European Journal of Business, Economics & Management

Vol. 1 No. 3 2025

Article **Open Access**

Research on Cloud Computing Resource Scheduling Strategy Based on Big Data and Machine Learning

Jiaying Huang [1,*]

[1] EC2 Core Platform, Amazon.com Services LLC, Seattle, WA, 98121, United States
* Correspondence: Jiaying Huang, EC2 Core Platform, Amazon.com Services LLC, Seattle, WA, 98121, United States

Received: 14 August 2025
Revised: 27 August 2025
Accepted: 10 September 2025
Published: 13 September 2025

Abstract: Optimizing the efficient scheduling of cloud platforms is crucial for enhancing the performance and resource utilization of these platforms. The intelligent scheduling optimization framework proposed in this article combines big data and deep learning technologies, which can effectively solve a series of severe challenges in cloud platform resource scheduling, such as redundant resource preparation, inaccurate capacity estimation, and high scheduling costs. Suggest building a multidimensional feature model that combines historical and real-time monitoring information, and using LSTM for accurate resource prediction. Based on this result, a reinforcement learning based automatic adjustment hybrid scheduling algorithm was designed to achieve intelligent dynamic scheduling of different resources. Meanwhile, a graph partitioning mechanism is adopted to reduce scheduling complexity and improve system scalability. On the Google Cluster Trace dataset, the proposed solution can significantly improve resource utilization, increasing it from raw utilization to 21.5%, reducing the average waiting time of tasks by 18.3%, and lowering SLA default rates by 13%. The deployment experiment in the Kubernetes environment further validated the feasibility and effectiveness of the proposed solution. The research results provide evidence and understanding for the application of learning based intelligent scheduling technology in cloud infrastructure.

Keywords: cloud computing; resource scheduling; machine learning

1. Introduction

Resource scheduling is the core of cloud service platforms and has a significant impact on the efficiency, service quality, and utilization of cloud service systems. Traditional scheduling strategies are unable to meet the constantly changing task loads, diverse resource types, and complex business environments, resulting in unnecessary resource waste, service delays, and other issues. In recent years, big data and machine learning technologies have provided new ideas for establishing predictive models based on historical operations, real-time monitoring data, etc., and further improving scheduling strategies through deep learning and reinforcement learning to achieve dynamic management of multidimensional resources such as computing and storage. This article attempts to construct an intelligent scheduling framework based on big data and machine learning models, proposes an efficient cloud service platform resource scheduling strategy, and verifies its significant effects in improving utilization, shortening waiting time, and ensuring service quality through implementation, providing a theoretical basis and practical means for cloud computing intelligent scheduling.

2. Overview of Cloud Computing Resource Scheduling

Cloud computing resource scheduling refers to the dynamic configuration of computing, storage, and network resources on a cloud platform based on resource requirements and system conditions for resource allocation. The scheduling objects include virtual machines, containers, queues, etc., which consist of resource identification, task matching, priority judgment, execution order decision-making, and other processes. Due to the dynamic and diverse nature of resource scheduling in cloud computing environments, the scheduler should make scheduling decisions in a short period of time in order to maintain system operation [1]. The implementation of scheduling strategies relies on the platform's resource abstraction mechanism, monitoring methods, and scheduling algorithms, aiming to select the optimal strategy among many feasible scheduling options based on real-time system status and task characteristics. Resource scheduling can be roughly divided into centralized and distributed scheduling. Centralized scheduling is fully managed by a central decision controller, while distributed scheduling delegates the decision-making power of resource scheduling to various branch nodes to improve the concurrency of system scheduling. Scheduling algorithms can be divided into two forms in practical applications: static scheduling and dynamic scheduling. Static scheduling relies on the status information submitted by tasks to make decisions, while dynamic scheduling optimizes resource scheduling strategies through real-time status.

3. Resource Scheduling Issues in Cloud Computing Platforms

3.1. Mismatch between Reserved Resources and Actual Needs

In actual cloud computing platforms, when executing scheduling tasks, the scheduler generally requires the size of CPU and memory resources to allocate corresponding resources based on this. However, due to the lack of real-time monitoring of resource consumption, fixed allocation methods are mostly used for resource configuration. Therefore, in order to ensure task submission, the actual demand is often exaggerated, resulting in many unused reserved resources [2]. In this way, not only does it result in resource waste, but it also causes uneven distribution of resources. This phenomenon of "disconnection between resource request and usage" is quite common in containers represented by Kubernetes. Some tasks that provide long-term use of a small amount of resources can always be scheduled into the task queue with higher priority, blocking the entry of short tasks, which leads to unsatisfactory resource allocation results.

3.2. Inaccurate Load Forecasting

In the process of resource reservation and task sorting in scheduling systems, some traditional prediction methods often fail to obtain sufficiently accurate results. Prediction methods such as average value and nearest neighbor sampling have poor ability to identify abnormal peaks, period reversals, and long short period overlaps in load sequences. Due to the lack of time series characteristics in the prediction model itself, it is sensitive to input value jitter, which can cause prediction anomalies under high load conditions and affect scheduling decisions. If the prediction is too conservative, it will lead to scheduling queue congestion; If the prediction is too aggressive, it will waste a lot of resources and affect subsequent scheduling tasks. Although some platforms have also adopted window based calibration methods, the effect is not significant under conditions of multi tenancy and heterogeneous loads [3].

3.3. Unequal Allocation of Computing and Storage Resources

In the scenario of multi resource collaborative scheduling, the correlation between computing resources (CPU, memory) and storage space (disk) is strong, and most schedulers lack the ability to simultaneously consider and adjust the behavior of both. Traditional scheduling programs often prioritize the selection of tasks based on the reserved capacity of the host, without considering whether the input data required for this task

exists on the destination host, which can lead to a large amount of data migration and network congestion problems. Especially in batch processing tasks with special data requirements, if the scheduling results do not pay attention to the I/O bottleneck problem, it is possible to keep the job in a waiting state for a long time. In addition, due to the different forms of task loads, the scheduler is required to distinguish the mutual constraints and compatibility of different resources. Existing platform scheduling methods are often based on linear calculation of weight coefficients, lacking the ability to model resource structures and cannot implement multidimensional scheduling for different resources [4].

3.4. High Computational Complexity

Scheduling decisions in the system are manifested as high-dimensional constraint combination optimization. In the cloud platform where multiple tasks exist at the same time, complex node architecture, and policy dimensions are rich, the scheduler needs to analyze and judge a large number of states in a very short time. The task combination for each scheduling cycle may consist of hundreds or thousands of tasks and thousands of nodes, and the scheduler needs to perform multiple resource filtering, condition checking, and priority evaluation, resulting in heavy computational tasks. Compared to traditional single-objective scheduling, modern scheduling systems often have multi-objective synchronous optimization, such as maximum resource utilization, minimum SLA violation rate, fastest cold start time, etc., resulting in overly dispersed decision paths. In addition, in order to facilitate the modular design of scheduling, the scheduler itself will be divided into multiple steps to complete, and each step needs to ensure the consistency of the state and the interaction rules between modules, making the computing task more cumbersome.

4. Optimization Strategies for Resource Scheduling in Cloud Computing Platforms

To solve the problems of low resource utilization and slow scheduling response, this paper establishes an intelligent optimization framework that integrates big data, deep learning, and graph partitioning mechanisms. The optimization strategy flow of resource scheduling in cloud computing platforms is shown in Figure 1 below:

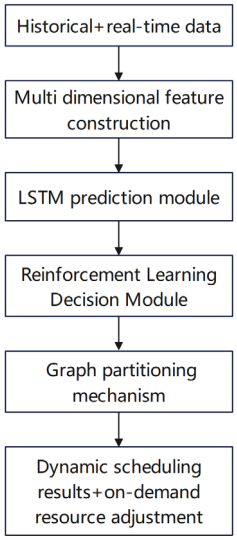

Figure 1. Flow Chart of Resource Scheduling Optimization Strategy for Cloud Computing Platform.

This process is built on historical and real-time data, constructing multidimensional feature inputs, and using LSTM for load trend prediction and analysis. The predicted values are used as the basic input for the reinforcement learning decision module. Then, a

graph partitioning mechanism is applied to achieve multi task concurrent scheduling, and an on-demand resource feedback mechanism is used to optimize resource allocation in real time, thereby improving the scheduling performance of the cloud computing platform.

4.1. Predicting Using Historical and Real-Time Data

In the cloud computing platform, the scheduler will arrange job orders, resource allocation, and priority reset according to the prediction results of computing resource load. To improve prediction accuracy and response efficiency, it is necessary to establish a multidimensional feature structure that integrates historical and real-time data and apply it as time-series data in the simulation process of subsequent models. The core is the collection and processing of data, using advanced data modeling to perform forward inference on the workload of the job [5].

The data sources can be divided into two categories: one is historical static databases, including information such as task execution time, average task time, and task types; The second is the real-time data collected by monitoring devices, such as node resources, waiting queue length, input/output rate, network traffic, and other information. The above data all have temporal and diverse characteristics, and require unified encoding and standardization processing to form a learnable structure.

After standardizing and normalizing the above data, a predictive input matrix can be constructed:

$$X_t = [x_t^{(1)}, x_t^{(2)}, \ldots, x_t^{(n)}] \in \mathbb{R}^n \quad (1)$$

Among them, $x_t^{(i)}$ represents the value of the i-th feature at time t, and n is the feature dimension. Within the scheduling period Δt, a time series input $\{X_{t-\Delta t}, \ldots, X_t\}$ is formed as the sliding window for the prediction model. This feature vector not only contains single point indicator values, but can also incorporate derived statistical measures (such as standard deviation, maximum value, offset ratio, etc.) to enhance the model's volatility perception ability.

4.2. Using Deep Learning for Complex Pattern Learning

The task load in cloud computing platforms usually exhibits severe randomness, high disturbance, and cross period dependencies, which are the strengths of time models in deep learning. The most common ones are RNN models with gate mechanisms (LSTM and GRU), which capture hidden state changes in the time dimension and perform dynamic state control.

For example, LSTM consists of an input gate, a forget gate, and an output gate, each with the task of saving current state information, forgetting historical states, and outputting transformation mapping content. These tasks can dynamically remember resource usage and state discrimination in the scheduling system. The update of model status can be represented as:

$$h_t = \tanh(W_h \cdot [x_t, h_{t-1}] + b) \quad (2)$$

Among them, $x_t \in \mathbb{R}^n$ represents the input vector at the current time, which includes multidimensional resource indicators such as CPU, memory, disk IO, etc; h_{t-1} is the hidden state of the previous time step; W_h and b are network weights and biases. By learning from the sequence of $\{x_{t-k}, \ldots, x_t\}$, the model can output a predicted value for x_{t+1}, which is the future resource state.

During the training phase, historical data is used for supervised learning of future conditions to reduce prediction errors; After training, it is used as a prediction service module, and the scheduler calls its results as important inputs for scheduling scores to assist in resource priority decision-making, task sorting, and node allocation. It greatly

enhances the sensitivity of scheduling and enables the scheduling system to better understand resource trends, providing sufficient data support for optimizing scheduling strategies in the next step.

4.3. Adopting an On-demand Resource Allocation Mechanism

The core of the on-demand resource allocation mechanism lies in flexibly configuring resources based on the real-time status of task execution, effectively solving the problem of resource waste caused by the traditional fixed allocation mode of "allocation locking". This mechanism relies on runtime resource monitoring and feedback mechanisms to continuously track resource consumption during task execution, adjust allocation limits in real-time, or initiate resource recycling and scaling operations.

From a technical implementation perspective, the scheduling framework needs to integrate resource usage awareness components, use container monitoring systems (such as Metrics Server) to obtain dynamic usage of CPU, storage space, and input/output devices, and configure policy triggering conditions. When the resource utilization rate of a task is detected to be lower than the preset value during a specific period of time, the system will automatically execute a resource reduction plan; When the resource usage continues to approach its peak, the resource expansion process is initiated. In the Kubernetes platform, such operations can be completed through horizontal or vertical auto scaling function modules. Resource allocation decisions are often constructed based on the following formula:

$$r_{\text{target}} = \alpha \cdot r_{\text{peak}} + (1 - \alpha) \cdot r_{\text{avg}} \tag{3}$$

Among them, r_{peak} represents the peak utilization rate within the observation window, r_{avg} represents the average value, and α is the weight parameter. This function balances extreme state and steady-state demands, generating a new resource target configuration r_{target} for dynamically updating the operational specifications of Pods or virtual machines.

This mechanism needs to rely on the resource estimation mechanism in the scheduler to implement, which means that after dynamic changes occur, the scheduling decision needs to be restarted to avoid scheduling failure or resource competition. This mechanism can more effectively release the resources of some nodes to high priority tasks, in order to improve the resource utilization level of the entire system.

4.4. Adopting a Divide and Conquer Approach to Optimize the Calculation Process

With the expansion of cloud computing platforms, schedulers have to deal with resource allocation issues between thousands of tasks and hundreds of nodes during each iteration. The scheduling process is a multi-objective and multi constraint combinatorial optimization problem, which can lead to exponential growth in its search space, directly causing computational bottlenecks. To improve the efficiency of scheduling response, the "divide and conquer" strategy is introduced, which divides scheduling into multiple independent sub problems to be solved simultaneously.

This scheme adopts a weighted undirected graph structure G (V, E) for scheduling scenario modeling, where the vertex set V corresponds to the task to be scheduled or available resources, and the edge set E represents the constraint relationship between tasks or between tasks and resources. The weighted values of each edge in the graph represent the strictness or execution cost of scheduling constraints. Using graph partitioning mechanisms (such as METIS) to decompose the original graph G into multiple subgraphs with lower coupling, enabling independent scheduling optimization within each subgraph and effectively reducing the computational burden of the global scheduler.

The objective function for graph partitioning is as follows:

$$\min \sum_{(u,v)\in C} w_{uv} \text{ s.t. } |V_i| \approx |V_j|, \forall i, j \tag{4}$$

Among them, $C \subset E$ is the set of edges spanning subgraphs, w_{uv} represents edge weights, and the constraint conditions need to ensure that each subgraph is of similar size to prevent load shifting. The partitioned data is used to establish a sub scheduler, which runs in a distributed or parallel structure to improve the overall scheduling throughput.

Each sub scheduler only needs to be responsible for scheduling the subgraph tasks assigned to it, and heuristic methods (such as minimum load priority) can be used to efficiently perform local optimal scheduling. The scheduling coordinator will fuse the divided scheduling results, and in case of resource conflicts or node overlaps, priority principles or re-matching methods can be chosen.

5. Case Study on Cloud Computing Resource Scheduling Based on Big Data and Machine Learning

In order to verify whether the scheduling optimization scheme proposed in this article is applicable to real-world scenarios, an experimental platform was built on the Google Cluster Trace database, which integrates LSTM based resource load prediction, reinforcement learning driven resource allocation, and a graph partitioning algorithm-based scheduling optimization mechanism to evaluate the performance of the entire system.

A set of experimental methods was designed as follows: training an LSTM model with multivariate time series data to obtain estimated CPU and memory usage for a certain period of time in the future; Combine the estimated results with the status of the task queue to form a scheduling state vector, which serves as input for the reinforcement learning model to generate corresponding resource allocation decisions; The task node pairing problem is represented graphically, and solved by decomposing it into relatively independent subgraphs through graph division to reduce the computational burden of centralized scheduling process (Table 1).

Table 1. Comparison of Key Performance Indicators of the System before and after Optimization Strategy Deployment.

Index	Unoptimized Strategy	Optimize Scheduling Strategy
Average Resource Utilization Rate	64.2%	85.7% (↑21.5%)
Average Waiting Time for Tasks (seconds)	3.82	3.12(↓18.3%)
SLA Default Rate	7.6%	6.6% (↓13.0%)

This collaborative mechanism greatly improves the system's adaptability and real-time response capability, further verifying the practical feasibility of the strategy proposed in this paper.

6. Conclusion

In order to effectively solve the key technical problems of resource scheduling in cloud computing platforms, this paper proposes an intelligent scheduling structure that integrates multidimensional data modeling, deep prediction, and graph partitioning optimization. The LSTM model is used for short-term resource prediction, reinforcement learning is used for dynamic tuning strategies, and the graph partitioning mechanism is used to further improve task parallel processing capabilities. Through testing on actual datasets, the superiority and effectiveness of the proposed method have been demonstrated. The proposed strategy has excellent scalability and real-time performance, providing effective support for intelligent scheduling practices in complex cloud environments.

References

1. Y. Li, Z. Li, F. Zhang, and R. Qu, "Research on heterogeneous computing resource scheduling mechanism for power information system operation," In *Second International Conference on Physics, Photonics, and Optical Engineering (ICPPOE 2023)*, March, 2024, pp. 245-252.
2. S. Gheisari, and H. ShokrZadeh, "LATA: learning automata-based task assignment on heterogeneous cloud computing platform," *The Journal of Supercomputing*, vol. 80, no. 16, pp. 24106-24137, 2024, doi: 10.1007/s11227-024-06292-6.
3. X. Tang, F. Liu, B. Wang, D. Xu, J. Jiang, Q. Wu, and C. P. Chen, "Workflow scheduling based on asynchronous advantage actor-critic algorithm in multi-cloud environment," *Expert Systems with Applications*, vol. 258, p. 125245, 2024, doi: 10.1016/j.eswa.2024.125245.
4. H. Zhou, "A novel approach to cloud resource management: hybrid machine learning and task scheduling," *Journal of Grid Computing*, vol. 21, no. 4, p. 68, 2023, doi: 10.1007/s10723-023-09702-w.
5. Y. Xing, "Work scheduling in cloud network based on deep Q-LSTM models for efficient resource utilization," *Journal of Grid Computing*, vol. 22, no. 1, p. 36, 2024, doi: 10.1007/s10723-024-09746-6.