Journal of Computer Technology and Software

ISSN:2998-2383

Vol. 3, No. 6, 2024

Continuous Control-Based Load Balancing for Distributed Systems Using TD3 Reinforcement Learning

Yifei Duan
University of Pennsylvania, Philadelphia, USA
dyifei@seas.upenn.edu

Abstract: This paper addresses the issue of resource waste and performance degradation caused by load imbalance in distributed systems. It proposes a load balancing optimization method based on the TD3 (Twin Delayed Deep Deterministic Policy Gradient) algorithm. By modeling system scheduling as a Markov Decision Process with a continuous action space, the agent can dynamically generate task migration ratios based on system states, enabling precise control over multi-node resource allocation. A state space is designed that includes indicators such as CPU utilization and task queue length. A reward function is constructed to comprehensively account for latency, resource utilization, and migration overhead. In multiple experimental scenarios, the proposed method outperforms mainstream algorithms such as Q-Learning, DQN, and PPO in terms of average latency, resource usage, and scheduling robustness. Further tests are conducted in environments involving multi-resource collaborative scheduling and node failure disturbances, validating the strategy's stability and adaptability. The experiments also demonstrate the convergence of the model during training, indicating that the proposed strategy is highly trainable and generalizable. It effectively enhances the overall scheduling efficiency of distributed systems.

Keywords: Distributed system; load balancing; reinforcement learning; TD3 algorithm

1. Introduction

With the rapid development of cloud computing, big data, and microservices architecture, distributed systems have become a fundamental component of modern computing infrastructure [1]. As the scale of these systems continues to grow, imbalanced task distribution among nodes has become increasingly prominent. This imbalance significantly impacts overall system performance and resource utilization. Load balancing plays a critical role in distributed systems. Its core objective is to efficiently schedule requests and resources to maintain stable operation under conditions of high concurrency and availability [2]. However, traditional load-balancing strategies largely rely on static rules or predefined heuristic methods. These approaches lack adaptability to dynamic environments and fall short of meeting the rising demand for intelligent and adaptive solutions in complex application scenarios.

In recent years, reinforcement learning has demonstrated strong adaptability and optimization potential in areas such as automatic control and intelligent scheduling. This has introduced new perspectives for addressing load balancing challenges in distributed systems. Reinforcement learning interacts with the environment to iteratively update its policy in pursuit of maximizing long-term rewards [3]. This makes it well-suited for dealing with the complex and interrelated factors in distributed systems, such as task distribution, resource status, and node responsiveness. In practical applications, reinforcement learning can dynamically optimize request scheduling and task migration. This leads to improved resource utilization, reduced response time, and enhanced system fault tolerance and robustness. As a result, the integration of reinforcement learning into distributed system load balancing has become a focal point in both academic research and engineering practice [4, 5].

Although reinforcement learning has achieved some success in load balancing applications, most existing studies focus on modeling problems in discrete action spaces. This limits the expressiveness and optimization accuracy of the learned policies. In reality, many scheduling behaviors in distributed systems — such as task migration ratios and resource weight adjustments — are inherently continuous in nature. Therefore, there is a need for reinforcement learning methods capable of handling high-dimensional, continuous action spaces to more accurately simulate real-world scheduling behaviors and enable fine-grained and efficient strategy optimization [6, 7].

TD3 (Twin Delayed Deep Deterministic Policy Gradient) is a reinforcement learning algorithm specifically designed for continuous action space optimization. It has shown strong performance in high-dimensional policy learning tasks. By employing dual policy networks and delayed updates, TD3 mitigates the problem of overestimation in value functions. This improves the stability of the learning process and the quality of the resulting policies. Applying TD3 to load balancing in distributed systems enables continuous decision-making under dynamic load and resource conditions. This enhances the intelligence of scheduling strategies and improves overall system performance. The method also demonstrates strong generalization capabilities and has the potential to address the scheduling bottlenecks faced by traditional strategies in rapidly changing environments.

This study aims to develop a load-balancing optimization framework for distributed systems based on the TD3 algorithm. It explores how continuous policy learning can achieve optimal resource allocation and scheduling. A multi-node distributed environment will be simulated to design state spaces, action spaces, and reward mechanisms tailored to different system conditions. The study will evaluate the convergence and performance of reinforcement learning in real-world system scenarios. Theoretically, this work enriches the application paradigm of reinforcement learning in resource scheduling problems. Practically, it offers an efficient and intelligent optimization approach for resource management in large-scale distributed systems, with significant theoretical value and practical prospects.

2. Related work

Achieving efficient load balancing in distributed systems has long been a key topic in scheduling research. Traditional methods, such as Round-Robin, Least Connection, or threshold-based static rules, are commonly used to allocate tasks. These approaches perform reasonably well in small-scale or evenly loaded systems [8, 9, 10]. However, in complex environments with significant differences in node performance and frequent load fluctuations, they often lead to local node overload and degraded system responsiveness. Moreover, such methods lack deep awareness of system states and cannot adjust scheduling strategies based on historical data, resulting in clear performance bottlenecks when handling large-scale, heterogeneous, and high-concurrency workloads [11].

In recent years, with the rise of machine learning in scheduling optimization, researchers have explored the use of supervised learning and deep learning methods to improve load balancing strategies. For example, some studies train regression models on historical data to predict node load and then make scheduling decisions accordingly. Others use neural networks to model resource demands, enhancing the foresight and flexibility of scheduling[12]. However, these approaches often rely on large amounts of labeled data and struggle to adapt to dynamic changes in the system environment. In contrast, reinforcement learning learns optimal strategies through agent-environment interaction. It offers strong adaptability and does not require labeled supervision, making it a growing focus in load scheduling research[13].

Among reinforcement learning methods, algorithms such as Deep Q-Network (DQN), Proximal Policy Optimization (PPO), and Deep Deterministic Policy Gradient (DDPG) are widely applied in scheduling decision scenarios. Existing studies have explored their effectiveness in cloud resource allocation, virtual machine migration, and container scheduling. Some research has also introduced multi-agent learning into scheduling systems to enhance cooperation among nodes. However, most of these methods are based on discrete action spaces, which makes it difficult to capture optimal strategies for continuous decision variables like task migration ratios or resource adjustment magnitudes. To address this limitation, some studies have begun to explore the advantages of reinforcement learning algorithms such as TD3 in continuous action spaces. This paper, under this context, focuses on the specific application and performance validation of the TD3 algorithm in distributed system load balancing.

3. Method

This study uses the TD3 algorithm (Twin Delayed Deep Deterministic Policy Gradient) to build an optimization model for distributed system load balancing, and learns the optimal resource scheduling strategy through continuous interaction between the agent and the system environment. The model architecture is shown in Figure 1.

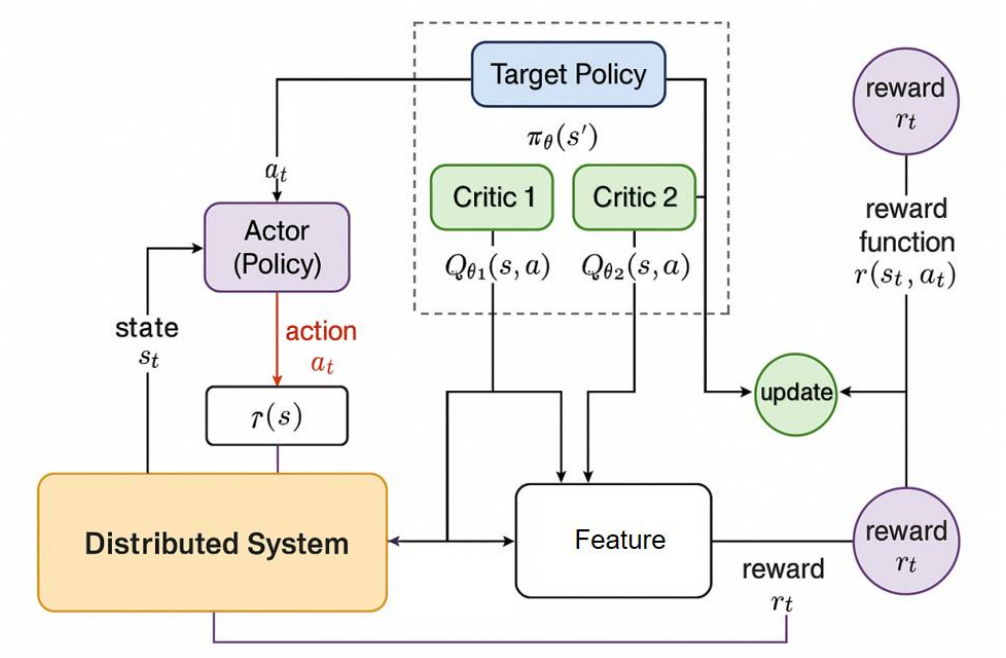

Figure 1. Overall model architecture diagram

This network architecture diagram shows the load balancing optimization process based on the TD3 algorithm. The agent generates continuous actions a_t according to the current system state s_t through the Actor network to realize task scheduling in the distributed system. The two Critic networks calculate the Q value of the action respectively, and take the minimum value for the target value calculation to improve the stability of the policy estimation. The reward function comprehensively considers the system delay, resource utilization and migration overhead, and continuously optimizes the policy network through the interactive feedback mechanism to realize intelligent balanced scheduling of the system load.

The system environment is modeled as a Markov decision process (MDP), defined as a five-tuple (S, A, R, P, γ), where S represents the system state space, A represents the action space, R is the reward function, P represents the state transition probability, and γ is the discount factor. The state space includes indicators such as the CPU usage, memory usage, and current task queue length of each node; the action space is the continuous decision of the agent on the task migration ratio at each time step.

The goal of the agent is to maximize the future cumulative expected reward $J(\pi)$, that is:

$$J(\pi) = E_{s_0 \sim p}[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)]$$

Among them, strategy π is a deterministic strategy, which means action $a_t = \pi(s_t)$ taken in state s_t. The TD3 algorithm uses two independent Q value functions Q_{θ_1} and Q_{θ_2} to alleviate the problem of strategy overestimation, and uses the smaller one when updating the target Q value to enhance stability. The target Q value is defined as follows:

$$y = r + \gamma \cdot \min_{i=1,2} Q_{\theta'_i}(s', \pi_\phi(s') + \varepsilon)$$

Among them, ε is an exploration item that adds Gaussian noise to enhance the robustness of the strategy. The policy network π_ϕ updates the parameters by maximizing the output of the Q function, and the optimization goal is:

$$\nabla_\phi J(\phi) = E_{s \sim D}[\nabla_a Q_{\theta_1}(s, a)|_{a=\pi_\phi(s)} \cdot \nabla_\phi \pi_\phi(s)]$$

During the training process, TD3 uses a delayed update mechanism, that is, the policy network is updated once every d times the Q network is updated to improve the stability of the policy update. At the same time, in order to suppress policy drift, the target network parameters are updated using a soft update strategy after each round of update:

$$\theta' \leftarrow \tau\theta + (1-\tau)\theta'$$

In order to enable the agent to learn to make optimal load migration decisions in a dynamic system, the reward function is designed to take into account system latency, resource utilization, and migration cost. The reward function is defined as:

$$r(st, at) = -\alpha \cdot \text{avg\_latency} + \beta \cdot \text{resource\_utilization} - \delta \cdot \text{migration\_cost}$$

Among them, α, β, δ is a weight factor used to balance the system performance indicators. The above method enables the intelligent agent to optimize the strategy in the high-dimensional continuous state-action space, effectively improve the dynamic scheduling capability and resource utilization efficiency of the system, and realize adaptive and efficient load balancing scheduling.

4. Experiment

4.1 Datasets

This study adopts the publicly available "ClusterData: Google Cluster Trace" dataset as the foundational data source for distributed system load balancing optimization experiments. The dataset originates from the scheduling and resource usage logs of a large-scale real-world cluster. It includes operational data from over 10,000 nodes over a one-month period, covering key metrics such as CPU usage, memory consumption, task requests, and job durations.

The dataset records detailed information on job submissions, task scheduling, resource allocation, and task completions within the cluster scheduling system. It accurately reflects dynamic load variations and fluctuations in resource states, making it well-suited for modeling the state space and reward functions in a reinforcement learning environment. Additionally, its high-frequency and multi-dimensional sampling structure provides strong support for training policies in continuous action spaces.

During data preprocessing, representative time segments and node samples were selected. Missing values and outliers were cleaned, and fields such as CPU usage, memory occupancy, and task length were normalized. These steps enabled the construction of essential state variables and reward feedback for the reinforcement learning environment. This dataset provides a highly reliable experimental foundation for optimizing load-balancing strategies.

4.2 Experimental Results

1) Experiments comparing this algorithm with other algorithms

This paper first gives the comparative experimental results, and the experimental results are shown in Table 1.

Table 1: Comparative experimental results

Method	Average Latency (ms)	CPU Utilization (%)	Task Success Rate (%)	Migration Cost
Q-Learning[14]	128.6	73.2	91.4	0.38
DQN[15]	115.3	76.8	93.1	0.42
PPO[16]	104.7	81.5	95.8	0.36
DDPG[17]	97.2	83.9	96.7	0.34

Ours	91.6	86.4	97.5	0.31

As shown in the table, the TD3-based load balancing strategy (Ours) outperforms other methods across multiple key performance metrics. In terms of average latency, the TD3 algorithm achieves a significantly lower value of 91.6 ms, which is markedly better than other reinforcement learning algorithms. This indicates stronger real-time responsiveness in high-concurrency scheduling scenarios and more effective mitigation of imbalance pressure among system nodes.

Regarding resource utilization, the TD3 strategy achieves a CPU utilization rate of 86.4%, showing a clear improvement over other methods. This demonstrates its ability to finely control task scheduling behavior in continuous action spaces, thereby enabling deeper exploitation of computational resources. The continuous control strategy also avoids the abrupt scheduling changes often observed in discrete strategies, contributing to smoother resource allocation across the system.

Moreover, the TD3 algorithm also delivers superior overall performance in task success rate and migration overhead. With a task success rate of 97.5% and the lowest migration cost of 0.31, it effectively reduces the system burden caused by task migration while ensuring system stability. This highlights its strong scheduling robustness and comprehensive optimization capabilities. These results fully validate the rationality and practical value of applying TD3 to load balancing strategy optimization in distributed systems.

2) Experimental results on the impact of task migration rate on system latency

Furthermore, this paper also gives the experimental results of the impact of task migration rate on system latency, as shown in Figure 2.

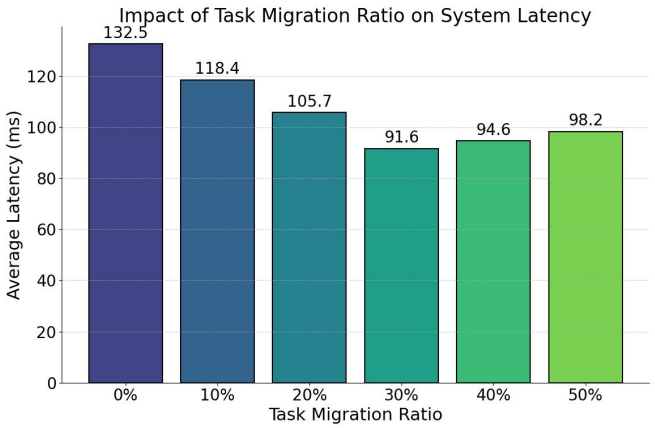

Figure 2. Impact of Task Migration Ratio on System Latency

The experimental results indicate that the task migration ratio has a significant impact on the system's average latency. When the migration ratio increases gradually from 0% to 30%, the average latency drops significantly from 132.5 ms to 91.6 ms. This suggests that a moderate level of task migration helps achieve dynamic load balancing, effectively alleviating overload issues on individual nodes and improving overall response efficiency.

However, when the migration ratio is further increased to 40% and 50%, the system latency begins to rise. This implies that overly frequent task migrations may introduce additional communication and scheduling overhead. Such overhead can undermine local system stability, causing the benefits of load optimization to be offset by the cost of migration, ultimately leading to a decline in overall performance.

This experiment validates the TD3-based scheduling strategy's fine-grained control over task migration. By learning optimal migration policies under varying load conditions, the model can improve resource utilization while keeping migration-induced delays under control. This demonstrates strong adaptability and system awareness.

3) Robustness experiment of TD3 strategy under different node scales

Furthermore, this paper presents a robustness experiment of the TD3 strategy under different node scales, and the experimental results are shown in Figure 3.

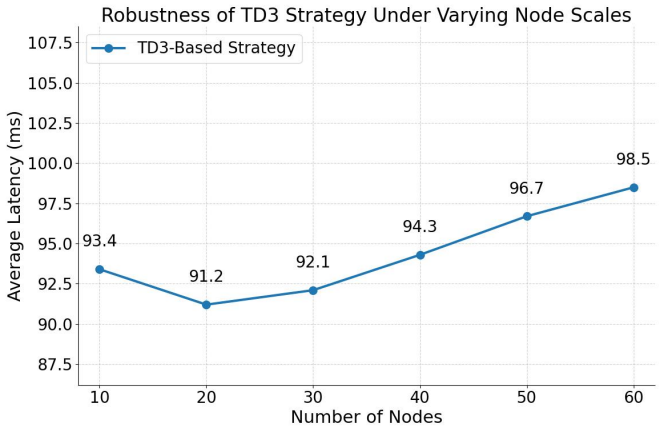

Figure 3. Robustness experiments under different node scales

The experimental results show that the TD3 strategy demonstrates strong robustness under varying node scales. When the number of nodes ranges from 10 to 30, the system's average latency remains between 91 ms and 93 ms, with only minor fluctuations. This indicates that the strategy exhibits good generalization and stable scheduling performance in small-scale systems.

As the number of nodes further increases to 60, the average latency slightly rises to 98.5 ms. This trend suggests that in large-scale environments, uncertainties in system load and increased communication complexity may raise the cost of executing the scheduling strategy. Nevertheless, the overall performance remains within an acceptable range, indicating that the model possesses a certain degree of scalability.

Overall, this experiment confirms the adaptability of the proposed TD3-based scheduling strategy under different node scale scenarios. Through continuous action control, the model can flexibly respond to the load balancing demands of various system structures. It maintains favorable service quality and

response speed, offering strategic support for practical deployment in large-scale distributed systems.

4) Testing the strategy generalization capability in multi-resource collaborative scheduling scenarios

Furthermore, this paper also gives the experimental results of the strategy generalization ability test in the multi-resource collaborative scheduling scenario, as shown in Figure 4. The figure provides a detailed view of how the proposed strategy performs when handling complex scheduling tasks involving multiple resource types, demonstrating its capability to generalize across varying system configurations and workload distributions.

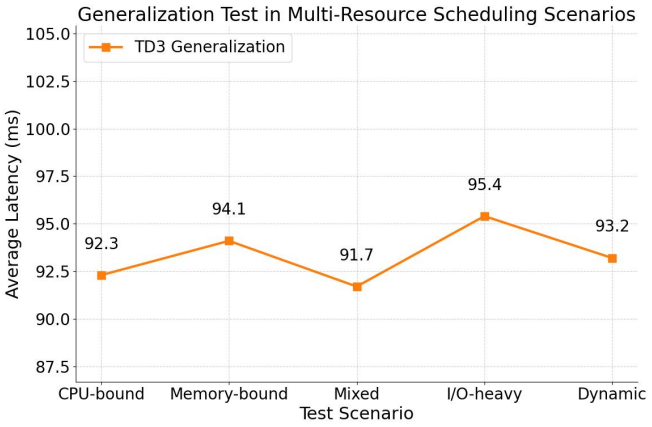

Figure 4. Testing the strategy generalization capability in multi-resource collaborative scheduling scenarios

The experimental results show that the proposed TD3 strategy maintains relatively stable performance under various resource-constrained scenarios. The average latency fluctuates within a narrow range, with a minimum of 91.7 ms in the Mixed scenario and a maximum of 95.4 ms in the I/O-heavy scenario. This reflects the model's strong adaptability and generalization when facing different resource bottlenecks. Notably, achieving the lowest latency under mixed resource loads indicates high scheduling sensitivity in balancing multi-resource allocations.

In the Memory-bound and I/O-heavy scenarios, system latency increases slightly to 94.1 ms and 95.4 ms, respectively. This may be attributed to the inherent access characteristics of these resources and the finer scheduling granularity they require. However, the latency remains within an acceptable range, demonstrating the reinforcement learning strategy's stability in non-ideal conditions. Particularly in I/O-intensive scheduling tasks, the strategy effectively controls latency, indicating a strong responsiveness to changes in system state.

Overall, this experiment confirms that the scheduling policy learned by the TD3 algorithm is effective not only in the training environment but also in unseen test scenarios. The strategy demonstrates consistent performance when applied to previously unencountered system conditions, indicating that it has successfully captured generalized scheduling patterns. This generalization capability is crucial for practical deployment across diverse resource scheduling conditions, where system dynamics and workload characteristics may vary significantly. Moreover, it provides strong theoretical support for policy transfer and cross-scenario adaptation, enabling the model to be extended to other distributed systems with minimal retraining or reconfiguration.

5) Loss function changes with epoch

This paper also provides a graph illustrating the change of the loss function over the course of training epochs, as presented in Figure 5. The figure visually demonstrates how the model's loss evolves during the learning process, offering insights into the convergence behavior and training stability of the proposed algorithm.

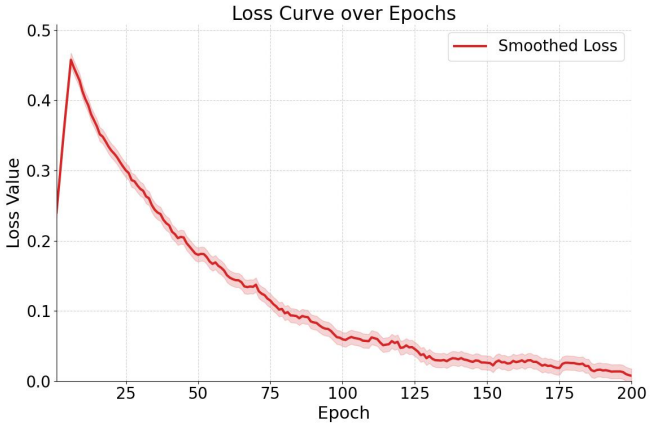

Figure 5. Loss function drop graph

From the loss function curve, it can be observed that the model's loss value drops rapidly during the early training phase. This indicates that the TD3 strategy quickly learns representative scheduling behaviors within the first few epochs. It shows strong initial responsiveness to system states and load feedback. The steep decline at this stage reflects the strategy's fast adaptation and fitting to the dynamic environment.

As training progresses, especially after the 100th epoch, the loss value gradually stabilizes. It fluctuates within a narrow range and remains below 0.05. This suggests that the model has largely converged. Both the policy network and the value network reach a relatively steady state, enabling accurate evaluation of load conditions and execution of appropriate scheduling decisions in the distributed system.

Overall, the results validate the trainability and convergence of the TD3 model for continuous control in load balancing tasks. The steadily declining loss function curve further implies that the model continuously improves its policy through iterative optimization. This trend reflects a stable learning process in which the policy and value networks gradually adapt to the system dynamics. As training progresses, the model becomes increasingly capable of making precise scheduling decisions. These findings lay a solid foundation for the practical deployment of the proposed approach and demonstrate its potential for generalization across diverse and dynamic real-world scenarios.

6) Dynamic testing of real-time interference to strategy adaptability

Finally, this paper also presents the dynamic test results of the real-time interference to the strategy adaptability, as shown in

Figure 6. The figure highlights how the proposed strategy responds to unexpected disturbances during system operation, providing visual evidence of its adaptability and robustness in dynamic environments.

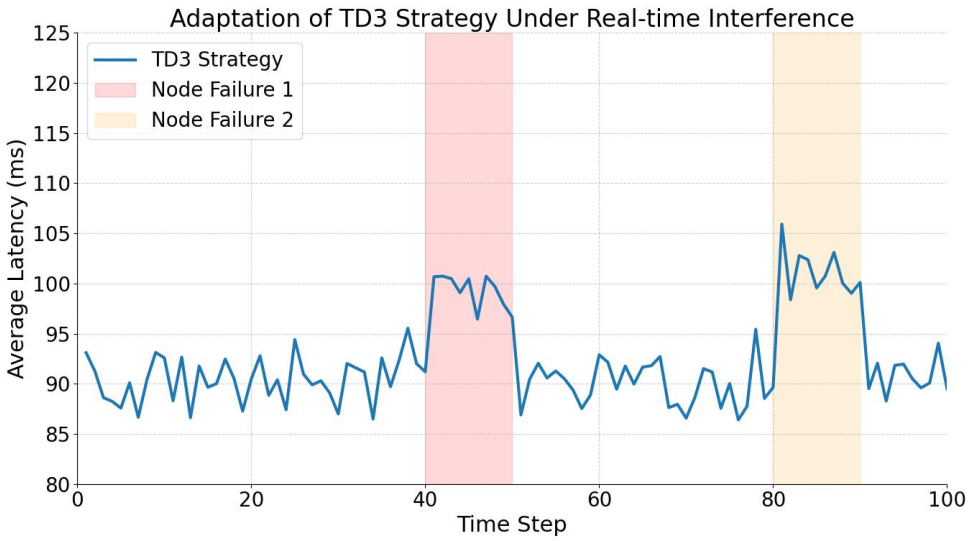

Figure 6. Adaptation of TD3 Strategy Under Real-time Interference

As shown in the figure, before any node failures occur, the system latency remains around 90 ms with minimal fluctuation. This indicates that the TD3 strategy effectively maintains load balance and scheduling stability under normal operating conditions. During this phase, the model responds smoothly to changes in system state, and the policy output shows strong control consistency.

When the first node failure occurs between time steps 40 and 50, a temporary increase in latency is observed, reaching approximately 100 ms. However, the system quickly recovers to normal levels, demonstrating the strategy's ability to absorb disturbances. Similarly, during the second disturbance between time steps 80 and 90, the system again experiences a short-lived fluctuation followed by rapid stabilization. This shows that the model can adapt quickly to unknown disruptions, maintaining overall system performance without long-term degradation.

These experimental results validate that the TD3-based load balancing strategy exhibits strong environmental adaptability and dynamic adjustment capabilities in the face of sudden node dropouts and other real-time disturbances. This feature is crucial for improving the reliability and robustness of distributed systems in real-world deployments.

5. Conclusion

This study addresses the problem of load balancing in distributed systems and proposes a scheduling strategy optimization method based on the TD3 algorithm. By introducing a continuous action space reinforcement learning mechanism into task migration and resource scheduling, the method enables fine-grained perception of system states and dynamic decision-making optimization. Experimental results show that the proposed method delivers excellent scheduling performance across various typical scenarios. It significantly reduces system latency, improves resource utilization, and effectively controls migration costs.

Under complex conditions such as system scaling, multi-resource collaborative scheduling, and unexpected fault disturbances, the proposed strategy maintains strong stability and robustness. It demonstrates good generalization capability and practical value. In particular, under dynamic disturbances and resource bottlenecks, the model adjusts its strategy promptly in response to changes, highlighting the potential of deep reinforcement learning in controlling uncertain systems.

The proposed method not only offers an efficient and adaptive solution to load balancing in distributed systems but also provides new theoretical and experimental support for the application of reinforcement learning in intelligent scheduling, edge computing, and large-scale cloud infrastructure management. The flexibility of the model architecture and the continuity of policy representation enable its feasibility for migration and deployment across diverse system architectures. Future work may consider integrating multi-agent learning mechanisms into the scheduling framework to enhance collaborative decision-making among nodes. In addition, incorporating more complex network topologies and modeling data communication delays in realistic deployment environments could improve policy adaptability to real-world scenarios. Furthermore, introducing advanced techniques such as self-supervised learning and meta-learning may offer new directions for rapid policy generalization and autonomous evolution.

References

[1] Zhou, Guangyao, Wenhong Tian, Rajkumar Buyya, Ruini Xue, and Liang Song. "Deep Reinforcement Learning-based Methods for Resource Scheduling in Cloud Computing: A Review and Future Directions." arXiv preprint arXiv:2105.04086 (2021).

[2] Lu, Yu, et al. "Deep reinforcement learning based optimal scheduling of active distribution system considering distributed generation, energy storage and flexible load." Energy 271 (2023): 127087.

[3] Al-Saadi, Mudhafar, Maher Al-Greer, and Michael Short. "Reinforcement learning-based intelligent control strategies for optimal power management in advanced power distribution systems: A survey." Energies 16.4 (2023): 1608.

[4] Liu, Ji, et al. "Heterps: Distributed deep learning with reinforcement learning based scheduling in heterogeneous environments." Future Generation Computer Systems 148 (2023): 106-117.

[5] Pei, Yansong, et al. "Multi-task reinforcement learning for distribution system voltage control with topology changes." IEEE Transactions on Smart Grid 14.3 (2023): 2481-2484.

[6] Cao, Di, et al. "Physics-informed graphical representation-enabled deep reinforcement learning for robust distribution system voltage control." IEEE Transactions on Smart Grid 15.1 (2023): 233-246.

[7] Kiumarsi, Bahare, K. G. Vamvoudakis, H. Modares, and Frank L. Lewis. "Optimal and Autonomous Control Using Reinforcement Learning: A Survey." IEEE Transactions on Neural Networks and Learning Systems, vol. 29, no. 6, pp. 2042–2061, June 2018.

[8] Zhang, Bin, et al. "Multi-agent deep reinforcement learning based distributed control architecture for interconnected multi-energy microgrid energy management and optimization." Energy Conversion and Management 277 (2023): 116647.

[9] Gao, Hongjun, et al. "A cloud-edge collaboration solution for distribution network reconfiguration using multi-agent deep reinforcement learning." IEEE Transactions on Power Systems 39.2 (2023): 3867-3879.

[10] Xiang, Yue, Yu Lu, and Junyong Liu. "Deep reinforcement learning based topology-aware voltage regulation of distribution networks with distributed energy storage." Applied Energy 332 (2023): 120510.

[11] Alfaverh, Fayiz, Mouloud Denaï, and Yichuang Sun. "Optimal vehicle-to-grid control for supplementary frequency regulation using deep reinforcement learning." Electric Power Systems Research 214 (2023): 108949.

[12] He, Nan, et al. "Leveraging deep reinforcement learning with attention mechanism for virtual network function placement and routing." IEEE Transactions on Parallel and Distributed Systems 34.4 (2023): 1186-1201.

[13] Gu, Jinlei, et al. "A metaverse-based teaching building evacuation training system with deep reinforcement learning." IEEE Transactions on Systems, Man, and Cybernetics: Systems 53.4 (2023): 2209-2219.

[14] Jayakumar, Steffi, and S. Nandakumar. "Reinforcement learning based distributed resource allocation technique in device-to-device (D2D) communication." Wireless Networks 29, no. 4 (2023): 1843–1858.

[15] Llorens-Carrodeguas, Alejandro, Cristina Cervelló-Pastor, and Francisco Valera. "DQN-based intelligent controller for multiple edge domains." Journal of Network and Computer Applications 218 (2023): 103705.

[16] Shi, Wenlong, et al. "Coordinated operation of active distribution network, networked microgrids, and electric vehicle: A multi-agent PPO optimization method." CSEE Journal of Power and Energy Systems (2023).

[17] Wu, Q., Zhang, Z., Zhu, H., Fan, P., Fan, Q., Zhu, H., & Wang, J. (2023). Deep reinforcement learning-based power allocation for minimizing age of information and energy consumption in multi-input multi-output and non-orthogonal multiple access internet of things systems. Sensors, 23(24), 9687.