Journal of Computer Technology and Software

ISSN: 2998-2383

Vol. 3, No. 5, 2024

Adaptive Container Migration in Cloud-Native Systems via Deep Q-Learning Optimization

Wenxuan Zhu
University of Southern California, Los Angeles, USA
zhuwenxu@usc.edu

Abstract: This paper proposes a reinforcement learning-based container migration optimization method to address the challenges of dynamism, high dimensionality, and multi-objective optimization in cloud-native environments. The migration process is modeled as a Markov Decision Process. A Deep Q-Network is used to learn the policy between system states and actions. A state feature vector is constructed to comprehensively represent resource usage, network latency, and container distribution. This guides the model to generate migration strategies with global optimality. A composite reward function is designed to balance multiple objectives. It considers load balancing, migration cost, and service latency. This ensures the model performs well across all scheduling goals. In the experimental section, a public cloud computing dataset is used to validate the model. The results show superior performance in key metrics such as resource utilization, load balancing, migration efficiency, and service delay. In addition, multiple comparative experiments and parameter sensitivity analyses are conducted. These explore the impact of key hyperparameters, such as learning rate and scheduling frequency, on system performance. The findings further demonstrate the effectiveness and stability of the proposed method in dynamic resource scheduling tasks. Through systematic modeling and policy optimization, this paper provides an adaptive and intelligent solution to the container migration problem. It supports the improvement of resource management and system responsiveness in cloud-native platforms under complex operating conditions.

Keywords: Container scheduling; reinforcement learning; migration optimization; system performance

1. Introduction

In the context of rapid digitalization and intelligent development, cloud computing has emerged as a fundamental enabling technology. It plays a key role in driving the digital transformation of various industries[1,2]. As business demands become more diverse and the need for elastic resource management grows, traditional virtualization technologies have shown limitations in flexibility and resource efficiency. Against this backdrop, the concept of cloud-native computing has emerged and quickly become the mainstream direction for the next-generation cloud architecture. Its core idea is to achieve rapid application development, elastic deployment, and efficient operations through container technology, microservices architecture, and DevOps practices. In a cloud-native environment, containers become the basic unit of resource scheduling. They are lightweight, fast to start, and highly portable. These features significantly enhance system agility and scalability[3].

However, as the number of deployed containers grows rapidly and business workloads change dynamically, efficient container migration becomes a key challenge in cloud-native systems. Container migration affects not only load balancing of computing resources but also service availability and response latency. In high-concurrency and low-latency scenarios, poor migration strategies may lead to resource waste, performance degradation, or even service disruption. In complex environments such as multi-tenancy, public cloud, and edge computing, container migration decisions face high uncertainty and dynamic conditions. This requires more intelligent and adaptive migration strategies[4].

Reinforcement learning, an intelligent algorithm that interacts with the environment to continuously optimize decision-making policies, has shown strong capabilities in solving dynamic decision and optimization problems. Compared with rule-based methods or static optimization models, reinforcement learning can adjust migration strategies in real time based on changes in system state[5]. It enables fine-grained control and intelligent scheduling of resources. When facing multi-dimensional metrics, non-linear system behavior, and complex resource constraints, its deep policy learning and self-evolving ability can better handle the complexity and uncertainty of the migration process. Therefore, applying reinforcement learning to optimize container migration strategies can improve the generalization and adaptability of these strategies. It also offers the potential to provide more intelligent and efficient resource scheduling mechanisms in cloud-native environments[6].

From the perspective of system-level resource management, optimizing container migration involves not only performance improvement of individual nodes or applications but also collaborative optimization of resource allocation at the data center level. In multi-tenant environments, services have diverse resource demands and performance goals. Proper scheduling of container migration tasks can reduce resource fragmentation, improve overall cluster utilization, and avoid hot spots and idle resources[7]. As green computing gains

importance, energy saving has also become a key concern for cloud service providers. Intelligent migration strategies can help reduce energy consumption while ensuring service quality. This enables dual optimization of performance and energy efficiency. Therefore, in practical applications, research on container migration has significant economic and social value.

In summary, under the trend of widespread adoption of cloud-native architectures, optimizing migration strategies in container environments has become a vital direction for ensuring the efficient operation of modern cloud platforms. By leveraging the intelligent decision-making capabilities of reinforcement learning, container migration can gain higher autonomy and flexibility. This helps address dynamic resource changes and diverse workload demands in cloud environments. The research holds great potential and practical significance in building the next generation of resilient, intelligent, and green cloud-native infrastructures.

2. Method

This study aims to optimize container migration in cloud-native environments and proposes a migration decision method based on reinforcement learning, which models the container migration process as a Markov decision process (MDP). The model architecture is shown in Figure 1.

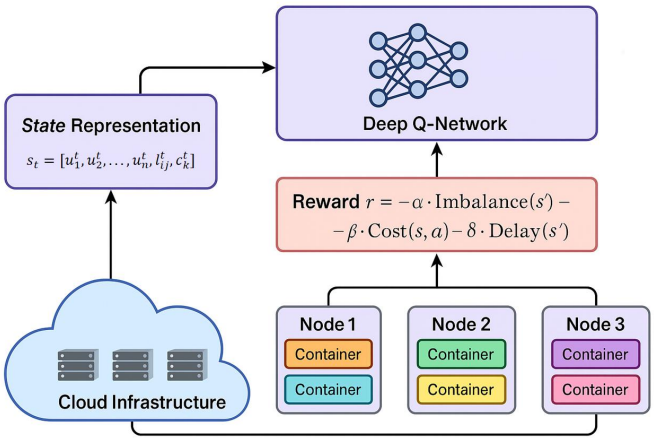

Figure 1. Overall model architecture

The process takes system state, action set, state transition function and reward function as core elements, and defines the following four-tuple (S, A, P, R) . Among them, S represents the state set of the system at any time, including multi-dimensional information such as container load, node resource utilization, network delay, etc.; A is the action set, which represents optional migration decisions, such as migrating a container to a specified target node; $P(s'|s,a)$ represents the probability distribution of transitioning to state C after taking action a in state s; $R(s,a)$ is the immediate reward obtained by performing action a in state s, which is used to guide the direction of strategy optimization.

In terms of state representation, this study constructs a high-dimensional feature vector to represent the system state s_t , namely:

$$s_t = [u_1^t, u_2^t, ..., u_n^t, l_{ij}^t, c_k^t]$$

u_i^t represents the CPU and memory utilization of the i-th node at time t, l_{ij}^t represents the network delay between node i and node j, and c_k^t represents the node where the k-th container is currently located and its resource usage. This state vector can fully reflect the current cluster load distribution and network status, providing accurate environmental information for strategy learning.

In the design of reinforcement learning strategy, Deep Q-Network (DQN) is used as the decision framework to estimate the state-action value function $Q(s,a;\theta)$, where θ is the neural network parameter. The goal is to minimize the following mean square error loss function:

$$L(\theta) = E_{(s,a,r,s')}[(r + \gamma \max_{a'} Q(s', a'; \theta^-) - Q(s,a;\theta))]$$

Where γ is the discount factor and θ^- represents the fixed target network parameters, which are used to improve training stability. By repeatedly sampling interaction experience to update the Q function, the optimal strategy can be approximated.

In order to improve the convergence efficiency and resource perception of the migration strategy, a composite reward function is designed to comprehensively consider multi-objective optimization, including system load balancing, migration overhead and service latency. The specific reward function form is as follows:

$$R(s,a) = -\alpha \cdot \text{Imbalance}(s')$$
$$-\beta \cdot Cost(s,a) - \delta \cdot Delay(s')$$

$\text{Imbalance}(s')$ represents the imbalance of the load of each node in the new state, $Cost(s,a)$ is the system resource consumption generated by executing the migration action, $Delay(s')$ reflects the average response delay of user requests, and α, β, δ is the coefficient for adjusting the weight of each target. Through policy iteration, the agent can gradually optimize the overall performance of the system while maximizing the long-term return.

To further improve training efficiency and avoid the strategy from falling into local optimality, the experience replay mechanism and ε -greedy exploration strategy are introduced. In each iteration, the agent selects a random action with probability ε and selects the action $a^* = \arg\max_a Q(s,a)$ with the highest current valuation with probability $1-\varepsilon$. At the same time, all interaction data

are stored in the replay buffer pool in the form of tuples (s, a, r, s') , and small batches of experience are regularly sampled from it for network training. This mechanism ensures the diversity and stability of strategy updates, thereby more effectively learning container migration strategies with generalization capabilities in complex cloud-native environments.

3. Experiment

3.1 Datasets

This study uses the Google Cluster Data (Google Borg Trace) as the dataset for training and evaluating the container migration optimization model. The dataset was collected from a large-scale data center during real production operations. It includes detailed information such as resource usage across thousands of servers, job scheduling records, and container instance lifecycles. The dataset has high representativeness and complexity, reflecting real-world operating conditions.

It features a short sampling interval, continuous time series, and rich records. These characteristics provide a realistic and complex background for container migration decision-making. Key fields in the dataset include CPU and memory usage, task start and end times, machine status changes, and task priorities. This information can be used to construct the system state space and guide the optimization of migration objectives.

To meet the modeling requirements of reinforcement learning, the raw log data must be preprocessed. This includes data cleaning, normalization, and state vector construction. The goal is to generate training samples with temporal continuity and characteristics of resource scheduling.

The main reason for selecting this dataset lies in its large scale, realistic scenarios, and complex structure. It can effectively simulate resource dynamics and container scheduling behavior in a cloud-native environment. With this dataset, the adaptability of reinforcement learning strategies to complex and dynamic resource conditions can be validated in a non-synthetic setting. It provides a solid foundation and data support for deploying the model on real cloud platforms.

3.2 Experimental Results

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

Table 1: Comparative experimental results

Method	CPU Utilization	Load Imbalance	Migration Cost	Avg. Latency
Ours	87.4	0.031	12.5	84.6
Heuristic Threshold Policy[8]	79.2	0.087	25.8	113.2
Randomized Placement[9]	76.5	0.105	21.4	126.7
SGX-aware container[10]	82.6	0.066	18.2	102.3
COSCO[11]	85.1	0.049	15.6	95.4

The experimental results show that the proposed reinforcement learning-based container migration method achieves the best performance in resource utilization. It reaches a CPU utilization rate of 87.4 percent, which is significantly higher than other comparison algorithms. This result indicates that by introducing deep reinforcement learning strategies, the model can dynamically perceive system states and make more informed migration decisions. This leads to efficient scheduling and allocation of resources. In contrast, traditional heuristic strategies and random migration methods lack modeling of global states and long-term objectives. As a result, they show clear limitations in resource consolidation.

In terms of load balancing, the proposed method achieves a load imbalance degree of only 0.031, which is much lower than other baseline models. This demonstrates that the reinforcement learning model can effectively learn the distribution characteristics of resources across nodes. It can proactively adjust container placement to reduce the pressure on hotspot nodes and achieve more balanced load distribution. In comparison, although the SGX strategy performs well under static conditions, it lacks real-time adaptability under dynamic load changes. This results in unstable balancing performance.

Regarding migration cost, the proposed method maintains low overhead, with a migration cost of only 12.5. This is more than 50 percent lower than that of heuristic methods. This improvement is due to the reinforcement learning model incorporating both resource load and migration cost into the reward function. Through policy iteration, it balances migration overhead with scheduling performance. Other methods often overlook the network and system burden introduced by migration, which leads to unnecessary resource waste.

In terms of service response latency, the proposed method also shows strong advantages. The average response time is 84.6 milliseconds, which outperforms all comparison models. This suggests that the reinforcement learning strategy can optimize both system performance and service quality during migration scheduling. It ensures high resource utilization while significantly reducing user request latency. Overall, the method demonstrates strong global adaptability and intelligent decision-making in dynamic, multi-objective container migration scenarios. It effectively meets the real-world demands of elastic scheduling and efficient operations in cloud-native environments.

Next, the hyperparameter sensitivity experiment of the learning rate is given, as shown in Table 2.

Table 2: Learning rate hyperparameter sensitivity experimental results

LR	CPU Utilization	Load Imbalance	Migration Cost	Avg. Latency
0.004	82.9	0.062	19.3	104.7
0.003	85.0	0.048	15.8	95.8
0.002	86.2	0.037	13.9	89.3
0.001	87.4	0.031	12.5	84.6

The experimental results show that the learning rate has a significant impact on the performance of the reinforcement learning model in container migration tasks. As the learning

rate gradually decreases from 0.004 to 0.001, the overall performance of the model steadily improves. This improvement is particularly evident in CPU utilization and load balancing. The results indicate that a lower learning rate helps the model update its policy more stably. It avoids fast convergence that may lead to policy oscillation or local optima, thereby achieving better resource allocation.

For the load balancing metric, the system reaches its best performance when the learning rate is set to 0.001. The imbalance degree drops to 0.031, the lowest among all tested settings. Under this configuration, the reinforcement learning model captures the resource differences across nodes more effectively. Through refined action selection, it enables reasonable container migration. This reduces pressure on hotspot nodes and enhances the coordination of system-wide scheduling.

Migration cost and service latency also decrease as the learning rate becomes lower. At a learning rate of 0.001, the model shows more stable behavior in avoiding unnecessary migrations. The migration cost is only 12.5, and the average response time drops to 84.6 milliseconds. This shows that an appropriate learning rate can balance exploration and exploitation. It allows the model to maintain system efficiency while avoiding frequent or redundant migration actions. As a result, it optimizes user experience while ensuring performance.

Overall, the learning rate is a critical hyperparameter in reinforcement learning training. Its selection directly affects the effectiveness of the final migration policy and the system's performance. The experimental results confirm that, with proper tuning, the model can fully leverage the strengths of reinforcement learning in dynamic scheduling and multi-objective optimization. This further demonstrates its potential for adaptive migration optimization in complex cloud-native environments.

Finally, this paper gives an analysis of the impact of different scheduling frequencies on system performance, and the experimental results are shown in Figure 2.

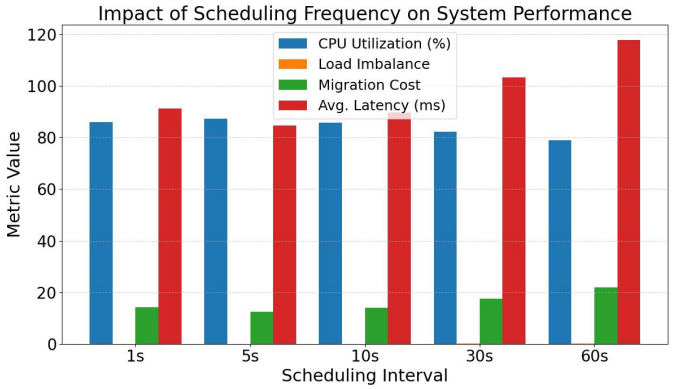

Figure 2. Analysis of the impact of different scheduling frequencies on system performance

The experimental results in the figure show that scheduling frequency has a significant impact on the overall performance of container migration strategies. Shorter scheduling intervals, such as 1 second and 5 seconds, help the system respond more promptly to changes in resource status. This enables finer-grained migration decisions. As a result, higher CPU utilization and better load balancing are observed. At a 5-second scheduling frequency, the system achieves high CPU utilization and the lowest imbalance degree. This indicates that the reinforcement learning strategy can quickly adapt to dynamic workloads and optimize resource distribution within short cycles.

However, as the scheduling frequency decreases to 30 seconds and 60 seconds, the system performance drops noticeably. This is especially evident in migration cost and service latency. The delay in executing scheduling actions prevents the system from responding quickly to load changes across nodes. Some nodes remain overloaded, while underutilized resources are not released in time. This leads to reduced scheduling efficiency and longer user request response times. These results further suggest that lagging scheduling weakens the advantage of reinforcement learning in dynamic environments.

For migration cost, the results show that moderate frequencies, such as 5 seconds and 10 seconds, result in relatively low overhead. This indicates that the model can balance response speed and migration stability within this range. A very high frequency may improve agility but can lead to excessive and unnecessary migrations, increasing system overhead. Conversely, a very low frequency may cause load buildup, indirectly increasing migration pressure. This trend shows that scheduling frequency is a key factor in balancing policy efficiency and resource consumption.

Overall, the experiment confirms that scheduling frequency plays a decisive role in the performance of container migration strategies. A well-chosen scheduling interval can significantly improve system responsiveness, resource utilization, and service quality. In a reinforcement learning-based migration optimization framework, scheduling frequency should be treated as a critical control parameter. It needs to be dynamically adjusted based on system scale, load variability, and application characteristics to fully realize the potential of intelligent migration strategies.

4. Conclusion

This study addresses the key challenges of container migration in cloud-native environments and proposes an intelligent optimization method based on reinforcement learning. It aims to handle dynamic resource changes, workload fluctuations, and multi-objective scheduling demands. By formulating the problem as a Markov Decision Process and applying a Deep Q-Network to learn the state-action mapping, the model achieves adaptive policy adjustment and long-term performance optimization. Experimental results show that the proposed method outperforms classical strategies in improving resource utilization, reducing load imbalance, controlling migration overhead, and optimizing service latency. This confirms its adaptability and effectiveness in complex real-world environments.

The study highlights that in cloud-native architectures, containers serve as the core unit for resource scheduling and service deployment. The intelligence of migration decisions

directly affects the overall system efficiency and stability. Introducing reinforcement learning brings dynamic awareness and policy evolution to traditional scheduling mechanisms. This allows the system to learn from historical experiences and continuously refine its scheduling behavior. The approach demonstrates strong scalability and migration value in scenarios such as multi-tenancy, edge computing, and elastic scaling. It provides both theoretical support and a practical path for building future-oriented, efficient resource management platforms.

In addition, the study conducts a systematic analysis of key factors including scheduling frequency, reward function design, and state space construction. It reveals their deep impact on overall system performance. The model shows good convergence and generalization ability. It also offers a highly modular and tunable solution for multi-objective container scheduling. By deploying intelligent migration strategies, cloud platforms can respond more accurately to resource supply and demand changes. This enables comprehensive optimization of green computing, cost control, and service quality. It enhances the intelligent operation of the underlying infrastructure.

Future research may explore deploying and validating the model in real distributed production environments. It is worth investigating the integration of reinforcement learning with emerging intelligent technologies such as federated learning and graph neural networks. This can better meet the needs of distributed and heterogeneous resource management. Further work can also focus on improving the adaptability of container migration strategies under complex network conditions, sudden workload spikes, or cross-cluster scenarios. As cloud-native technologies continue to evolve, the proposed method is expected to contribute to large-scale cloud platforms, edge computing architectures, and intelligent scheduling systems. It can help build the next generation of cloud infrastructure that is more efficient, agile, and intelligent.

References

[1] Ahmad I, AlFailakawi M G, AlMutawa A, et al. Container scheduling techniques: A survey and assessment[J]. Journal of King Saud University-Computer and Information Sciences, 2022, 34(7): 3934-3947.

[2] Menouer T. KCSS: Kubernetes container scheduling strategy[J]. The Journal of Supercomputing, 2021, 77(5): 4267-4293.

[3] Lai W K, Wang Y C, Wei S C. Delay-aware container scheduling in kubernetes[J]. IEEE Internet of Things Journal, 2023, 10(13): 11813-11824.

[4] Rausch T, Rashed A, Dustdar S. Optimized container scheduling for data-intensive serverless edge computing[J]. Future Generation Computer Systems, 2021, 114: 259-271.

[5] Oleghe O. Container placement and migration in edge computing: Concept and scheduling models[J]. IEEE Access, 2021, 9: 68028-68043.

[6] Mao Y, Fu Y, Zheng W, et al. Speculative container scheduling for deep learning applications in a kubernetes cluster[J]. IEEE Systems Journal, 2021, 16(3): 3770-3781.

[7] Jorge-Martinez D, Butt S A, Onyema E M, et al. Artificial intelligence-based Kubernetes container for scheduling nodes of energy composition[J]. International Journal of System Assurance Engineering and Management, 2021: 1-9.

[8] Zhu L, Wu F, Hu Y, et al. A heuristic multi-objective task scheduling framework for container-based clouds via actor-critic reinforcement learning[J]. Neural Computing and Applications, 2023, 35(13): 9687-9710.

[9] Arnau Q, Barrena E, Panadero J, et al. A biased-randomized discrete-event heuristic for coordinated multi-vehicle container transport across interconnected networks[J]. European Journal of Operational Research, 2022, 302(1): 348-362.

[10] Vaucher S, Pires R, Felber P, et al. SGX-aware container orchestration for heterogeneous clusters[C]//2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS). IEEE, 2018: 730-741.

[11] Tuli S, Poojara S R, Srirama S N, et al. COSCO: Container orchestration using co-simulation and gradient based optimization for fog computing environments[J]. IEEE Transactions on Parallel and Distributed Systems, 2021, 33(1): 101-116.