud computing network load, the effectiveness and real-time information in the cloud computing communication network are guaranteed.

References

[1] Zhang, Y., & Zhou, Y. (2018). Distributed coordination control of traffic network flow using adaptive genetic algorithm based on cloud computing. Journal of Network and Computer Applications, 119, 110-120.

[2] Tamura, Y., & Yamada, S. (2016). Reliability computing and management considering the network traffic for a cloud computing. Annals of Operations Research, 244, 163-176.

[3] Zheng, G., Zhang, H., Li, Y., & Xi, L. (2020). 5G network-oriented hierarchical distributed cloud computing system resource optimization scheduling and allocation. Computer Communications, 164, 88-99.

[4] Chen, S. L., Chen, Y. Y., & Kuo, S. H. (2017). CLB: A novel load balancing architecture and algorithm for cloud services. Computers & Electrical Engineering, 58, 154-160.

[5] Mishra, S. K., Sahoo, B., & Parida, P. P. (2020). Load balancing in cloud computing: a big picture. Journal of King Saud University-Computer and Information Sciences, 32(2), 149-158.

[6] Kumar, C., Marston, S., Sen, R., & Narisetty, A. (2022). Greening the cloud: a load balancing mechanism to optimize cloud computing networks. Journal of Management Information Systems, 39(2), 513-541.

[7] Shafiq, D. A., Jhanjhi, N. Z., Abdullah, A., & Alzain, M. A. (2021). A load balancing algorithm for the data centres to optimize cloud computing applications. Ieee Access, 9, 41731-41744.

[8] Komathi, A., Kishore, S. R., Velmurugan, A. K., Begum, A. S., & Muthukumaran, D. (2024). Network load balancing and data categorization in cloud computing. Indonesian Journal of Electrical Engineering and Computer ScienceThis link is disabled, 35(3), 1942-1951.
[9] Hasan, R. A., & Mohammed, M. N. (2017). A krill herd behaviour inspired load balancing of tasks in cloud computing. Studies in Informatics and Control, 26(4), 413-424.
[10] Gao, R., & Wu, J. (2015). Dynamic load balancing strategy for cloud computing with ant colony optimization. Future Internet, 7(4), 465-483.
[11] Alghamdi, M. I. (2022). Optimization of load balancing and task scheduling in cloud computing environments using artificial neural networks-based binary particle swarm optimization (BPSO). Sustainability, 14(19), 11982.
[12] Gamal, M., Rizk, R., Mahdi, H., & Elnaghi, B. E. (2019). Osmotic bio-inspired load balancing algorithm in cloud computing. Ieee Access, 7, 42735-42744.
[13] Khan, A. R. (2024). Dynamic load balancing in cloud computing: optimized RL-based clustering with multi-objective optimized task scheduling. Processes, 12(3), 519.
[14] Moura, J., & Hutchison, D. (2016). Review and analysis of networking challenges in cloud computing. Journal of Network and Computer Applications, 60, 113-129.
[15] Kang, L., & Ting, X. (2015, July). Application of adaptive load balancing algorithm based on minimum traffic in cloud computing architecture. In 2015 International Conference on Logistics, Informatics and Service Sciences (LISS) (pp. 1-5). IEEE.
[16] Chen, J., Chen, J., & Guo, K. (2023). Queue-aware service orchestration and adaptive parallel traffic scheduling optimization in SDNFV-Enabled cloud computing. IEEE Transactions on Cloud Computing, 11(4), 3525-3540.