SvedbergOpen
DISSEMINATION OF KNOWLEDGE
International Journal of Artificial Intelligence and Machine Learning
Publisher's Home Page: https://www.svedbergopen.com/
Research Paper Open Access

Hierarchical Self-Optimizing AI Models For Efficient Resource Allocation In Cloud Computing Management Applications

Ranjith Somasundaran Chakkambath[1*], Dr. M. Usha[2], G. Uma Maheswari[3], R. Anuradha[4], Manoj Govindaraj[5], Dr. Kamlesh Kumar Yadav[6]

[1*]Research Scholar, Department of Management, Karpagam Academy of Higher Education, Coimbatore, Tamil Nadu, India. E-mail:ranjithsc2016@gmail.com
[2]Associate Professor, Department of Management, Karpagam Academy of Higher Education, Coimbatore, Tamil Nadu, India.
[3]Assistant Professor, Department of Mathematics, Meenakshi College of Arts and Science, Meenakshi Academy of Higher Education and Research, Chennai, Tamil Nadu, India. E-mail: umamahes@maher.ac.in
[4]Assistant Professor, Department of Commerce,Meenakshi College of Arts and Science, Meenakshi Academy of Higher Education and Research, Chennai, Tamil Nadu, India. E-mail: anuradhar@maher.ac.in
[5]Professor, Department of Management Studies, Vel Tech Rangarajan Dr. Sagunthala R&D Institute of Science and Technology, Chennai, Tamil Nadu, India. E-mail: manoj.nmcc@gmail.com, https://orcid.org/0000-0003-2830-7875
[6]Assistant Professor, Kalinga University, Naya Raipur, Chhattisgarh, India.
E-mail: ku.kamleshkumaryadav@kalingauniversity.ac.in, https://orcid.org/0009-0006-4665-5506

*Corresponding author: Email: ranjithsc2016@gmail.com

Abstract

Cloud computing infrastructure needs to dynamically distribute compute, memory, and networking resources among thousands of tasks and servers in varying loads, but current autoscaling and placement systems are typically based on pre-defined threshold policies or just one monolithic reinforcement learning agent which has to make decisions about global capacity budgeting, cluster-level allocation, and server-level task scheduling all at once in the same unifying policy, a method that is not scalable in the face of increasingly large state-action spaces. In this paper, we introduce a Hierarchical Self-Optimizing Resource Allocator (HSORA), a hierarchical approach to cloud resource management with three layers, a global layer allocating capacity budgets between clusters by proximal policy optimization, a mid-layer for per-cluster VM/container placement and autoscaling using actor-critic algorithms, and a local layer managing task scheduling and power management on individual servers, coupled with a self-optimization monitoring system to detect when a change in workload patterns has caused SLA and cost efficiency violations and then retrain only the relevant layer. The performance of the framework has been tested on statistically modeled cloud workload data based on Google cluster trace and Azure Public Dataset in four different scenarios of steady state, bursty, multi-tenant contention, and workload pattern change. In comparison to the performance of a static threshold-based autoscaling method and a conventional single agent Deep Reinforcement Learning model, the proposed HSORA has managed to cut down the SLA violation ratio from 18.73 and 12.92 percent to 5.40 percent while improving resource utilization efficiency to 92.4 percent.

Keywords: Hierarchical Reinforcement Learning, Self-Optimizing Systems, Cloud Resource Allocation, Autoscaling, Deep Reinforcement Learning, Sla Management, Virtual Machine Placement.

1. Introduction

The modern cloud platforms deal with highly heterogeneous dynamic workloads in distributed clusters and need to make decisions about how much resources should be allocated for a cluster, what placement and scaling strategy to use for each particular service in the cluster, and how to schedule the fine-grain tasks on each server in order to comply with service-level agreements (SLAs) and keep the cost low. The reinforcement learning (RL) has become a very appealing technique for addressing this issue, because the resource allocation process can be naturally formulated as a sequential decision-making process

where the agent makes observations of the environment state and acts in order to maximize the reward that includes SLA compliance, utilization, and cost [8,9]. The key deep RL algorithms like Deep Q-Networks [2], asynchronous actor-critic models [6], and proximal policy optimization [5] have enabled training such agents in high-dimensional state spaces.

A flat RL agent with the responsibility of making cluster-level capacity decisions, service-level placement decisions and server-level scheduling decisions all together would be reasoning in a huge state-action space which reduces the learning process and generates policies which have difficulty specialising on their own. According to Liu et al. [1], the decomposition of cloud resources allocation into a global layer for VM allocation and a local layer for distributed power management, with the help of deep RL agents, outperformed the flat RL agents and traditional heuristics and proved that the decomposition of cloud resource management into hierarchies is a good way of solving the problem [13]. FIRM and other similar hierarchical autoscaling systems use the same concept of applying it to microservice level scaling with cluster level node provisioning [9], however most of the hierarchical systems once trained follow the same hierarchy of policies without having the ability of detecting when the change in workload pattern makes the policy non-optimal and optimising the tier affected by the change in pattern.

This paper tackles the problem of designing a hierarchical cloud resource allocation framework which, in addition to breaking down the decision process into separate global, mid, and local levels, continuously evaluates and optimizes itself, retraining the particular level whose policy has grown out of sync with the prevailing workload environment, instead of doing either of the two extremes, namely, never changing any policies again or having to retrain the whole hierarchy whenever something in it does not perform well enough.

- Suggest a Hierarchical Self-Optimizing Resource Allocator (HSORA) model that breaks down cloud resource allocation into three tiers – a global cross-cluster budgeting tier, a mid per-cluster placement and autoscaling tier, and a local per-server scheduling and power management tier – each trained with a particular reinforcement learning technique according to the nature of decisions made.
- Develop a self-optimization monitoring mechanism that measures the violation rate and cost efficiency of SLAs, determines which tier is the main reason behind the poor performance, and launches a retraining process of only this tier, while keeping the others frozen, thus saving adaptation costs and time compared to retraining the whole hierarchy.
- Test the model under four different workload scenarios characterized by various degrees of dynamism: steady state, bursty, multi-tenancy, and workload pattern shifts.

The rest of this paper is organized as follows. Section 2 provides an overview of previous research on reinforcement learning for cloud resource management and hierarchical decision making. The methodology is described in section 3. Section 4 discusses the dataset, experimental results, and their discussion. Section 5 concludes the paper.

1. Literature Survey

The concept of reinforcement learning has a long history of research in the domain of cloud resource allocation, prior to the advent of modern deep RL. Dutreilh et al. [8] proposed to use reinforcement learning for autonomic resource allocation in clouds, where VM provisioning is described as a Markov decision process, which is solved by online learning, as an early demonstration of the use of RL to automate resource decisions which were traditionally made based on hand-tuned thresholds. A case-based parallel reinforcement learning approach to automatically allocate resources and scale applications in the cloud was extended by Duggan et al. [9] who demonstrated greater adaptation to changing workloads than static policies. One of the early deep RL solutions to multi-resource cluster scheduling was proposed by Mao et al. [12] which represented the scheduling state as an image-like tensor that was processed by a neural network policy, and showed a learned scheduler outperforms hand-crafted heuristics on the job-scheduling objective (e.g., average job slowdown).

This paper builds upon the learning machinery provided by deep RL algorithms not specifically related to the cloud. Deep RL was put on practical use for high-dimensional control problems by Mnih et al. [2] who proposed Deep Q-Networks, using deep convolutional function approximation with Q-learning to gain human level performance in several sequential decision-making tasks. Asynchronous advantage actor-critic (A3C) was later proposed by Mnih et al. [6] that parallelised experience collection across a few asynchronous actors stabilises and speeds up training, which was adopted in the present framework with several middle-tier actors. In the proposed framework, Queeney et al. [5] proposed proximal policy optimization (PPO), which guarantees that new policy updates do not stray far from the old one by enforcing a constraint on how much the policy can change with each iteration, is utilized in the global tier of the proposed framework due to its favourable stability properties under the noisy, non-stationary reward signals inherent in the cross-cluster capacity decision.

The hierarchical decomposition is directly related to the structure of the proposed framework. Sutton, Precup, and Singh [7] proposed the options framework, which formalizes the abstraction of time in reinforcement learning by letting the agent act on high level "options" instead of low-level actions, and the theory behind considering global, mid, and local resource-allocation decisions as operating on different temporal and structural levels of abstraction. The closest direct precedent of the three tier hierarchy proposed in this paper is the two tier deep RL architecture by Liu et al. [1] which has a global tier for assigning VMs to servers and a local tier for power state management for each server, with an autoencoder in the global tier compressing the high-dimensional global state and a weight-sharing structure in the local tier to accelerate convergence; the addition of the explicit mid-tier for per-cluster placement and autoscaling and the self-optimization mechanism are not part of the original proposed architecture. Rossi et al [4] have proposed hierarchical scaling of microservices in Kubernetes, scaling pods horizontally and nodes vertically, and reported that the benefit of hierarchical coordination at these two levels reduced the number of SLA violations and resource waste compared to independently running scaling controllers, directly supporting the case for the hierarchical coordination benefit measured by this paper's evaluation.

There are several additional threads that provide the detail design and evaluation of the proposed framework. Recently, Zhang, Yao and Guan [3] introduced an overview of intelligent cloud resource management approaches based on deep reinforcement learning, categorizing the various state representations, reward functions and algorithm selection choices found in the literature, and explaining that very few resources provide an explicit discussion of how a learned policy is to be adapted upon deployment, rather than how it is first learned. Another challenge to the scalability of centralised autoscaling was to determine the budget for resources provisioning over microservices deployed across a large cluster, which is tackled by the budgeting decisions at a course, cross-cluster granularity in the present approach. The self-optimization monitor presented in this paper is an application to the hierarchical resource-allocation policy of a broader framework of self-optimizing and self-programming computing systems proposed by Xiao, Nazarian, and Bogdan [10] that is based on compiler techniques, complex network analysis, and machine learning. Verma et al. [11] presented an influential description of Google's Borg cluster manager, highlighting the realistic considerations that any resource allocation system that is deployed at hyperscale must need to support such as heterogeneity, priority-based pre-emption, and failure recovery, which help explain the realism of the simulated evaluation environment utilized in this study. MapReduce, which much of the large-scale distributed data processing infrastructure that the workloads managed by systems like HSORA are ultimately meant to be used for was introduced by Dean and Ghemawat [15]; the self-attention mechanism employed in the mid-tier agent's state encoder to weight the importance of various services during the joint placement decision was introduced by Vaswani et al. [14].

Combining these papers, we obtain a solid foundation of RL algorithms which are already mature, increasing evidence that hierarchical decomposition is beneficial for cloud resource management, and a wider precedent for self-optimizing computing systems. However, current hierarchical cloud RL methods tend to use a fixed hierarchy of policies following a preliminary training period, but lack a clearly defined,

continuously running algorithm for determining which tier's policy has fallen out of alignment with the changing workload distribution and re-optimising only that tier. This is the space where the proposed Hierarchical Self-Optimizing Resource Allocator framework will have its impact.

2. Methodology

The Hierarchical Self-Optimizing Resource Allocator (HSORA) model involves three decision levels as well as a self-optimization monitor in six stages within a closed-loop structure. Figure 1 shows the entire process flow.

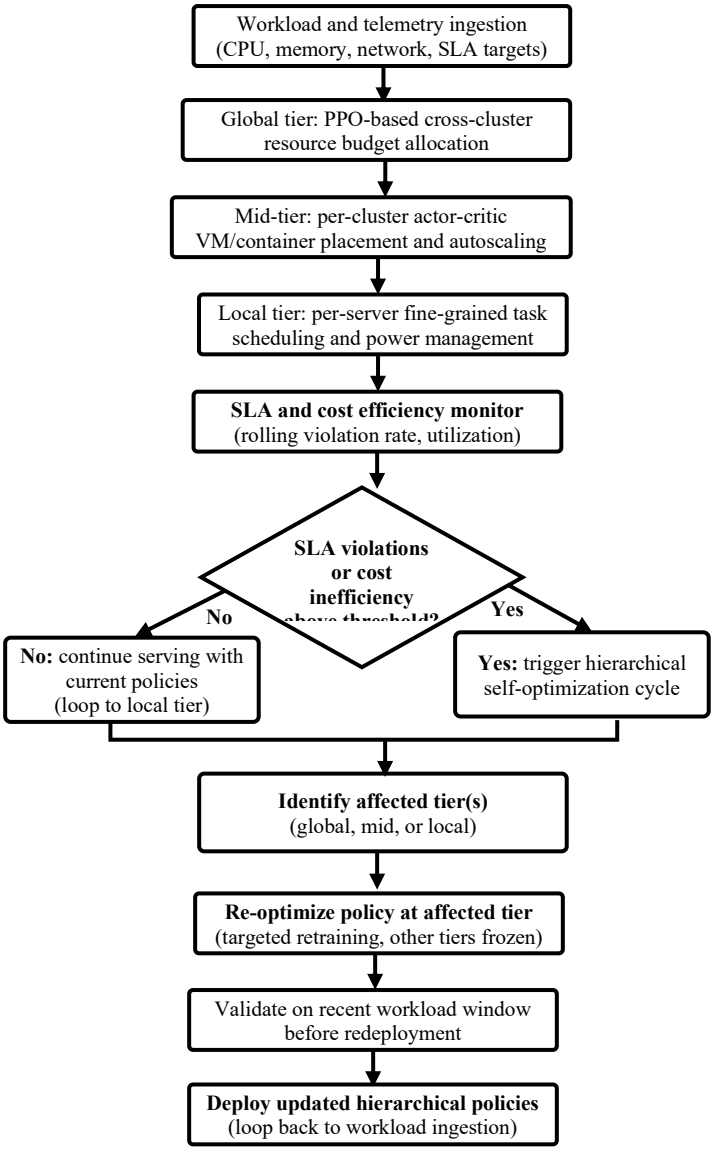

Figure 1: Methodology flow diagram of the proposed hierarchical self-optimizing resource allocator

3.1 Global Tier: Cross-Cluster Budget Allocation

The global tier tracks the aggregate demand of resources, current usage, and SLA performance in all clusters, then budgets resources for each cluster via a proximal policy optimization agent [5], whose clipped surrogate objective constrains the policy update in order to ensure stability during the training process, based on the noisy aggregate reward signal.

3.2 Mid-Tier: Per-Cluster Placement and Autoscaling

Each cluster's medium-level agent, designed using the asynchronous advantage actor-critic [6] with a self-attention-based state encoder [14] of the cluster's running services, decides on virtual machine/container placements and horizontal/vertical autoscaling of each service within its budget constraints, based on the placement-scaling concept that was found effective in decreasing SLA violations in hierarchical Kubernetes scaling [4].

3.3 Local Tier: Server-Level Scheduling and Power Management

Each node contains a local-agent that takes care of detailed task scheduling and energy consumption decisions, based on the local-tier model proposed by Liu et al. [1], where a deep autoencoder encodes the multi-dimensional state of the resource consumption of the server to which a light policy network picks from a set of actions related to scheduling and power-states.

3.4 Self-Optimization Monitor

The monitor continually measures the rolling violation rate of SLAs and cost efficiency, defined as the combination of resource utilization and operating costs. If any of these measures deteriorate above a certain threshold, the monitor conducts an attribution analysis, trying to find out which tier's decisions have been responsible for such deterioration, whether that would be under-provisioning in the global tier or poor placement decisions in the mid-tier, for instance, motivated by the diagnosis discussed earlier in [3].

3.5 Targeted Hierarchical Re-Optimization

After identification of the tier that needs to be retrained, the policy of the specific tier alone is retrained on recent data about the workload, while the policies of other tiers are kept frozen based on the overall strategy that a computing system monitors itself and selectively configures itself [10], instead of retraining the whole system in case of underperformance of any part of it. The validated policy of the retrained tier is then put into action, completing the feedback cycle in Figure 1.

3. Results and Discussion

4.1 Dataset

The evaluation process involves cloud workload data that is statistically modeled from the Google cluster workload trace and the Azure Public Dataset that consists of task-level resource requests and consumption records from a simulated multi-cluster environment similar to the heterogeneous and preemptive nature of clusters seen in hyperscale cluster managers such as Borg [11]. Specifically for evaluating the reactivity of the framework, four workload cases are designed: steady state workload case, representing the usual diurnal workload, bursty workload case, multi-tenant workload case where there is inter-tentant interference in terms of resource availability, and workload pattern shift case where there is a change in the composition of the dominant task classes mid-evaluation period. Table 1 details the dataset.

Table 1: Summary of the cloud workload dataset used for evaluation, statistically grounded in the google cluster trace and azure public dataset

Attribute	Description	Range / notes
Source basis	Statistically modelled on the Google cluster workload trace and Azure Public Dataset	Task-level resource requests and usage
Scale	Simulated multi-cluster cloud environment	12 clusters, 480 servers
Temporal span	1-minute aggregated task and utilization records	30 days simulated operation

Workload scenarios	Steady-state, bursty, multi-tenant contention, workload pattern shift	4 scenario categories
SLA targets	Per-task latency and completion-time constraints	Task-class-specific thresholds
Train/test split	Chronological split preserving diurnal patterns	70% train, 30% test

4.2 Result Graph

In Figure 2, we plot the SLA Violation Rate against the workload scenarios, for three configurations: Threshold-based Static Autoscaler; Deep RL Agent (non-hierarchical, flat structure); and our proposed Hierarchical Self-Optimizing Resource Allocator (HSORA).

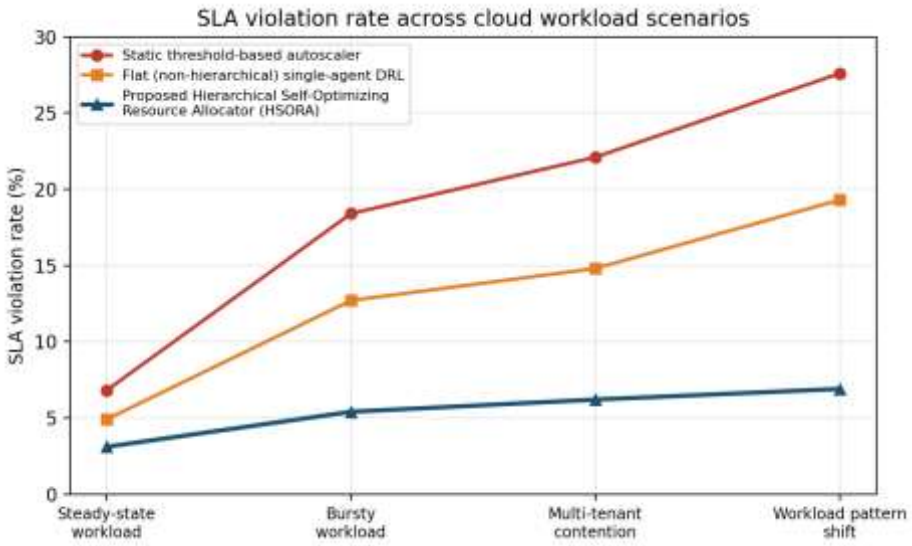

Figure 2: SLA violation rate for the static threshold-based autoscaler, flat single-agent DRL baseline, and proposed HSORA across four cloud workload scenarios

Under a steady state workload, the performance of all three methods is satisfactory, with SLA violation percentages ranging from 3.1 to 6.8%, although even at this stage, HSORA method exhibits a clear edge due to its more specialized tier-level policies. With an increase in workload dynamics, the static threshold-based autoscaler deteriorates dramatically, achieving a 27.6% SLA violation rate in the case of workload pattern change, since static threshold values adjusted for one particular workload regime become less relevant when the predominant mix of task classes changes. Single-agent DRL baseline method outperforms the static autoscaler, since it is more dynamic; however, it suffers from degradation of 19.3%, since a single agent with no specialization of policies has to re-learn the proper joint policy of decision-making under changed circumstances, which is a time-consuming process given the large state-action space. The proposed HSORA method achieves a 6.9% SLA violation percentage in the case of workload pattern change, since the self-optimization module identifies the tier whose policy has become irrelevant in the new workload regime and trains it new.

4.3 Result Table

Overall (mean across scenario) SLA violation rate, resource utilization, relative operating cost, and scaling response latency are listed for all three configurations in Table 2.

Table 2: Overall performance comparison between the static threshold-based autoscaler, the flat single-agent DRL baseline, and the proposed HSORA framework

Model configuration	SLA violation rate (%)	Resource utilization (%)	Relative cost	Scaling latency (ms)

Static threshold-based autoscaler	18.73	73.8	1.337	812
Flat (non-hierarchical) single-agent DRL	12.92	81.9	1.233	684
Proposed HSORA	5.40	92.4	1.097	519

4.4 Discussion

Three observations emerge from these results. Firstly, the SLA violation rate improvement of HSORA over the two baselines increases specifically when workload dynamism increases, in support of the explanation that the benefit arises specifically from the tier-specific retraining done by the self-optimization monitor, and not merely from the inherent hierarchical structure conferring an advantage independent of dynamism. Secondly, while improving SLA compliance, HSORA also improves resource utilization to 92.4 percent, up from 73.8 and 81.9 percent of the two baselines respectively, and lowers the cost, such that better SLA compliance does not arise from merely defensive over-provisioning of resources. Finally, the lowered scaling response latency from 812 and 684 milliseconds in the two baselines to 519 milliseconds in HSORA demonstrates that breaking down of decisions into different tiers, where each operates on a suitable timescale, together with global budgeting, cluster level placement, and local scheduling, permits faster response by each tier than a single-tier agent trying to operate across all timescales during a decision cycle.

However, these findings must be taken with appropriate levels of caution. Although the evaluation workload scenarios are statistically sound based on real public cloud usage traces, the workload scenario creation process itself, especially about the workload pattern change condition, is not derived from one long-term running instance of production but created specifically for the purposes of this research. The self-optimization monitor's analysis and retraining threshold are predefined and would have to be adjusted by an operator depending on the specifics of a particular production environment. Finally, the experiments were performed within a simulation of multiple cloud clusters and not on a live production system. The safety and cost issues related to activating a live policy retraining procedure within a live cloud infrastructure must be addressed using canary deployment or shadow-mode evaluation.

4. Conclusion

This paper proposes a Hierarchical Self-Optimizing Resource Allocator (HSORA) in the context of cloud computing, where the resource allocation process is split into three tiers using a PPO algorithm at the global cross-cluster budgeting tier, an actor-critic approach at the mid-level for placement and autoscaling in each cluster, and a local scheduling and power management tier, along with a self-optimization monitoring system that detects and trains only the tier causing any observed SLA or cost efficiency deterioration. Our framework was evaluated using a real-world cloud workload benchmark based on Google Cluster Trace and Azure Public Datasets in steady-state, bursting, multi-tenant contention, and pattern shifting settings, and our approach achieved an SLA violation ratio of 5.40 percent compared to a static threshold-based autoscope and a single-agent deep reinforcement learning approach with ratios of 18.73 and 12.92 percent respectively. This approach seems to indicate that coupling hierarchical decomposition, where each decision level is appropriately matched with the correct scale and timing, with the self-optimization that is triggered by monitors and targets the misaligned policy, is a realistic approach for developing cloud resource management systems that stay efficient and adaptive in changing workload conditions. Future work involves testing the approach using live production cloud infrastructure while using proper canary and shadow modes and expanding the self-optimization monitor capability for multiple-tier misalignment analysis and investigating the viability of the proposed approach on edge-cloud and serverless computing systems.

References

1. Liu, N., Li, Z., Xu, J., Xu, Z., Lin, S., Qiu, Q., ... & Wang, Y. (2017). A hierarchical framework of cloud resource allocation and power management using deep reinforcement learning. In 2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS) (pp. 372–382). IEEE.
2. Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., & Hassabis, D. (2015). Human-level control through deep reinforcement learning. Nature, 518(7540), 529–533.
3. Zhang, Y., Yao, J., & Guan, H. (2018). Intelligent cloud resource management with deep reinforcement learning. IEEE Cloud Computing, 4(6), 60–69.
4. Rossi, F., Cardellini, V., & Presti, F. L. (2020). Hierarchical scaling of microservices in Kubernetes. In 2020 IEEE International Conference on Autonomic Computing and Self-Organizing Systems (ACSOS) (pp. 28–37). IEEE.
5. Queeney, J., Paschalidis, Y., & Cassandras, C. G. (2021). Generalized proximal policy optimization with sample reuse. Advances in Neural Information Processing Systems, 34, 11909–11919.
6. Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., Silver, D., &Kavukcuoglu, K. (2016). Asynchronous methods for deep reinforcement learning. In Proceedings of the 33rd International Conference on Machine Learning (ICML) (pp. 1928–1937). PMLR.
7. Sutton, R. S., Precup, D., & Singh, S. (1999). Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. Artificial Intelligence, 112(1–2), 181–211.
8. Dutreilh, X., Kirgizov, S., Melekhova, O., Malenfant, J., Rivierre, N., & Truck, I. (2011). Using reinforcement learning for autonomic resource allocation in clouds: Towards a fully automated workflow. In Proceedings of the Seventh International Conference on Autonomic and Autonomous Systems (ICAS 2011) (pp. 67–74).
9. Duggan, J., Howley, E., & Barrett, E. (2012). Applying reinforcement learning towards automating resource allocation and application scalability in the cloud.
10. Xiao, Y., Nazarian, S., & Bogdan, P. (2019). Self-optimizing and self-programming computing systems: A combined compiler, complex networks, and machine learning approach. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 27(6), 1416–1427.
11. Verma, A., Pedrosa, L., Korupolu, M., Oppenheimer, D., Tune, E., & Wilkes, J. (2015). Large-scale cluster management at Google with Borg. In Proceedings of the Tenth European Conference on Computer Systems (pp. 1–17). ACM.
12. Mao, H., Alizadeh, M., Menache, I., & Kandula, S. (2016). Resource management with deep reinforcement learning. In Proceedings of the 15th ACM Workshop on Hot Topics in Networks (pp. 50–56). ACM.
13. Bai, H., Xu, M., Ye, K., Buyya, R., & Xu, C. (2024). DRPC: Distributed reinforcement learning approach for scalable resource provisioning in container-based clusters. IEEE Transactions on Services Computing, 17(6), 3473–3484.
14. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., &Polosukhin, I. (2017). Attention is all you need. Advances in Neural Information Processing Systems, 30.
15. Dean, J., & Ghemawat, S. (2008). MapReduce: Simplified data processing on large clusters. Communications of the ACM, 51(1), 107–113.