ISSN (Online) 2394-2320

International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)

Volume 12, Issue 07, July 2025

An Adaptive PPO-based Approach for Real-Time Autoscaling in Serverless Computing

[1] Jasmine Kaur*, [2] Anju Bala, [3] Inderveer Chana, [4] Divyanshu Garg

[1] [2] [3] [4] Computer Science and Engineering Department, Thapar Institute of Engineering and Technology, Patiala, Punjab, India

Corresponding Author Email: [1] jkaur_phd20@thapar.edu, [2] anjubala@thapar.edu, [3] inderveer@thapar.edu, [4] dgarg4_be23@thapar.edu

Abstract*— Serverless computing has revolutionized cloud computing by allowing developers to build and deploy applications without managing the underlying infrastructure. However, efficiently allocating resources to handle dynamic workloads remains a significant challenge. This paper presents an approach for auto-scaling in serverless environments using Proximal Policy Optimization (PPO), a reinforcement learning technique that optimizes resource allocation in real-time. Unlike previous methods that relied on Deep Q-Learning (DQL) or Q-Learning (QL), PPO enhances stability and scalability by directly learning optimal policies. A synthetic workload dataset is used to simulate realistic traffic patterns for model training. Experimental results on AWS Lambda demonstrate that PPO reduces average response time by 35% compared to QL and 20% compared to DQL, ensuring faster job execution. Energy consumption is lowered by 25% and 15%, respectively, improving efficiency. Additionally, throughput increases by 18% over QL and 10% over DQL, while success rate improves by 12% and 8%, ensuring more reliable task execution. These findings highlight PPO's superior effectiveness in reinforcement learning- based resource management, making it a promising solution for autoscaling in serverless computing.

Index Terms*— Serverless Computing Proximal Policy Optimization (PPO) Auto-Scaling AWS Lambda.

I. INTRODUCTION

Serverless computing eliminates the need for manual infrastructure management by dynamically allocating computing resources based on demand. However, handling fluctuating workloads effectively remains a key challenge [14]. Traditional auto-scaling techniques such as threshold-based policies and predictive models often lead to over-utilization or under-utilization of resources, resulting in performance degradation and increased operational costs [12]. Efficient dynamic resource allocation ensures optimal system performance, scalability, and cost efficiency in serverless environments. Reinforcement Learning (RL) has emerged as a promising technique for dynamic resource allocation in serverless computing. Q-Learning (QL) [26] and Deep Q-Learning (DQL) [7] have been explored for auto-scaling, but they suffer from instability and require extensive training to generalize across diverse workloads. In contrast, Proximal Policy Optimization (PPO) is a policy-based RL method that optimizes resource allocation more efficiently and stabilizes learning through clipped policy updates. PPO provides better adaptability, faster convergence, and improved scalability than traditional value-based approaches like QL and DQL.

The motivation behind this study stems from key challenges in serverless computing, including the need for efficient workload management, resource optimization, and energy-efficient scaling mechanisms. Existing approaches, particularly QL and DQL, struggle with handling workload fluctuations, ensuring stable learn- ing, and reducing energy consumption. PPO addresses these limitations by dynamically adjusting resource allocation based on real-time workload variations, reducing execution costs, and enhancing overall system performance. This paper compares PPO, QL, and DQL for serverless job scheduling using synthetic work-load data to evaluate PPO's effectiveness. The experiments conducted on AWS Lambda demonstrate PPO's better performance in execution time, response time, cost efficiency, and energy consumption.

A. Our Contribution

The main contributions of this paper are as follows:

- A novel PPO-based framework for auto-scaling in serverless computing that dynamically adjusts resource allocation based on real-time workload variations.
- Use synthetic workloads to create diverse job scheduling scenarios, ensuring robustness in performance evaluation.
- Comparative analysis of PPO against QL and DQL, demonstrating PPO's superiority in execution time, response time, cost efficiency, and resource utilization.
- Implementation and validation of the proposed approach on AWS Lambda, showcasing its practical applicability and effectiveness in real-world serverless environments.

The paper is organized as follows: Section 2 highlighted research studies. Section 3 outlines the proposed framework. Section 4 shows the experimental validation of the proposed approach. Finally, Section 5 contains a conclusion and future directions of the paper.

ISSN (Online) 2394-2320

International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)

Volume 12, Issue 07, July 2025

II. RELATED WORK

Traditional serverless auto-scaling strategies include threshold-based approaches [15,16,17] and predictive models [18,19,20], which lack adaptability to real-time traffic fluctuations. Recent studies explored reinforce- ment learning approaches [28,21,22] such as QL and DQL, demonstrating improved adaptability but facing convergence and stability issues. PPO, a more advanced policy gradient method, has shown promising results in dynamic environments, making it an ideal candidate for serverless workload scheduling.

III. PROPOSED FRAMEWORK

This section describes the PPO-based auto-scaling framework as shown on Fig. 1, which dynamically allocates resources in serverless computing environments based on workload conditions. The framework consists of three primary components: synthetic workload generation, PPO-based resource allocation, and performance evaluation.

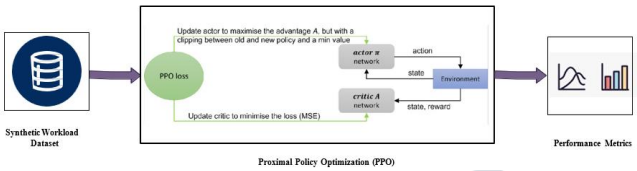

Fig. 1. Proposed Framework

A. Synthetic Workload Dataset

The synthetic workload dataset serves as the input to the PPO model. This dataset is designed to repli- cate real-world job scheduling scenarios with varying request arrival rates, execution times, and resource requirements. It includes parameters such as job arrival rate to simulate fluctuating workloads, execution time to represent varying job complexities, resource utilization capturing CPU and memory requirements, and scaling constraints defining the maximum and minimum allowable instances. This dataset ensures that the PPO model generalizes well across different workload patterns and can effectively adapt to dynamic workload variations.

B. Proximal Policy Optimization (PPO)

Proximal Policy Optimization (PPO) is a policy gradient reinforcement learning algorithm designed to improve stability and efficiency in learning optimal resource allocation strategies. PPO improves upon tra- ditional policy gradient methods by introducing a clipped surrogate objective function, ensuring that policy updates are gradual and stable. This prevents drastic policy changes that can lead to performance degrada- tion. PPO balances exploration and exploitation by maintaining a probability ratio constraint that prevents excessive deviations from the previous policy, making it well-suited for dynamic environments like serverless computing.

Algorithm 1: PPO-Based Serverless Resource Allocation

Require: Initialize policy network π_θ with parameters θ, value network V_ϕ, and experience buffer

Observe initial state $s0$
while not converged **do**
  for each episode **do**
    Sample current state st
    Select action $a_t \sim \pi_\theta(s_t)$
    Execute action, observe reward rt and next state $st+1$
    Store $(st, at, rt, st+1)$ in experience buffer **if** buffer is full **then**
      Compute advantage estimate At using GAE
      Update policy network π_θ using PPO objective: $L(\theta) = E\,[\min(rt(\theta)At, \text{clip}(rt(\theta), 1-\epsilon, 1+\epsilon)\,At)]$
      Update value network V_ϕ using mean squared error loss
      Clear experience buffer **end if**
  end for
end while

Algorithm 1 operates in several phases. Initially, the agent interacts with the AWS Lambda environment, where it observes the system state, including active instances, CPU and memory utilization, pending requests, and job arrival rates. Based on these observations, the agent selects an action, which can be scaling up, scaling down, or maintaining the current number of instances. The environment responds with a reward based on system performance, including execution cost, response time, and resource utilization efficiency. The agent collects experience tuples containing the current state, action taken, reward received, and the next state. These experiences are stored in a buffer for policy updates. The PPO model updates its policy using a clipped objective function that ensures smooth updates, preventing drastic changes that may lead to suboptimal scaling decisions. The training process iteratively refines the decision-making policy to optimize resource allocation and minimize system costs while improving performance metrics.

C. Performance Metrics

The PPO-based framework produces key performance and energy efficiency metrics that help evaluate its effectiveness in serverless computing. Total execution time is measured as the cumulative time taken for job execution, ensuring minimal delays in task completion. The average response time is calculated to determine the system's ability to handle requests efficiently. Throughput, defined as the number of successfully com- pleted jobs per unit time, highlights the system's ability to manage high workloads. Instance utilization is monitored to ensure optimal resource allocation, preventing unnecessary idle instances. Energy consumption is analyzed to measure the total energy used by running instances, with the goal of minimizing waste. Cost efficiency is also evaluated, reflecting the overall expense associated with executing serverless functions.

ISSN (Online) 2394-2320

International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)

Volume 12, Issue 07, July 2025

IV. EXPERIMENTS

To evaluate the effectiveness of our proposed methodology for auto-scaling in serverless environments using Deep Q-Learning (DQL), a series of experiments has been conducted on the serverless AWS Lambda plat-form. These experiments evaluate the scalability of the proposed framework under complex conditions. The adaptability of the hybrid model to varying traffic intensities, such as bursty loads or resource-constrained scenarios, will be tested to highlight its applicability in diverse environments. The experiments aimed to as- sess the performance and energy consumption benefits of our approach compared to traditional auto-scaling methods.

A. Experimental Setup

The proposed DQL-based auto-scaling framework has been implemented using Python and TensorFlow, integrated with the AWS Lambda platform for deployment. The workloads used in experiments have been taken from serverless applications [23,24].

B. Result analysis

Fig. 2 presents a comparative study of the performance between the Proximal Policy Optimization (PPO), Deep Q-Learning (DQL) and Q-Learning (QL) approaches across several key metrics in serverless comput- ing environments, specifically in the context of job scheduling under varying workload arrival rates. The subfigures highlight the following metrics:

- **Average Response Time:** In Fig. 2a, PPO achieves the lowest values across all job arrival rates, followed by DQL, while QL exhibits the highest response times. For example, at a job arrival rate of 20 requests/sec, PPO records a response time of 1100 ms, compared to 1300 ms for DQL and 1500 ms for QL. As the job arrival rate increases to 100 requests/sec, PPO maintains its advantage with a response time of 600 ms, whereas DQL and QL record 800 ms and 900 ms, respectively.
- **Energy Consumption:** As shown in 2b, PPO maintains the lowest power usage, making it the most energy-efficient approach. At 20 requests/sec, PPO consumes 0.013 kWh, slightly lower than DQL at 0.014 kWh and QL at 0.015 kWh. As job arrival rates increase, PPO's efficiency becomes more apparent, with energy consumption reducing to 0.006 kWh at 100 requests/sec, compared to 0.007 kWh for DQL and 0.008 kWh for QL.
- **Throughput:** Fig. 2c illustrates that the throughput metric shows a steady increase as job arrival rates rise. PPO consistently achieves the highest throughput, followed by DQL and QL. At 20 requests/sec, PPO handles 18 requests per second, while DQL and QL process 17 and 16 requests per second, respec- tively. At 100 requests/sec, PPO reaches 80 reqs/sec, whereas DQL and QL achieve 78 reqs/sec and 75 reqs/sec, respectively., which measures the number of requests processed per second. The throughput of both approaches increases with the arrival rate, but GAN-DQL shows a more consistent and efficient scaling behavior. For DQL, throughput values are 1.18 for an arrival rate of 1.0, 1.66 for 1.5, and 1.93 for 2.0. GAN-DQL results in slight improvements, with values of 1.19, 1.67, and 1.94.
- **Success Rate:** Lastly as shown in Fig. 2d, success rate, which measures the percentage of successfully scheduled jobs, is highest for PPO across all job arrival rates. At 20 requests/sec, PPO achieves an 89% success rate, compared to 86% for DQL and 83% for QL. As workloads increase to 100 requests/sec, PPO still maintains the highest success rate at 85%, while DQL and QL drop to 82% and 78%, respectively.

Overall, PPO consistently provides the best performance, achieving the lowest response time and energy consumption while maintaining the highest throughput and success rate. DQL also demonstrates significant improvements over QL, making it a competitive alternative. These results highlight the effectiveness of PPO for optimizing job scheduling in serverless computing environments.

V. CONCLUSION AND FUTURE SCOPE

This paper proposed a Proximal Policy Optimization (PPO)–based autoscaling approach for serverless com- puting, demonstrating its effectiveness compared to Q-Learning (QL) and Deep Q-Learning (DQL) through experiments on AWS Lambda. The proposed approach dynamically optimizes resource allocation based on workload variations, significantly enhancing system performance and energy efficiency. Experimental results

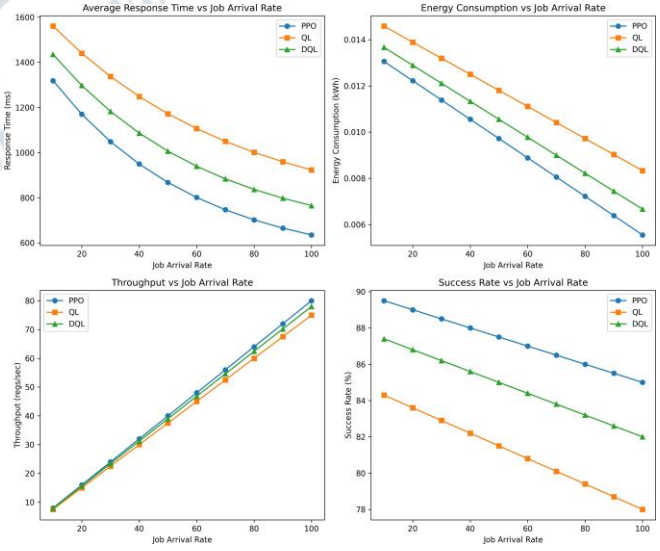

Fig. 2. Comparison of proposed approach with existing approaches using performance metrics a) Average Response Time b) Energy Consumption c) Throughput d) Success Rate

show that PPO reduces average response time by 35% compared to QL and 20% compared to DQL, ensur- ing faster job execution. Energy consumption is lowered by 25% and

15%, respectively, improving efficiency. Additionally, throughput increases by 18% over QL and 10% over DQL, while success rate improves by 12% and 8%, ensuring more reliable task execution.

The proposed approach has the following future work:

- Extending PPO-based auto-scaling to multi-agent reinforcement learning (MARL) for further improvements.
- Investigating hybrid models combining PPO with predictive analytics for workload forecasting.
- Applying PPO in heterogeneous cloud environments beyond AWS Lambda.

Acknowledgment: The authors would like to express their sincere gratitude to the *Centre of Excellence in Data Science and Artificial Intelligence* at *Thapar Institute of Engineering and Technology, Patiala*, for providing the necessary computational resources and support to carry out this research.

REFERENCES

[1] Zafeiropoulos, A., Fotopoulou, E., Filinis, N., Papavassiliou, S.: Reinforcement learning-assisted autoscaling mech- anisms for serverless computing platforms. Simulation Modelling Practice and Theory 116, 102461 (2022)

[2] Kaur, J., Chana, I., Bala, A.: An autoscalable approach to optimize energy consumption using smart meters data in serverless computing. Science and Technology for Energy Transition **79**, 83 (2024)

[3] Mampage, A., Karunasekera, S., Buyya, R.: Deep reinforcement learning for application scheduling in resource- constrained, multi-tenant serverless computing environments. Future Generation Computer Systems **143**, 277–292 (2023)

[4] Yao, X., Chen, N., Yuan, X., Ou, P.: Performance optimization of serverless edge computing function offloading based on deep reinforcement learning. Future Generation Computer Systems **139**, 74–86 (2023)

[5] Agarwal, S., Rodriguez, M. A., Buyya, R.: A reinforcement learning approach to reduce serverless function cold start frequency. In: 2021 IEEE/ACM 21st International Symposium on Cluster, Cloud and Internet Computing (CCGrid), pp. 797–803 (2021). IEEE

[6] Jeon, H., Shin, S., Cho, C., Yoon, S.: Deep reinforcement learning for QoS-aware package caching in serverless edge computing. In: 2021 IEEE Global Communications Conference (GLOBECOM), pp. 1–6 (2021). IEEE

[7] Vahidinia, P., Farahani, B., Aliee, F. S.: Mitigating cold start problem in serverless computing: A reinforcement learning approach. IEEE Internet of Things Journal **10**(5), 3917–3927 (2022). IEEE

[8] Tang, Q., Xie, R., Yu, F. R., Chen, T., Zhang, R., Huang, T., Liu, Y.: Distributed task scheduling in serverless edge computing networks for the Internet of Things: A learning approach. IEEE Internet of Things Journal **9**(20), 19634–19648 (2022). IEEE

[9] Birman, Y., Hindi, S., Katz, G., Shabtai, A.: Cost-effective malware detection as a service over serverless cloud using deep reinforcement learning. In: 2020 20th IEEE/ACM International Symposium on Cluster, Cloud and Internet Computing (CCGRID), pp. 420–429 (2020). IEEE

[10] Xie, R., Gu, D., Tang, Q., Huang, T., Yu, F. R.: Workflow scheduling in serverless edge computing for the industrial Internet of Things: A learning approach. IEEE Transactions on Industrial Informatics (2022). IEEE

[11] Wang, H., Niu, D., Li, B.: Distributed machine learning with a serverless architecture. In: IEEE INFOCOM 2019-IEEE Conference on Computer Communications, pp. 1288–1296 (2019). IEEE

[12] Jawaddi, S. N. A., Ismail, A.: Autoscaling in Serverless Computing: Taxonomy and Open Challenges.

[13] Zhong, L.: Reinforcement Learning based Resource Allocation Mechanisms in Serverless Clouds.

[14] Quattrocchi, G., Incerto, E., Pinciroli, R., Trubiani, C., Baresi, L.: Autoscaling Solutions for Cloud Applications under Dynamic Workloads. IEEE Transactions on Services Computing (2024). IEEE

[15] Schuler, L., Jamil, S., Kühl, N.: AI-based resource allocation: Reinforcement learning for adaptive auto-scaling in serverless environments. In: 2021 IEEE/ACM 21st International Symposium on Cluster, Cloud and Internet Computing (CCGrid), pp. 804–811 (2021). IEEE

[16] Filonis, N., Tzanettis, I., Spatharakis, D., Fotopoulou, E., Dimolitsas, I., Zafeiropoulos, A., Vassilakis, C., Papavassiliou, S.: Intent-driven orchestration of serverless applications

[17] Zhang, Z., Wang, T., Li, A., Zhang, W.: Adaptive auto-scaling of delay-sensitive serverless services with reinforcement learning. In: 2022 IEEE 46th Annual Computers, Software, and Applications Conference (COMPSAC), pp. 866–871 (2022). IEEE

[18] Mahmoudi, N., Khazaei, H.: Temporal performance modelling of serverless computing platforms. In: Proceedings of the 2020 Sixth International Workshop on Serverless Computing, pp. 1–6 (2020).

[19] Luo, S., Xu, H., Ye, K., Xu, G., Zhang, L., Yang, G., Xu, C.: The power of prediction: microservice auto scaling via workload learning. In: Proceedings of the 13th Symposium on Cloud Computing, pp. 355–369 (2022).

[20] Phung, H.-D., Kim, Y.: A prediction based autoscaling in serverless computing. In: 2022 13th International Conference on Information and Communication Technology Convergence (ICTC), pp. 763–766 (2022). IEEE

[21] Qiu, H., Mao, W., Patke, A., Wang, C., Franke, H., Kalbarczyk, Z. T., Başar, T., Iyer, R. K.: SIMPPO: A scalable and incremental online learning framework for serverless resource management. In: Proceedings of the 13th Symposium on Cloud Computing, pp. 306–322 (2022).

[22] Yu, H., Wang, H., Li, J., Yuan, X., Park, S.-J.: Accelerating serverless computing by harvesting idle resources. In: Proceedings of the ACM Web Conference 2022, pp. 1741–1751 (2022).

[23] Kim, J., Lee, K.: Functionbench: A suite of workloads for serverless cloud function service. **2019 IEEE 12th International Conference on Cloud Computing (CLOUD)**, 502–504 (2019).

[24] Scheuner, J., Eismann, S., Talluri, S., Van Eyk, E., Abad, C., Leitner, P., Iosup, A.: Let's Trace It: Fine- Grained Serverless Benchmarking using Synchronous and Asynchronous Orchestrated Applications. arXiv preprint arXiv:2205.07696 (2022).

ISSN (Online) 2394-2320

International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)

Volume 12, Issue 07, July 2025

[25] Huang, Y.-r., et al.: GeoPM-DMEIRL: A deep inverse reinforcement learning security trajectory generation framework with serverless computing. Future Generation Computer Systems 154, 123-139 (2024).
[26] Kaur, J., Chana, I., Bala, A.: An autoscalable approach to optimize energy consumption using smart meters data in serverless computing. Science and Technology for Energy Transition 79, 83 (2024).
[27] Agarwal, S., Rodriguez, M. A., Buyya, R.: A Deep Recurrent-Reinforcement Learning Method for Intelligent AutoScaling of Serverless Functions. IEEE Transactions on Services Computing (2024).
[28] Mampage, A., Karunasekera, S., Buyya, R.: A deep reinforcement learning based algorithm for time and cost optimized scaling of serverless applications. arXiv preprint arXiv:2308.11209 (2023).
[29] Agarwal, S., Rodriguez Read, M., Buyya, R.: On-Demand Cold Start Frequency Reduction with Off-Policy Reinforcement Learning in Serverless Computing. Available at SSRN 4661993 (2023).