Artificial Intelligence and Computing Innovations

Intelligence Science Publishing

https://www.ispublishing.com

ISSN: 3070-4944

Vo.4, No.4, 2024

Reinforcement Learning-Based Elastic Scaling Policy Optimization for Microservices

Xuexian Li[1]*, Zichen Song[2]

[1]Northeastern University, Boston, USA

[2]George Mason University, Fairfax, USA

**Corresponding author: ericli15112437668@gmail.com*

Abstract: This paper focuses on the intelligent optimization of elastic scaling strategies in microservice architectures. It proposes an adaptive resource control method based on reinforcement learning. The method constructs a multi-dimensional state space that incorporates key performance indicators such as CPU utilization, response time, and queue length. A policy network is used to make dynamic decisions on scaling actions. To improve learning efficiency and system responsiveness, the method introduces an advantage function for policy optimization. It also integrates a temporal dependency modeling mechanism to enhance the model's awareness of historical workload patterns. A reward function is designed based on multiple system metrics. It jointly considers resource cost, performance assurance, and system stability, enabling fine-grained guidance of the policy behavior. In the experimental section, a sensitivity analysis is conducted with different discount factors, dynamic workload patterns, and environmental disturbances. The evaluation systematically examines the stability and robustness of the proposed method. Comparative results with mainstream baselines show significant improvements in accuracy, precision, and recall. The results demonstrate strong policy convergence and efficient scheduling, confirming the effectiveness of the proposed approach in microservice elastic control tasks.

Keywords: Policy network; reinforcement learning; microservice scheduling; system robustness

1. Introduction

In the context of rapid digital transformation and the fast advancement of cloud computing technologies, microservice architecture has emerged as a mainstream approach for building large-scale distributed systems. Its modularity, decoupling capability, and elasticity allow systems to better adapt to changing business requirements while improving maintainability and scalability [1]. However, with the widespread adoption of microservices, issues related to resource management and service orchestration have become increasingly prominent. In particular, under dynamic workloads and unpredictable traffic surges, how to

effectively implement elastic scaling has become a key challenge in ensuring system stability and performance [2].

Traditional elastic scaling strategies in microservices typically rely on static thresholds or rule-based control mechanisms. Although these methods are simple to implement, they lack global awareness of the system state and often fail to adapt to complex and variable runtime environments. For example, in high-concurrency scenarios, static resource configurations may not respond in time to sudden demand spikes, leading to performance degradation or system failure. In contrast, during low-load periods, excessive resource allocation can result in significant waste and increased operational costs. Therefore, developing an elastic scaling mechanism with dynamic awareness and intelligent decision-making has become essential for advancing microservice systems [3].

In recent years, with the rise of intelligent systems, reinforcement learning has shown great potential in areas such as resource scheduling and load balancing. As a trial-and-error-based decision-making method, reinforcement learning continuously optimizes strategies through interaction with the environment. It demonstrates strong adaptability and generalization, especially in complex systems where precise models or prior knowledge are lacking. Applying reinforcement learning to microservice elastic scaling can effectively overcome the limitations of traditional approaches. It enables adaptive control of resources concerning performance, cost, and stability, thereby enhancing system efficiency and service quality [4].

Furthermore, microservice systems exhibit high dynamics and heterogeneity. This includes information such as service dependencies, invocation latency, and container resource utilization. These dimensions often have strong temporal and nonlinear correlations. In this context, reinforcement learning must not only perceive and respond to current states but also optimize long-term strategies. This is essential to balance resource utilization efficiency with long-term system performance. Therefore, modeling reinforcement learning specifically for microservice environments and optimizing its policies is a critical task for implementing intelligent resource scheduling mechanisms.

Driven by the rapid evolution of multi-cloud deployments, container orchestration technologies, and service meshes, microservice environments have become increasingly diverse in deployment and operation. This places higher demands on the robustness and generalization of elastic scaling strategies. An effective strategy should make quick decisions under resource constraints while ensuring service-level agreements and minimizing operational costs. Thus, from both system and algorithmic perspectives, it is vital to explore effective reinforcement learning models for elastic control. Building a generalizable, adaptive, and responsive optimization framework will provide intelligent and efficient resource management for cloud-native applications. It also holds significant theoretical and practical value for advancing intelligent operations and maintenance.

2. Related Work

Elastic scaling in microservices has become a critical topic in cloud resource management [5]. Early research mainly focused on static rule configurations and threshold-based scaling strategies. These methods typically trigger scaling actions based on preset indicators such as CPU or memory usage. They are easy to implement and deploy, but often suffer from coarse granularity, slow response, and rigid policy structures. These limitations make them poorly suited to dynamic and volatile workloads. To address this, some studies have introduced adaptive adjustment mechanisms that monitor system load in real time and adjust scaling parameters accordingly. This improves flexibility but still depends heavily on predefined rules or manual parameter tuning. As a result, they fall short of achieving intelligent decision-making and global optimization [6].

With the advancement of forecasting techniques, some approaches have adopted time series analysis and load prediction to anticipate future resource demands. These methods often use sliding windows, ARIMA, or LSTM models for forecasting and then apply predictive insights to guide scaling decisions [7].

While this improves resource utilization efficiency to some extent, the strategy is highly dependent on prediction accuracy. When sudden load spikes occur or the model fails, the system may not respond effectively. Moreover, prediction-driven strategies often lack feedback from real system performance. They do not form a closed-loop optimization process and fail to refine their decisions through runtime outcomes.

To overcome these challenges, recent studies have increasingly applied reinforcement learning for policy optimization through interaction with the environment. Reinforcement learning has shown promising results in areas like resource scheduling, virtual machine management, and container orchestration. Its adaptability and exploratory capabilities make it well-suited for elastic scaling scenarios. Techniques include value function approximation, policy gradient methods, and model-based learning. These approaches model system states, actions, and rewards to continuously improve scheduling policies in complex environments. Researchers are also exploring how to integrate reinforcement learning with system-specific features, such as priority modeling, resource topology awareness, and latency constraints. These efforts aim to enhance the feasibility and applicability of learned strategies.

However, challenges remain in applying reinforcement learning to microservice elastic scaling. These include high-dimensional state spaces, sparse feedback signals, and slow convergence. In real-world deployments, reinforcement learning models often struggle to balance training efficiency with deployment controllability. In addition, designing multi-objective optimization mechanisms that consider cost, SLA compliance, and resource efficiency is a major research difficulty. Recent research on reliability-aware learning systems has highlighted the importance of enhancing model perception and representation capabilities in complex and uncertain environments, providing useful insights for improving state modeling quality and decision stability in intelligent control tasks [8]. Therefore, improving the architecture of reinforcement learning algorithms, enhancing their ability to perceive and model complex system states, and strengthening the stability and transferability of policies remain key directions in the field of microservice elasticity controll.

3. Method

To solve the problem of dynamic resource scheduling and elastic scaling in microservice systems, this paper proposes a policy optimization method based on reinforcement learning, which aims to achieve intelligent control of the number of service instances. The entire method takes the current state of the microservice system as input, learns a policy function through the agent, generates actions based on multi-dimensional indicators such as system load, latency, and resource utilization, and dynamically adjusts the number of service replicas. The model architecture is shown in Figure 1.

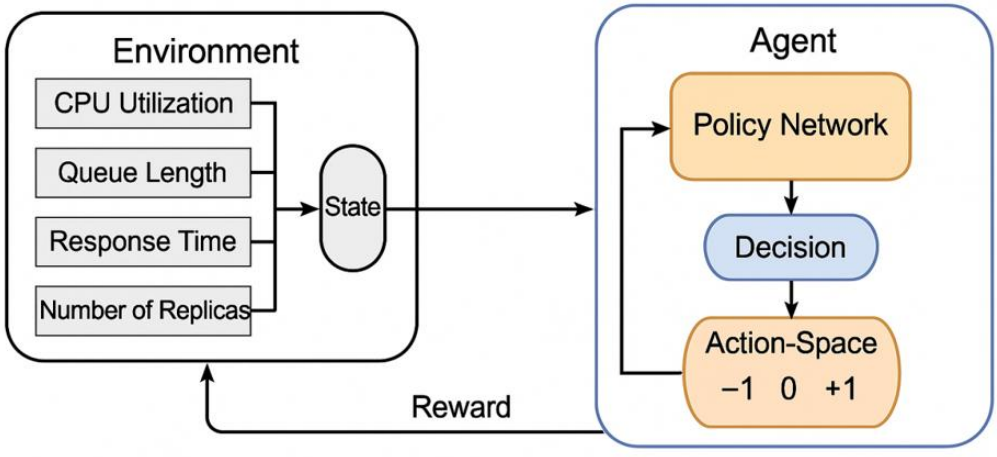

Figure 1. Overall model architecture

The system state is modeled as a multidimensional vector, which includes key indicators such as service CPU utilization u_t, request queue length q_t, response time d_t, etc., forming the state space $S_t = [u_t, q_t, d_t, ...]$. The corresponding action space is defined as the adjustment operation $A_t \in \{-1,0,+1\}$ on the number of service replicas, which represents shrinking, keeping unchanged, and expanding.

In the policy optimization process, the reinforcement learning agent continuously updates its policy network parameters by interacting with the environment. We use the policy gradient method in deep reinforcement learning, to maximize the expected cumulative reward function $J(\theta)$, which is defined as:

$$J(\theta) = E_{\pi\theta}[\sum_{t=0}^{\infty} \gamma^t R_t] \tag{1}$$

Where θ represents the parameters of the policy network, $\gamma \in (0,1)$ is the discount factor, and R_t is the instant reward at time t. The design of the reward function takes into account both service performance and resource cost to encourage the system to reduce resource waste while meeting performance indicators. Specifically, the reward function is defined as follows:

$$R_t = -\alpha \cdot d_t + \beta \cdot \Delta u_t - \lambda \cdot c_t \tag{2}$$

d_t represents the service response time, Δu_t represents the degree of improvement in resource utilization, c_t represents the current resource cost, and α、β、λ is the hyperparameter that controls the weight of each indicator.

To enhance the stability and generalization ability of policy learning, this paper introduces the advantage function $A(s_t, a_t)$ to replace the traditional value function difference estimation, thereby improving the accuracy of policy gradient estimation. The advantage function is defined as follows:

$$A(s_t, a_t) = Q(s_t, a_t) - V(s_t) \tag{3}$$

Where $Q(s_t, a_t)$ represents the action value after taking action s_t in state a_t, and $V(s_t)$ represents the state value function in this state. The policy gradient optimization objective combined with the advantage function can be further expressed as:

$$\nabla_\theta J(\theta) = E_{s_t, a_t \sim \pi\theta}[\nabla_\theta \log_{\pi\theta}(a_t \mid s_t) \cdot A(s_t, a_t)] \tag{4}$$

In addition, considering that the state changes in the actual system have obvious temporal correlation, time-dependent mechanisms such as recurrent neural networks or attention mechanisms are introduced into the policy network design to model the impact of historical states on current decisions. Through the above modeling, the agent can make more reasonable inferences about the operating dynamics of the service system and realize elastic scaling control for different workloads, thereby achieving the dual goals of performance guarantee and resource optimization.

4. Experimental Results

4.1 Dataset

This study uses the Train-ticket microservice dataset as the basis for validating the reinforcement learning-based policy optimization method. The dataset is widely adopted in research on resource scheduling, service dependency analysis, and elastic control in microservice systems. It is constructed based on a real-world high-speed rail ticketing scenario. The dataset simulates a complete multi-service collaborative system composed of more than ten interdependent service modules, including user

authentication, train inquiry, order management, seat reservation, and payment processing. These services form a complex invocation chain and reflect typical characteristics of microservice architecture.

The dataset provides a large number of runtime metrics. These include CPU usage, memory consumption, response time, request rate, container replica count, and queue length for each service module. It also covers various workload scenarios such as traffic bursts, high-concurrency access, and resource bottlenecks. These features enable a comprehensive evaluation of the adaptability and stability of elastic scaling strategies under different load conditions. In addition, the dataset includes service topology information and logs of inter-service calls. These elements offer rich support for modeling the state space in reinforcement learning.

The choice of this dataset allows effective validation of the proposed method's generalization capability and dynamic scheduling efficiency in complex service topologies. Its controllable simulation environment and comprehensive metric records provide a solid foundation for training reinforcement learning models and evaluating policies. This also facilitates the extension of future research into real-world container orchestration systems.

4.2 Experimental Results

In the experimental results section, the relevant results of the comparative test are first given, and the experimental results are shown in Table 1.

Table 1. Comparative Experimental Results

Method	Accuracy	Precision	Recall
DeepScaler [9]	87.4	84.2	85.6
Reclaimer [10]	88.9	85.7	86.8
MSARS [11]	90.3	88.1	87.4
Gym-hpa [12]	91.1	89.3	89.0
Waterfall ML [13]	92.5	90.6	90.2
Ours	94.2	92.8	93.1

From the comparative experimental results, all methods demonstrate certain scheduling capabilities in microservice elastic scaling scenarios, but their overall performance varies significantly. Traditional approaches such as DeepScaler and Reclaimer show relatively low accuracy and precision. This is mainly due to their reliance on fixed rules or static models, which are unable to adapt to dynamic changes in system states. Their response capacity is particularly limited when facing sudden workload spikes or complex inter-service dependencies. These methods lack closed-loop optimization and provide only simple feedback mechanisms, resulting in unstable scaling behavior under highly dynamic conditions. Their reclaiming efficiency is also constrained.

MSARS and Gym-hpa represent a class of reinforcement-based scheduling methods that rely on policy optimization or simulation learning. Compared to traditional rule-based strategies, they achieve notable performance improvements. These methods optimize scheduling policies through interaction with the environment and exhibit a certain level of adaptability. However, they still face limitations in state modeling and policy updating. They do not fully integrate multi-dimensional system indicators or long-term dependencies. As a result, their ability to balance resource cost and service quality remains suboptimal. For example, MSARS achieves high precision but shows relatively low recall, indicating that it may not strike the best balance between resource savings and performance guarantees.

Waterfall ML delivers a more stable overall performance. This is due to its combination of machine learning-based prediction with policy execution, which enhances both foresight and execution robustness. Its precision and recall are both at high levels, suggesting strong long-term scheduling capabilities. However, the method still depends heavily on prediction accuracy. Its real-time responsiveness under

unexpected loads requires further improvement. Additionally, the model often demands extensive parameter tuning, which increases deployment complexity.

In contrast, the method proposed in this study achieves the best results across all evaluation metrics. It reaches an accuracy of 94.2 percent, with precision and recall of 92.8 percent and 93.1 percent, respectively. The method constructs a task-aware state space and introduces temporal dependency modeling along with advantage function optimization. These features significantly improve the convergence speed and adaptability of the policy. Especially under complex service topologies and resource constraints, the reinforcement learning agent can dynamically balance performance assurance and resource utilization. This demonstrates strong policy generalization and high control precision. The results validate the effectiveness and superiority of the proposed method in microservice elastic scaling scenarios.

This paper also gives an analysis of the impact of different discount factors on strategy stability, and the experimental results are shown in Figure 2.

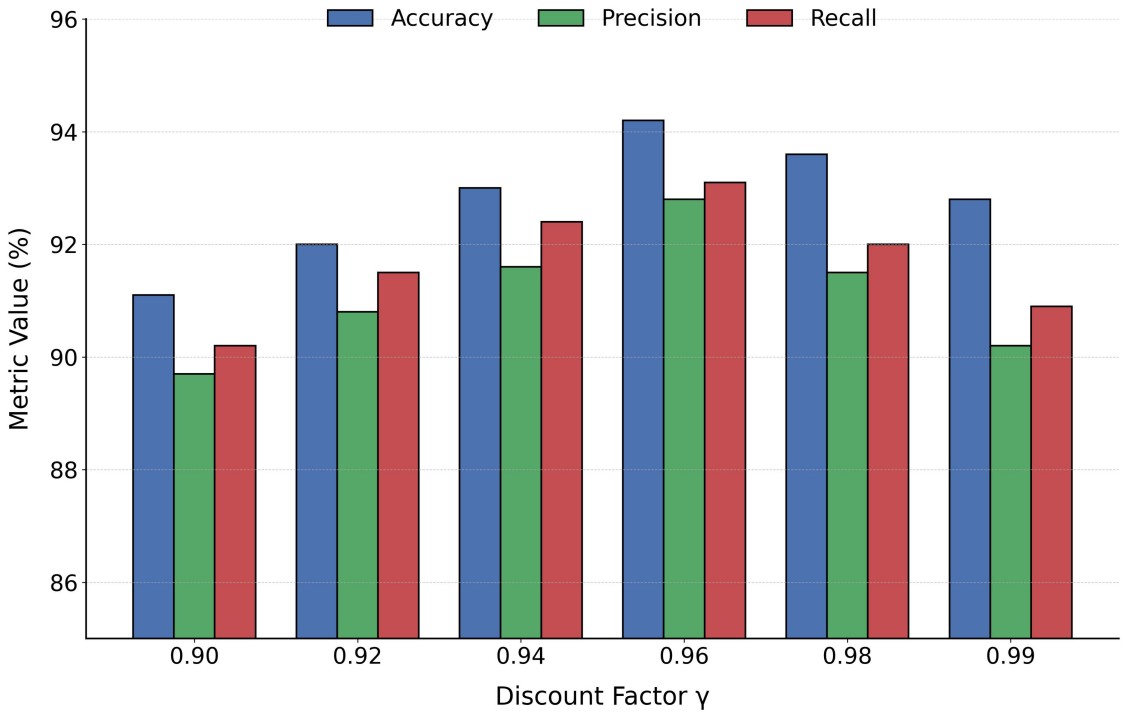

Figure 2. Analysis of the impact of different discount factors on strategy stability

As shown in the figure, the discount factor γ has a significant impact on the stability of the reinforcement learning policy. When γ equals 0.96, the model achieves peak performance with an accuracy of 94.2 percent, a precision of 92.8 percent, and a recall of 93.1 percent. This indicates that under this setting, the agent can achieve optimal service scheduling while maintaining efficient resource utilization. The result suggests that moderately increasing the weight on future rewards helps form more forward-looking and stable elastic control behavior in complex microservice environments.

When γ is set to lower values, such as 0.90 or 0.92, the model performs worse across all three metrics. This is mainly because a lower discount factor causes the policy to focus more on short-term rewards. As a result, the agent struggles to build effective long-term scaling logic when facing dynamic resource fluctuations and temporal dependencies between services. The policy becomes more volatile and conservative, making it less responsive to future workload trends.

As γ increases further to 0.98 and 0.99, the model still maintains relatively high performance, but it performs slightly below the peak achieved at $\gamma = 0.96$. This suggests that placing too much emphasis on long-term rewards may introduce instability during policy learning. The agent may become biased when balancing immediate scheduling decisions with future expectations. This slight performance drop also highlights the need to properly balance short-term actions and long-term optimization in reinforcement learning for elastic control.

Finally, this paper also gives a strategy robustness evaluation under dynamic load mode, and the experimental results are shown in Figure 3.

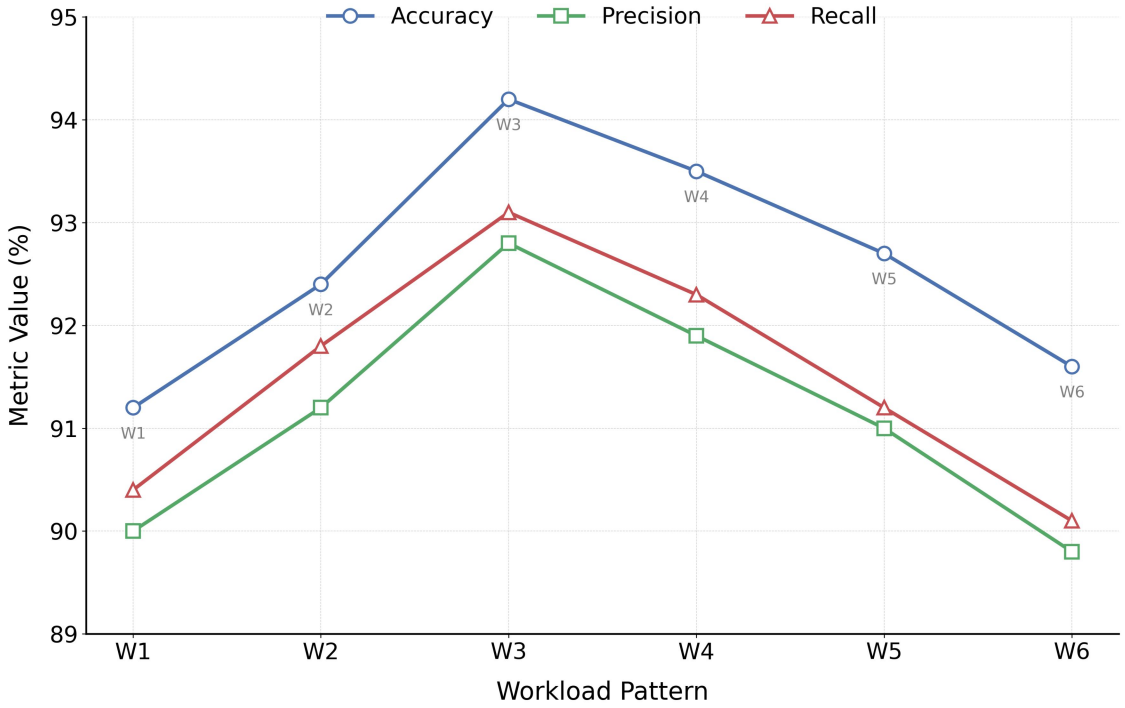

Figure 3. Strategy robustness evaluation under dynamic load mode

As shown in the experimental results of Figure 3, the reinforcement learning policy proposed in this study demonstrates strong robustness under different dynamic workload patterns. During the workload increase from W1 to W3, the accuracy, precision, and recall all show steady improvements, reaching their peaks at W3. The accuracy reaches 94.2 percent, indicating that the policy effectively adapts to resource demand changes under medium to high loads. It is capable of making timely scaling decisions to ensure system performance and stability. This performance demonstrates that the model has good dynamic awareness and control capabilities, allowing it to balance performance metrics with resource efficiency.

As the workload pattern shifts from W3 to W6, the three metrics exhibit a moderate decline. This downward trend reflects that under extreme or sustained high-pressure conditions, the policy must make more complex trade-offs between service latency control and resource cost. In particular, scenarios like W5 and W6 may involve sudden request surges and intensified resource bottlenecks, which challenge the responsiveness and precision of the policy. This results in slight fluctuations in performance.

Although performance decreases slightly under certain high-load patterns, the overall fluctuation remains limited. The model consistently maintains high performance levels. This stability suggests that the proposed method retains strong policy transferability and generalization in practical settings such as load spikes and nonlinear resource stress. By modeling historical states and integrating short-term and long-term

dependencies, the policy can achieve smooth regulation during both load surges and drops. It avoids over-scaling or unnecessary resource waste.

5. Conclusion

This study addresses the challenge of dynamic resource management and elastic control in microservice architectures. It proposes a reinforcement learning-based policy optimization method aimed at achieving efficient and stable service scaling under varying workloads. By constructing a task-aware state space and introducing advantage functions and temporal dependency modeling, the proposed method significantly improves policy convergence and generalization. It also outperforms existing approaches across multiple performance metrics. The method shows strong adaptability and can continuously make appropriate scaling decisions in response to different service topologies and dynamic resource conditions. This effectively enhances overall system performance.

The reinforcement learning mechanism explored in this work provides theoretical support for intelligent resource management in microservice systems. It also demonstrates the advantages of reinforcement learning in handling high-dimensional, time-varying, and multi-objective decision-making problems. Compared with traditional rule-based or prediction-based methods, the proposed method dynamically perceives subtle environmental changes and supports closed-loop feedback optimization. This establishes a solid foundation for building more stable and intelligent service management systems. In particular, under the growing demands of cloud-native infrastructure and multi-tenant hybrid deployment, control strategies with self-learning capabilities are expected to become a key direction in future service scheduling.

In addition, the experimental design of this study focuses on the sensitivity of the policy to hyperparameters, data disturbances, and workload patterns. This provides empirical evidence for future engineering deployment and parameter tuning. The results show that the policy maintains high stability and robustness across different scenarios. This confirms its feasibility and scalability in practical applications. In complex business environments, the method can automatically adjust resource allocation, improve system availability, reduce service latency, and minimize resource waste. These features offer a valuable foundation for intelligent operations and automated system management.

Looking ahead, with the continuous development of technologies such as multi-cloud collaboration, edge computing, and green computing, microservice deployment environments will become more diverse, and service coordination will grow increasingly complex. In this context, it remains an important direction to enhance the policy's ability to balance multiple objectives, support cross-domain transfer and incremental learning, and integrate structural knowledge and graph-based representations. Furthermore, integrating with system-level interfaces on large-scale scheduling platforms to realize end-to-end deployment from training to execution is a key challenge for practical adoption. With continued exploration and refinement, reinforcement learning-based elastic resource control is expected to play a central role in broader real-world applications, driving cloud computing and intelligent operations toward greater efficiency and intelligence.

References

[1] Y. Tang, "Research on Hierarchical Multi-Agent Reinforcement Learning Resource Orchestration for Large-Scale Heterogeneous Distributed Clusters with Dynamic Load Awareness," 2024.

[2] S. Li, "Adaptive Scheduling for Multi-Model Collaborative Distributed Inference under Resource Heterogeneity and Dynamic Workloads," 2024.

[3] Y. Chen, H. Pei, X. Dong et al., "Application of Deep Learning in Back-End Simulation: Challenges and Opportunities," Proceedings of the 2022 27th Asia and South Pacific Design Automation Conference (ASP-DAC), pp. 641-646, 2022.

[4] A. Abdel Khaleq and I. Ra, "Intelligent Microservices Autoscaling Module Using Reinforcement Learning," Cluster Computing, vol. 26, no. 5, pp. 2789-2800, 2023.

[5] M. Xu, C. Song, S. Ilager et al., "CoScal: Multifaceted Scaling of Microservices with Reinforcement Learning," IEEE Transactions on Network and Service Management, vol. 19, no. 4, pp. 3995-4009, 2022.

[6] H. Qiu, W. Mao, C. Wang et al., "AWARE: Automate Workload Autoscaling with Reinforcement Learning in Production Cloud Systems," Proceedings of the 2023 USENIX Annual Technical Conference (USENIX ATC 23), pp. 387-402, 2023.

[7] B. Barua and M. S. Kaiser, "AI-Driven Resource Allocation Framework for Microservices in Hybrid Cloud Platforms," arXiv preprint arXiv:2412.02610, 2024.

[8] J. Huang, "Reliability-Aware Lane Detection for Autonomous Driving in Complex Nighttime Environments," 2024.

[9] C. Meng, S. Song, H. Tong et al., "DeepScaler: Holistic Autoscaling for Microservices Based on Spatiotemporal GNN with Adaptive Graph Learning," Proceedings of the 2023 38th IEEE/ACM International Conference on Automated Software Engineering (ASE), pp. 53-65, 2023.

[10] Q. Fettes, A. Karanth, R. Bunescu et al., "Reclaimer: A Reinforcement Learning Approach to Dynamic Resource Allocation for Cloud Microservices," arXiv preprint arXiv:2304.07941, 2023.

[11] K. Hu, L. Wen, M. Xu et al., "MSARS: A Meta-Learning and Reinforcement Learning Framework for SLO Resource Allocation and Adaptive Scaling for Microservices," Proceedings of the 2024 IEEE International Symposium on Parallel and Distributed Processing with Applications (ISPA), pp. 590-599, 2024.

[12] J. Santos, T. Wauters, B. Volckaert et al., "gym-hpa: Efficient Auto-Scaling via Reinforcement Learning for Complex Microservice-Based Applications in Kubernetes," Proceedings of the NOMS 2023 IEEE/IFIP Network Operations and Management Symposium, pp. 1-9, 2023.

[13] A. Goli, N. Mahmoudi, H. Khazaei et al., "A Holistic Machine Learning-based Autoscaling Approach for Microservice Applications," CLOSER, vol. 1, pp. 190-198, 2021.