# A Survey on Reinforcement Learning Approaches for Serverless Computing Auto-scaling

Minh Ngoc Tran, Young Han Kim*
Soongsil University

mipearlska1307@dcn.ssu.ac.kr, younghak@ssu.ac.kr*

Abstract

Efficient auto-scaling is a vital requirement for serverless computing to to optimize service performance and resource provisioning. Recently, by utilizing reinforcement learning technique's benefits, many enhanced serverless auto-scaling approaches were proposed to improve the performance of threshold-based auto-scaling mechanisms in current serverless platform. This paper provides a short comprehensive review of current existing solutions, categorizes, analyzes their characteristics, and discusses their limitations.

## Ⅰ. INTRODUCTION

Serverless computing is the emerging paradigm for deploying applications at the cloud and edge environment. This paradigm enables users to focus on application development, while the serverless platform automatically handles service deployment, service scheduling, resource provisioning, etc. Auto-scaling is one of these automated service of serverless computiung. In current open-source and commercial serverless platforms, auto-scaling is controlled by using threshold-based mechanisms. When a specific resource consumption of all service instances reaches the pre-configured threshold, new instances are deployed or deleted. However, these static mechanisms are not the optimal solutions when handling dynamic service demands such as real time traffic adaptation, resource contention, or service-level-objectives (SLO) requirements, etc.

Recently, several works has applied Reinforcement Learning (RL) and Deep Reinforcement Learning (DRL) to optimize the auto-scaling performance of serverless computing. RL is known for its great capabilities of solving decision-making problems that have dynamic requirements. Hence, it is a suitable method for finding dynamic auto-scaling solution in complex serverless computing environment. To the best of our knowledge, there is lack of studies that put together and analyze existing RL auto-scaling approaches for serverless computing. The work [1] only listed and described several existing RL works without categorization or analysis.

This paper provides a short comprehensive review of current state-of-the-art research on RL auto-scaling for serverless computing. We categorizes and analyzes them based on types and characteristics. The current open issues and challenges of these works are also discussed.

## Ⅱ. REINFORCEMENT LEARNING SERVERLESS AUTO-SCALING APPROACHES TAXONOMY

Current RL serverless auto-scaling works have different optimization targets. In this section, we catogorize and discuss them based on 4 different aspects: Scaling methods (how the instances are scaled), Serverless sytem types (the solution targets which kind of serverless system), Concurrent services consideration (the solution optimizes for separate or multiple concurrent services), and Timing (when the auto-scaling decision is applied). Figure 1 shows the different aspects of related works that we discuss.

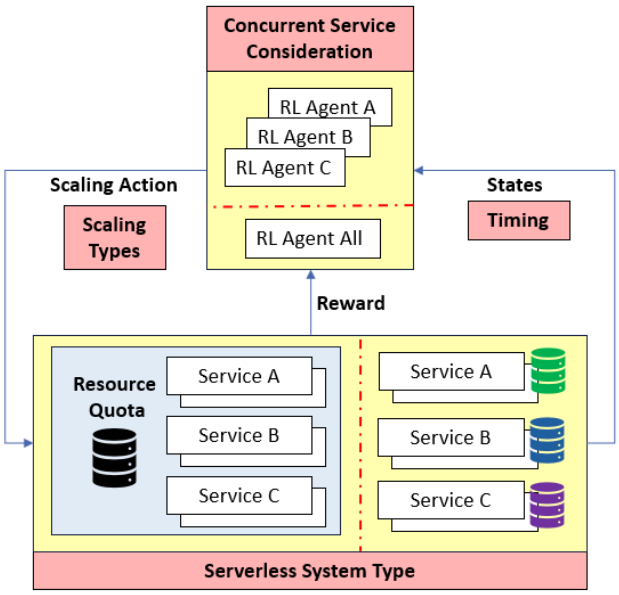


Figure 1. Different categories of Reinforcement Learning serverless auto-scaling

### A. SCALING METHODS

There are 2 types of scaling methods being used in existing works: Horizontal scaling and Hybrid scaling.

- Horizontal Scaling: There are 2 types of action sets for RL models used in related works of this category. The first type is modifying the number of instances. The sescond type is modifying the resource threshold of the serverless platform's default horizontal auto-scaler configuration. For the first type, in [2] and [3], the RL model analyzes current running services' request queue characteristics and system available resource to choose the appropriate number of instances to scale. In [4] and [5], the RL agent prepares the optimal number of service instances in advance to avoid the cold-start problem of serverless services. For the latter type, the Knative service's request concurrency scaling threshold is the target action set in [6]. The Kubernets horizontal auto-scaling's CPU and memory thresholds are the target action sets in [7], [8]
- Hybrid Scaling: RL agent's actions of this category simutaneously change both number of instances (horizontal scaling) and each instance's assigned resource (vertical scaling). Only 3 works [9], [10], [11] belongs to this category. The horizontal actions are similar to horizontal scaling only's work. The vertical actions are changing the container resource limitation in [9], [10] and function's memory size in [11].

### B. SERVERLESS SYSTEM TYPES

Based on the solution's optimization goal, current RL serverless auto-scaling works can be applied to 2 different kinds of serverless systems: Resource consumption-based and Resource quota-based.

- Resource consumption-based: In this type, the serverless system provides resource amount equal to how much the serverless services need to consume. Related works of this category do not have any upper total resource constraint or limitation defined in the RL agent's environment. Generally, these solutions target pay-as-you-go serverless users, optimize services' resource consumption and performance. Most of the current RL auto-scaling works belong to this category ([4]-[12]).
- Resource quota-based: In this type, the total available resource that serverless services can consume is limited by a quota. This quota can be the whole system resource (e.g. limited resource in edge environment) or an assigned part of the total resource. For this serverless environment, the RL agent need to analyze the remaning resource in each environment state to decide the optimal scaling action that does not overuse the resource quota. Only 2 works ([2] and [3]) of the same author belongs to this category.

### C. CONCURRENT SERVICES CONSIDERATION

The major of existing works optimizes auto-scaling performance for each separate service. A separate RL agent is required for each service. By optimizing each one, the aggregated performance of all services can also be optimized. Generally, these works target the resource consumption-based serverless system where there are no total system resource limitation. Hence, although all services are concurrently running inside the same system, each service's RL agent is assumed to have little impact on the others.

Meanwhile, few works consider the auto-scaling performance optimization problem of multiple concurrent services. In [9], the authors point out the performance degradation problem when different separate RL agents of each service run simultaneously. They propose a solution to integrated the reward of all concurrent RL agent to optimize the aggregate auto-scaling performance of the whole system. In [2] and [3], the authors consider the limited total resource of serverless computing system at the edge.. Hence, the RL agent need to decide a suitable action that can guarantee enough resource for all concurrent services.

### D. TIMING

There are 2 auto-scaling timing categories: reactive and proactive. Reactive auto-scaling methods decide and execute scaling action based on current monitored states of the environment. Meanwhile, proactive auto-scaling methods predict the future states of the environment and pre-execute scaling action before the future states actually happen.

Currently all of existing RL serverless auto-scaling works are reactive auto-scaling solutions.

### E. CATEGORIZATION SUMMARY

Table 1 presents how all current existing RL serverless auto-scaling works belongs to different categories that are discussed above. Note that all works belong to reactive scaling timing category. Hence, Table 1 does not include the Timing category aspect.

| Work | Scaling Method | System Type | Concurrent Service |
|---|---|---|---|
| [2] | Horizontal | Quota | Multiple |
| [3] | Horizontal | Quota | Multiple |
| [4] | Horizontal | Consumption | Separate |
| [5] | Horizontal | Consumption | Separate |
| [6] | Horizontal | Consumption | Separate |
| [7] | Horizontal | Consumption | Separate |
| [8] | Horizontal | Consumption | Separate |
| [9] | Hybrid | Consumption | Multiple |
| [10] | Hybrid | Consumption | Separate |
| [11] | Hybrid | Consumption | Separate |
| [12] | Horizontal | Consumption | Separate |

Table 1. Reinforcement Learning serverless auto-scaling works categorization

## III. CURRENT RESEARCH GAPS AND DIRECTION

In this section, we diecuss several current remaining challenges of current existing works and possible future directions.

- Proactive RL auto-scaling: All current works are reactive auto-scaling methods, which execute scaling actions only when new demands are detected. During new scaled instances initializing, the whole system performance will be affected until new instances are ready. Proactive auto-scaling has been proposed and tested in several normal machine learning auto-scaling methods and significantly improve auto-scaling performance against reactive methods. Hence, proactive methods should also be applied to RL solutions.
- Vertical Scaling Instances Restart: This is general problem of vertical scaling in hybrid scaling methods. A method that avoid service instances restart or redeployment when changing their assigned resource is required to further improve performance of hybrid scaling approaches. It can help to avoid additional resource usage while scaling new assigned resource instances. Additionally, service interruption can also be avoided. Recently, Kubernetes has announced this feature in the alpha test of version 1.27 [13], which can solve the mentioned problem. It is suggested to apply this feature In the future works of serverless computing auto-scaling.
- Scalability: This is a consideration for RL methods that consider multiple concurrent services. The simpler implemetation method is using a single RL agent that handle the environment consisting of all services ([2], [3]). However, the RL model is required to be re-trained if the total number of services changes. The more services the system has, the more complex the environment is. It can cause long training and convergence time. Meanwhile, if there are multiple agents for each separte service and a reward aggregation function is applied as proposed in [9], many RL models are required to be trained for each service. The complexity of the reward aggregation function should also be evaluated.

## ACKNOWLEDGMENT

This work was partly supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grants funded by the Korea government (MSIT) (No. 2022-0-01015, 6GRC and No. 2020-0-00946, Development of Fast and Automatic Service recovery and Transition software in Hybrid Cloud Environment).

## REFERENCES

[1] A. Y. Majid, E. Marin, “A Review of Deep Reinforcement Learning in Serverless Computing: Function Scheduling and Resource Auto-Scaling", arXiv Library, arXiv:2311.12839, Oct, 2023.

[2] M. Bensalem, E. Ipek, A. Jukan, “Scaling Serverless Functions in Edge Networks: A Reinforcement Learning Approach”, arXiv Library, arXiv:2305.13130, May, 2023

[3] M. Bensalem, F. Carpio, A. Jukan, “Towards Optimal Serverless Function Scaling in Edge Computing Network”, arXiv Library, arXiv:2305.13896, May, 2023.

[4] S. Agarwal, M. A. Rodriguez, R. Buyya, “A Reinforcement Learning Approach to Reduce Serverless Function Cold Start Frequency”, 2021 IEEE/ACM 21st International Symposium on Cluster, Cloud and Internet Computing (CCGrid), May, 2021.

[5] S. Agarwal, M. A. Rodriguez, R. Buyya, “A Deep Recurrent-Reinforcement Learning Method for Intelligent AutoScaling of Serverless Functions”, arXiv Library, arXiv:2308.05937, Aug, 2023.

[6] L. Schuler, S. Jamil, N. Kühl, “AI-based Resource Allocation: Reinforcement Learning for Adaptive Auto-scaling in Serverless Environments”, 2021 IEEE/ACM 21st International Symposium on Cluster, Cloud and Internet Computing (CCGrid), May, 2021.

[7] P. Benedetti, M. Femminella, G. Reali, K. Steenhaut, “Reinforcement Learning Applicability for Resource-Based Auto-scaling in Serverless Edge Applications”, 2022 IEEE International Conference on Pervasive Computing and Communications Workshops and other Affiliated Events (PerCom Workshops), March, 2022.

[8] A. Zafeiropoulos, E. Fotopoulou, N. Filinis, S. Papavassiliou, “Reinforcement learning-assisted autoscaling mechanisms for serverless computing platforms”, Simulation Modelling Practice and Theory, Volume 116, April, 2022.

[9] H. Qiu et. al, “SIMPPO: a scalable and incremental online learning framework for serverless resource management”, SoCC '22: Proceedings of the 13th Symposium on Cloud Computing, Nov, 2022.

[10] Z. Zhang, T. Wang, A. Li, W. Zhang, “Adaptive Auto-Scaling of Delay-Sensitive Serverless Services with Reinforcement Learning”, 2022 IEEE 46th Annual Computers, Software, and Applications Conference (COMPSAC), June, 2022.

[11] H. Wang, D. Niu, B. Li, “Distributed Machine Learning with a Serverless Architecture”, IEEE INFOCOM 2019 - IEEE Conference on Computer Communications, Apr, 2019.

[12] S. Agarwal, M. A. Rodriguez, R. Buyya, “On-Demand Cold Start Frequency Reduction with Off-Policy Reinforcement Learning in Serverless Computing”, Pre-print, SSRN, https://ssrn.com/abstract=4661993, Dec, 2023.

[13] Kubernetes, “Kubernetes 1.27: In-place Resource Resize for Kubernetes Pods (alpha)”, Available Online at https://kubernetes.io/blog/2023/05/12/in-place-pod-resize-alpha/, Last Accessed on Dec, 2023.