Enhancing Autoscaling Efficiency in Heterogeneous Edge Serverless Computing Environments Using real time Reinforcement Learning

Hadjou Ilyas and Young-Woo Kwon
Department of Computer science and Engineering
Kyungpook National University
hadjousslab@knu.ac.kr ywkwon@knu.ac.kr

Abstract— **Edge serverless computing extends the benefits of serverless paradigms to the network edge, offering on-demand resources closer to end-users. However, ensuring efficient resource allocation and scaling on diverse and dynamic edge infrastructures remains a challenge. This work explores the use of reinforcement learning (RL) to dynamically adapt autoscaling thresholds in edge serverless workflows. RL's ability to learn from real-time interactions with the edge environment can enable superior decision-making compared to traditional, predefined autoscaling methods. We propose RL-based autoscaling mechanisms designed specifically for edge environments, aiming to optimize resource utilization and uphold Quality of Service (QoS) guarantees. Our evaluation on real-world and simulated edge scenarios demonstrates the potential of this approach in achieving superior performance.**

Keywords: edge serverless computing, autoscaling, heterogeneity, reinforcement learning, resource allocation

I Introduction

Edge computing has emerged as a transformative paradigm, bringing computation and data storage closer to the data sources [**?**]. This proximity reduces latency and enhances the responsiveness of applications, making it particularly well-suited for real-time and bandwidth-intensive workloads [2]. The integration of serverless computing principles into edge environments, known as edge serverless computing, further amplifies these benefits by enabling on-demand provisioning and scaling of resources [3]. This dynamic scalability is crucial for efficiently handling the fluctuating workloads and diverse resource constraints characteristic of edge infrastructures [4].

However, achieving optimal autoscaling in edge serverless environments poses a significant challenge due to the inherent heterogeneity of edge devices and the unpredictable nature of real-time workloads [5]. Traditional autoscaling methods, often relying on static thresholds or heuristics, may struggle to adapt effectively to the dynamic and diverse conditions prevalent at the edge [6].

To address this challenge, we propose a novel approach that leverages reinforcement learning (RL) to dynamically adapt autoscaling thresholds in edge serverless workflows. Reinforcement learning (RL), with its capability to learn from real-time interactions with the environment, offers the potential to make better scaling decisions compared to traditional methods [7], [8]. By continuously learning and adapting to the evolving state of the edge environment, RL-based autoscaling can optimize resource utilization while upholding stringent Quality of Service (QoS) guarantees [9].

This paper presents a comprehensive exploration of reinforcement learning-based autoscaling mechanisms designed for edge serverless computing. We explore the design of specialized Reinforcement Learning (RL) environments and agents that encompass the distinctive features of edge scenarios. Our approach emphasizes real-time adaptation, enabling the RL agent to dynamically adjust autoscaling thresholds based on continuous feedback from the edge environment.

Through rigorous evaluation of both real-world and simulated edge scenarios, we plan to demonstrate the effectiveness of our proposed approach. Our upcoming findings might underscore the potential of RL-based autoscaling to achieve superior performance in resource efficiency, quality of service (QoS) adherence, and overall system responsiveness.

II Method

We design specialized RL environments and agents to investigate the performance of RL-assisted autoscaling in edge serverless environments. These environments simulate edge-specific characteristics such as resource constraints, network variations, and the proximity of computation to end devices. The RL agent interacts with the environment, adjusting autoscaling thresholds based on real-time feedback.

Each action taken by the RL agent modifies thresholds that govern the edge autoscaler. These actions aim to balance Quality of Service (QoS) guarantees, as potentially outlined in a Service Level Agreement (SLA), with resource efficiency. The agent aims to adapt promptly to workload changes at the edge, ensuring Quality of Service (QoS) and minimizing unnecessary resource consumption.

The reward function is critical in this setup. It penalizes SLA violations and excessive resource usage. This incentivizes the RL agent to find optimal threshold configurations that maintain performance targets while intelligently scaling resources up or down in response to the shifting edge workload, The process of using reinforcement learning to support autoscaling mechanisms is illustrated in Figure 1.

Multi-metric environments in reinforcement learning (RL) research are crucial for understanding the complex interplay of various factors influencing system performance. However, exploring high-dimensional state spaces, especially in edge computing, is computationally demanding. This study

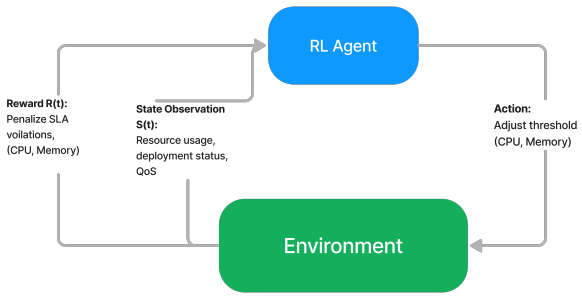

Fig. 1: Reinforcement Learning Mechanism for Autoscaling.

presents a practical approach to this challenge by selectively evaluating subsets of metrics, ensuring the inclusion of CPU usage and CPU threshold in all combinations.

Our motivation stems from the inherent computational limitations of edge computing environments. By systematically testing different combinations, we aim to reveal the relative importance and interactions of performance indicators. This methodological framework offers a principled way to balance computational feasibility with comprehensive analysis, advancing RL-driven optimization for edge computing.

A. RL Environment with Discrete State and Action Space

The state space within our newly designed Reinforcement Learning (RL) environment incorporates serval combinations of metrics. CPU usage is represented as the ratio of current usage to the maximum allowed, while the CPU threshold signifies the point at which autoscaling is triggered. Additionally, we consider the ratio of active serverless function instances to the maximum allowed, as well as the ratio of measured latency to the Service Level Agreement (SLA) defined latency. To ensure computational feasibility, each metric is meticulously discretized into a set of distinct values.

In our state representation, CPU usage, and latency are categorized into seven discrete states, ranging from 0% to 100%, with an additional state for values exceeding 150%. Similarly, latency follows the same seven-state discretization. However, due to the inherent limitations of active instances and CPU thresholds, which cannot exceed 100%, we utilize a five-state discretization for these metrics, ranging from 0% to 100% in 20% increments. Furthermore, our action space comprises three potential actions: decreasing the CPU usage threshold by 10%, maintaining its current value, or increasing it by 10%.

B. Horizontal Pod Autoscaling System

Horizontal Pod Autoscaling is a Kubernetes feature that automatically updates a workload resource, such as a Deployment or StatefulSet, to match demand. This differs from vertical scaling, which assigns more resources to existing Pods. If the load decreases and the number of Pods is above the configured minimum, the HorizontalPodAutoscaler instructs the workload resource to scale back down. The HPA controller running within the Kubernetes control plane, periodically adjusts the desired scale of its target (for example, a Deployment) to match observed metrics such as average CPU utilization, average memory utilization, or any other custom metric you specify, operates on the ratio between desired metric value and current metric value: 1.

$$desiredReplicas = [currentReplicas \times \left(\frac{currentMetricValue}{desiredMetricValue}\right)] \quad (1)$$

III Conclusion and Future Works

In this paper, we have presented a novel approach to address the challenges of autoscaling in edge serverless computing environments. We have proposed the utilization of reinforcement learning (RL) to dynamically adapt autoscaling thresholds, aiming to optimize resource utilization and uphold Quality of Service (QoS) guarantees. Our approach involves designing specialized reinforcement learning (RL) environments and agents that capture the unique characteristics of edge scenarios, such as resource constraints, network fluctuations, and proximity to end devices.

In our future work, we plan to conduct a comprehensive evaluation of our proposed reinforcement learning (RL)-based autoscaling mechanisms in both real-world and simulated edge scenarios. We will evaluate the performance of our approach using the Knative platform, a widely adopted open-source framework for serverless applications. Our evaluation will focus on key metrics such as resource efficiency, Quality of Service (QoS) adherence, and overall system responsiveness. We anticipate that our findings will demonstrate the effectiveness of reinforcement learning (RL)-based autoscaling in achieving superior performance compared to traditional methods in the dynamic and demanding edge computing landscape.

References

[1] Shi, W., Cao, J., Zhang, Q., Li, Y., & Xu, L. (2016). Edge Computing: Vision and Challenges. IEEE Internet of Things Journal, 3(5), 637-646. (CCGrid) (pp. 804-811). IEEE.
[2] Abbas, N., Zhang, Y., Taherkordi, A., & Skeie, T. (2018). Mobile Edge Computing: A Survey. IEEE Internet of Things Journal, 5(1), 450-465.
[3] Baldini, G., Castro, P., Lange, K., & Bhatotia, P. (2017). Serverless Computing: Current Trends and Open Problems. In Research Advances in Cloud Computing (pp. 1-20). Springer.
[4] Roberts, M. (2018). What is serverless? Communications of the ACM, 61(2), 50-57.
[5] Thivagar, M. L., & Chandrasekaran, R. (2023). Serverless Edge Computing: A Comprehensive Survey. ACM Computing Surveys (CSUR), 56(1), 1-37.
[6] Lorido-Botrán, T., Miguel-Alonso, J., & Lozano, J. A. (2014). A review of auto-scaling techniques for elastic applications in cloud environments. Journal of Grid Computing, 12, 559-592.
[7] Sutton, R. S., & Barto, A. G. (2018). Reinforcement Learning: An Introduction. MIT press.
[8] Mao, H., Netravali, R., Alizadeh, M., & Menache, I. (2016). Resource Management with Deep Reinforcement Learning. In Proceedings of the 15th ACM Workshop on Hot Topics in Networks (pp. 50-56).
[9] Wang, Y., Sheng, M., Wang, X., Li, L., & Wang, J. (2020). Learning-Based Autoscaling for Containerized Applications: A Survey. ACM Computing Surveys (CSUR), 53(1), 1-37.