DLINE JOURNALS

Journal of Intelligent Computing

Print ISSN: 0976-9005
Online ISSN: 0976-9013

JIC 2024;15 (4)
https://doi.org/10.6025/jic/2024/15/4/135-141

Research on Machine Learning-Based Microservices Load Balancing Algorithm

Kai Wang, Na Wang
Xinxiang Vocational and Technical College
Xinxiang, Henan,453006, China
btparrot@qq.com

ABSTRACT

The rapid progress of cloud computing technology has attracted widespread attention, and the importance of microservices architecture is also increasing. In this regard, we studied a new algorithm, Xgboost, which effectively addresses the load balancing issues in the current microservices cluster, thus better meeting user demands. We identified factors that significantly impact load balancing effectiveness through in-depth research on various features. We used ensemble learning to estimate the load received by each server node, thereby achieving load balancing. Experimental results confirm that our proposed algorithm significantly improves throughput, reduces interception errors, and lowers the system's average response time compared to other load-balancing algorithms.

Received: 27 June 2024
Revised: 3 September 2024
Accepted: 11 September 2024

Keywords: Microservices,Load Balancing,Ensemble Learning,Response Time Prediction, Xgboost

1. Introduction

In recent years, with the rapid development of cutting-edge information technologies such as big data and cloud computing, there has been an unprecedented upgrade in various fields such as science, engineering, trade, finance, etc., setting higher standards for the evolution of traditional software and hardware architectures [1]. With the advancement of technology, microservices architecture has become an important application approach that decomposes complex applications into a set of independent microservices that coordinate with each other to perform more functions, greatly improving the efficiency and reliability of the system [2]. By adopting the microservices architecture, various microservices can be installed across multiple servers to form a complex service structure known as a cluster [3]. Over time, the server load in the cluster may change, so to effectively handle client tasks; we must adopt efficient load-balancing algorithms to ensure effective control of task response time. Using this strategy, the load distribution among cluster servers becomes more balanced, and the overall efficiency of the cluster system is significantly improved. Although Dubbo and Spring Cloud can provide a series of simple load balancing algorithms, such as random, round-robin,

and shortest response time, they fail to fully utilize server scalability and address potential server resource scheduling issues [4]. Researchers have attempted to develop dynamic load-balancing algorithms to improve their deficiencies, especially when dealing with complex network environments. However, due to their limited capacity to handle heavy loads and inability to meet complex network environments, developing a method to achieve dynamic load balancing has become increasingly important [5].

2. Related Work

Currently, two major types of dynamic load balancing techniques have been developed: static load balancing algorithms, which are based on the same basic principles to achieve load balancing. Static load balancing algorithms can be implemented in various ways, including random, round, and weighted round-robin algorithms. Among them, the random algorithm allows the system to automatically select the optimal load from multiple servers [4], aiming for the minimum, maximum, or optimal load. Although the basic concept of static load balancing algorithms is relatively easy to grasp, they overlook the actual server load conditions, making it difficult to guarantee load balancing within the cluster when load variations exceed expected ranges [5]. The cluster load can be better assessed using dynamic load balancing algorithms, and more precise optimization can be provided to individual servers. For example, using the shortest response time algorithm, task distribution can be optimized upon receiving a detection request, thereby improving system performance. Although the shortest response time algorithm indirectly reflects real-time server performance, it also comes with significant network transmission costs, especially when dealing with large-scale multi-threading, making this issue more prominent [6]. The dynamic load balancing algorithm proposed by Li and Lina uses CPU, disk, and memory utilization as metrics for measuring server load. Observing the changes in these metrics within two seconds allows weight values to effectively allocate tasks to servers with the lowest load [7]. Liao proposed a load-balancing strategy inspired by the minimum connection algorithm, incorporating load evaluation parameters. Still, they failed to fully consider the impact of different task types on response time [8]. This paper introduces queue waiting time as an important parameter for evaluating server load. It constructs a predictive model based on task types to measure the balancing effect of CPU-intensive loads. However, the model can only reflect the influence of task types and cannot accurately reflect actual situations. Liao proposed a new approach that uses machine learning techniques to predict the response time of microservices in a multi-cloud environment, assuming a time correlation between response time and other factors [9]. However, this approach does not elaborate on how to select appropriate performance indicators. Although we selected some feature parameters, they did not consider server CPU utilization and other load metrics, thus being unable to evaluate the performance of server nodes fully. To address this challenge, we propose a machine learning-based dynamic microservices load-balancing algorithm. This algorithm simulates the consumption of CPU, memory, tape, and other resources of client devices and combines regression models to estimate the load balancing of client devices accurately. The load can be distributed to micro-servers to operate client devices [10] efficiently.

3. Machine Learning-Based Microservices Load Balancing Algorithm

3.1. System Framework Based on Microservices Cluster

With the advancement of technology, an increasing number of open microservices architectures are emerging, including Dubbo, Spring Cloud, and some other more advanced architectures. Dubbo, as the core of Alibaba's service-oriented management, enables scheduling, detection, supervision, and management and implements large-scale service management. On the other hand, Spring Cloud consists of various scalable sub-architectures that allow enterprises to build more flexible distributed services that better meet their needs. Building upon the previous research on these two open-source microservices cluster frameworks, we propose a new load-balancing algorithm that effectively addresses service discovery and load-balancing issues. The details can be referred to in Figure 1.

The microservices cluster system framework provided in this paper consists of a client, task processing module, service registry center, and microservices cluster. Each plays a different role and can achieve efficient service invocation, thus ensuring the system's efficient operation. The task processing module comprises four components: the monitor, network connector, load balancing scheduler, and service request receiver.

(1) The monitor module can handle two different types of tasks simultaneously: one receives tasks from clients and then passes them to servers, and the other collects feedback from client tasks on micro-servers and analyzes it comprehensively for better understanding and handling by clients.

(2) The service gateway is an indispensable component of the microservices cluster system, responsible for unified access, providing secure protection, and ensuring task security through filtering and verification. It also transfers tasks to the load-balancing scheduler for more efficient data transmission.

(3) The load balancing scheduler is responsible for coordinating and optimizing the load of the microservices cluster. It can quickly identify tasks to be processed and their corresponding optimization plans based on user demands. It helps users adjust the load quickly to achieve better efficiency.

(4) The service request receiver retrieves task execution results from microservice instances, integrates them effectively, and transmits them to the service monitor module.

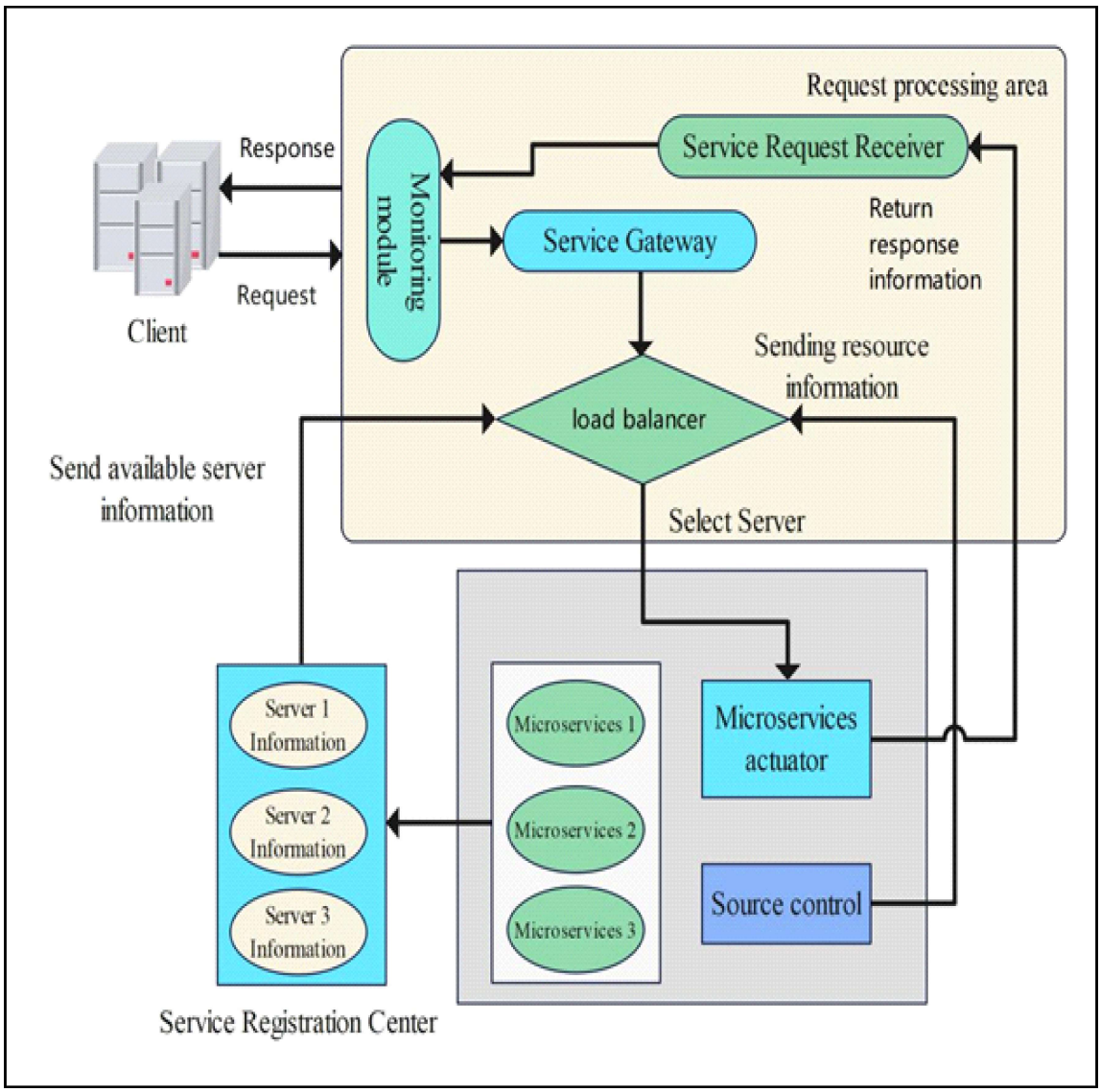

Figure 1. System framework based on microservices cluster

3.2. Load Balancing Scheduler Algorithm Based on Machine Learning Technology

This paper explores a load-balancing algorithm based on machine learning, which includes two main steps: offline simulation and online simulation. We will analyze these two steps in detail. The offline phase aims to build effective predictive models by collecting load status information and response times for various task types. Since this data is offline, it will not be considered in the actual server selection process. The online phase is the core part of the algorithm, where servers are selected based on pre-trained models. When tasks arrive at the load-balancing scheduler, the service registry center analyzes which servers can provide services and sends this information to the load-balancing scheduler. Then, the load balancing scheduler collects server resource usage and uses the pre-trained machine learning model to predict the response time of each server for that task. Finally, by comparing the predicted results, the server's performance is determined. We can pass the task and relevant information to the microservice executor by selecting the server with the fastest response time.

3.3. Load Balancing Algorithm

The core of the load balancing algorithm lies in carefully selecting suitable models, including factors such as random forest, gradient boosting, extreme gradient boosting, etc. In our actual model, we adopt the *xgboost* model and, through experimental verification, demonstrate its effectiveness in simulating load balancing, thus achieving better model performance. The ensemble learning model constructed by the Boosting algorithm can extract information from the stored training set of the first learner, enabling the use of this information to predict future trends and estimate their accuracy by comparing them, ultimately forming a complete learning system to meet the evolving environmental demands. Therefore, the base learners of the Boosting algorithm influence each other. Through continuous iterations, they can approximate the real data more closely and integrate it into other learning algorithms, thereby forming a more comprehensive learning model. The Boosting algorithm can effectively improve the performance of the model by first training the first base learner *f*(*c*) using the dataset {(*X; ,y*)}, then calculating the residual between the true value y and the prediction of the first base learner *f*(*xo*), i.e., *c=y-f(xo).* Next, it trains a new base learner to fit the residuals better and repeats this process to boost the model. By constructing a new sample set {(*x;, c*;)} and training a base learner, after m iterations, the final prediction result will be the sum of m base learners, i.e*., f..(.x) = f m.-.(c).T(x,0m*).

The GBDT algorithm is an optimized algorithm that estimates the residuals of the current model by computing the negative gradient of the loss function, thus constructing a more accurate learner to identify better and predict complex situations. Its construction process includes:

In the training set Data, initialize the base learner to understand the relationship between x and y better.

$$f_0(x)=\arg\min_{h_0}\sum_{i=1}^{N}L\left(y_i,h_0(x)\right) \tag{1}$$

In this model, *i* ranges from 1 to N, and the quantity of each sample is represented by *L(·).* The loss function is represented by h0(z), and the first training obtains n base learners composed of n samples. If the number of iterations exceeds m, then during the *m*-th iteration, the negative gradient of the *i*-th data will change:

$$\tilde{y}_{im}=-\frac{\partial L\left(y_{im-1},f_{m-1}\left(x_i\right)\right)}{\partial f_{m-1}\left(x_i\right)} \tag{2}$$

The Xgboost method and GBDT algorithm are both known as Boosting methods and have their advantages. However, the Xgboost method is more powerful as it not only regularizes but also takes more measures to control the complexity of base learners' decision trees, effectively avoiding overfitting. By combining strong learners, the Xgboost method significantly improves its performance.Compared to the GBDT algorithm, its key feature is the combination of strong learners and higher efficiency and accuracy.

In the Xgboost method, J represents the number of leaf nodes, and b represents the value of

each leaf node. This regularization approach helps reduce the complexity of the trees and effectively reduces the number of leaf nodes, thereby preventing the model from overfitting.

4. Experimental Results and Analysis

4.1. Data Collection

We collected data by monitoring each node in the Alibaba microservices production cluster in 2021 using the cluster-trace-microservices-v2021 dataset [15-16]. This dataset provides valuable information on each task's response time, CPU, memory, and disk usage, enabling us to evaluate their performance better. By training three different ensemble learning regression models (Random Forest, GBDT, Xgboost), we obtained more accurate results, and the Root Mean Square Error (RMSE) served as an important metric to measure the accuracy of these models. The calculation formula for RMSE is as follows:

$$\mathbf{RMSE} = \sqrt{\frac{1}{m}\sum_{i=1}^{m}\left(y_i - \bar{y}_m\right)^2} \tag{3}$$

Here, *m* represents the number of samples, *yi* represents the actual values, and *wi* represents the predicted results.

4.2. Model Comparison

In this section, we conducted eight experiments to comprehensively evaluate the models' effectiveness, accuracy, reliability, and operability. We used RMSE as the metric to compare the three models' training, prediction, and reliability. By training the established models, we obtained the accuracy of RMSE. For future RMSE, we conducted experiments using the established models to obtain accurate results. Through the training of the three different ensemble learning regression models, we obtained faster training RMSE. Figure 2 displays the predicted results for the future.

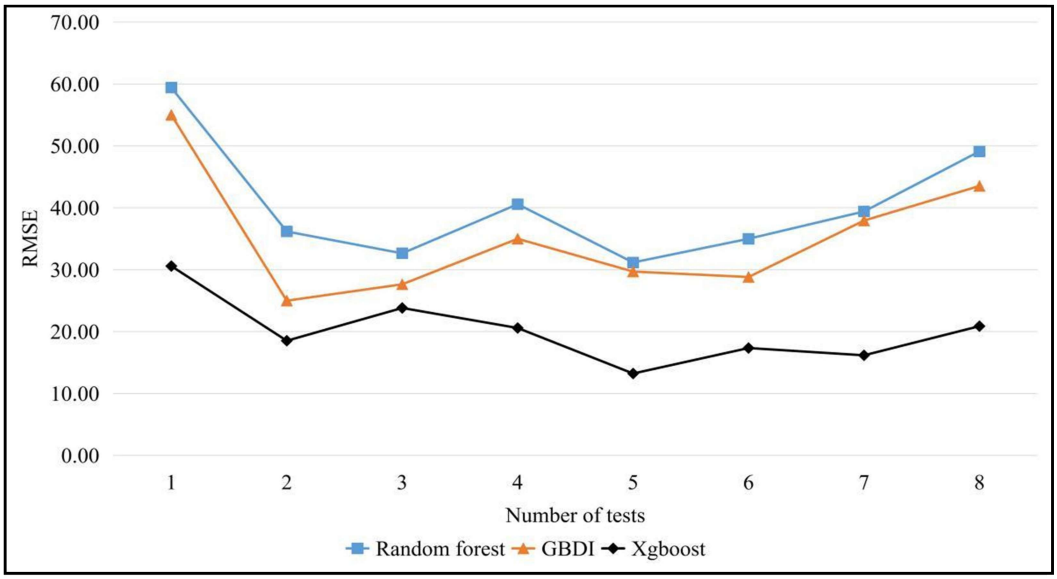

Figure 2 . Model Evaluation

After model testing, we found that although the training time of the Xgboost algorithm is longer than that of the Random Forest algorithm and GBDT algorithm, its accuracy, based on training RMSE data, improved by 79.2% compared to the former and 82.1% compared to the latter. The Xgboost model effectively reduces the residuals of the GBDT algorithm by introducing regularization terms and feature sampling from the Random Forest, thus avoiding the overfitting problem.

Through testing, we observed that the prediction time of the Xgboost algorithm is 16.5% shorter than the Random Forest algorithm but longer than the GBDT algorithm. However, in terms of accuracy, the Xgboost algorithm outperforms the Random Forest and GBDT algorithms, with an accuracy improvement of 49.3% and 41.7%, respectively.

After comparing various machine learning algorithms, we found that the Xgboost model exhibits extremely high predictive accuracy, making it the final choice for our machine learning algorithm.

4.3. Model Training Results

After in-depth research, we used the Xgboost algorithm for machine learning. We tracked a large amount of data using Alibaba's WeChat official account and randomly arranged this data, dividing it into two training sets and test sets in a ratio of 9:1. After careful optimization of the Xgboost model parameters and multiple experiments, we found that when these parameters were appropriately configured, the predictive performance was better. After evaluating the test set, the fitted effect of the trained prediction model was significantly better than that shown in Figure 3, with a notable improvement in prediction accuracy, even in minor details. Therefore, integrating the model into the load-balancing scheduler can achieve more accurate predictive results.

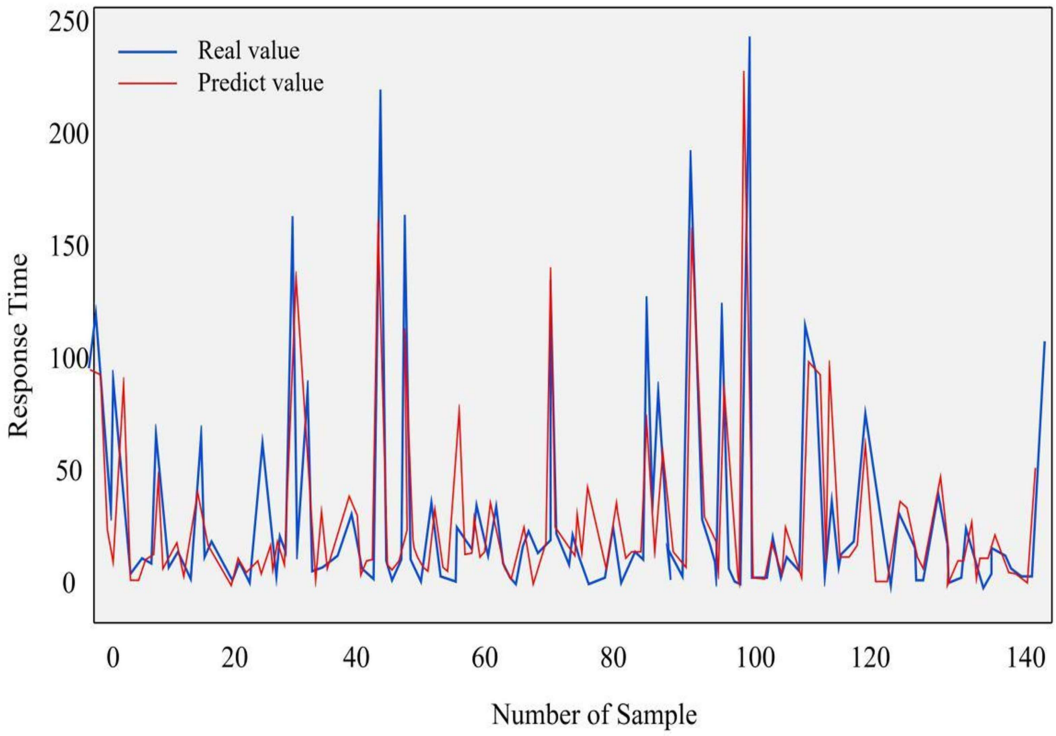

Figure 3. Model Predictive Fit

The performance parameters of SPRT were considered, and when the concurrent task quantity was between 300 and 450, the SQLB and Shortest Response Time (SRT) algorithms showed similar average response times as the SPRT algorithm. However, if the concurrent task quantity continued to increase, the server node load would also increase, and the SRT algorithm network communication cost would increase accordingly. As SQLB does not consider various performance indicators of server nodes, when the concurrent task quantity exceeds 500, the mean response time of all algorithms increases rapidly. Overall, compared to the LSTM model for load balancing prediction, the SQLB and SRT algorithms showed a 21.8%, 14.2%, and 18.6% reduction in mean response time, respectively, and it can be observed that the SPRT algorithm had a more significant improvement effect under high concurrency rates.

5. Conclusions

In response to the problem of poor load balancing in server nodes of microservice clusters, this paper comprehensively uses machine learning models to propose a load balancing algorithm based on the minimal predicted response time. The theoretical analysis considers the impact of job type, CPU utilization, storage utilization, and disk utilization on job response time. Regression models are built between target values and feature parameters, and these trained models are integrated into the load-balancing scheduler. When new job tasks arrive, the model is used to estimate the response time of the job on different server nodes, and the job is finally assigned to the server node with the minimum estimated response time, aiming to assign jobs to better-performing server nodes each time. The study shows that for high-parallel environments, the load balancing algorithm in this paper performs better in throughput, cut-off rate, and average response time compared to the LSTM time series model load balancing algorithm, Shortest Response Time algorithm, and SQLB.

References

[1] Benhadid, A., & Merahi, F. (2023). Complexity analysis of an interior-point algorithm for linear optimization based on a new parametric kernel function with a double barrier term. *Numerical Algebra, Control and Optimization, 13*(2), 224–238.

[2] Nguyen, T. H., Nguyen, X. T., Pham, D. A., et al. (2023). A new approach for mobile robot path planning based on RRT algorithm. *Modern Physics Letters B, 37*(18).

[3] Falahati, A., & Shafiee, E. (2022). Improve safety and security of intelligent railway transportation system based on balise using machine learning algorithm and fuzzy system. *International Journal of Intelligent Transportation Systems Research, 2022*, 1–15.

[4] Wang, H., Wang, Y., Liang, G., et al. (2021). Research on load balancing technology for microservice architecture. *MATEC Web of Conferences, 336*, 08002.

[5] Liang, H., Liu, G., Zou, J., et al. (2021). Research on calculation model of bottom of the well pressure based on machine learning. *Future Generation Computer Systems, 124*, 80–90.

[6] Chi, X., Yang, Q., Wan, Z., et al. (2023). The new full-Newton step interior-point algorithm for the Fisher market equilibrium problems based on a kernel function. *Journal of Industrial and Management Optimization, 19*(9), 7018–7035.

[7] Hufnagl, B., Stibi, M., Martirosyan, H., et al. (2021). Computer-assisted analysis of microplastics in environmental samples based on ìFTIR imaging in combination with machine learning. *Environmental Science & Technology Letters, 9*(1), 90–95.

[8] Labiadh, M., Obrecht, C., Ferreira da Silva, C., et al. (2021). A microservice-based framework for exploring data selection in cross-building knowledge transfer. *Service Oriented Computing and Applications, 15*, 97–107.

[9] Liang, H., Liu, G., Zou, J., et al. (2021). Research on calculation model of bottom of the well pressure based on machine learning. *Future Generation Computer Systems, 124*(6).

[10] Jiang, H. (2021). Research on robot path planning algorithm based on biological heuristic machine learning algorithm. *IOP Conference Series: Earth and Environmental Science, 769*(3), 032066.