SLO-Aware and Cost-Efficient Predictive Autoscaling for Kubernetes Microservices

SCA 2026 Supercomputing Asia — Gathering the Best of HPC in Asia
HPC Asia 2026

Hyunsu Jeong
Dept. of Computer Engineering
Jeju National University

Geonho Kim
Dept. of Computer Engineering
Jeju National University

Joon-Min Gil*
Dept. of Computer Engineering
Jeju National University

I. Introduction

Microservice-based applications are increasingly deployed on Kubernetes due to their flexibility and scalability. However, maintaining end-to-end latency within strict Service Level Objectives (SLOs) remains challenging, especially under bursty workloads and complex service dependencies. Existing approaches suffer from several fundamental limitations.

- The default Horizontal Pod Autoscaler (HPA) decides whether to scale only based on current resource usage, such as CPU or memory [1].
- Under sudden load spikes, HPA triggers scale-out after utilization increases, causing cold starts, temporary under-provisioning, and frequent SLO violations [2].
- Existing approaches either focus purely on resource metrics or require non-trivial changes to the Kubernetes control plane.

II. Overview of Proposed Method

We propose an autoscaling method that combines:

1. LSTM-based prediction of future CPU utilization
2. Latency-driven penalty based on p95 response latency and SLO threshold

- For each target microservice, an LSTM model predicts the next step CPU utilization
- The system monitors current p95 latency and compares it with an SLO latency target
- A latency aware weight increases when p95 latency exceeds the SLO target
- The composite load is defined as predicted CPU multiplied by this latency aware weight
- This composite metric is exported through KEDA as an external metric
- Scaling is performed based on the composite load metric value presented above, rather than the CPU utilization used by HPA.

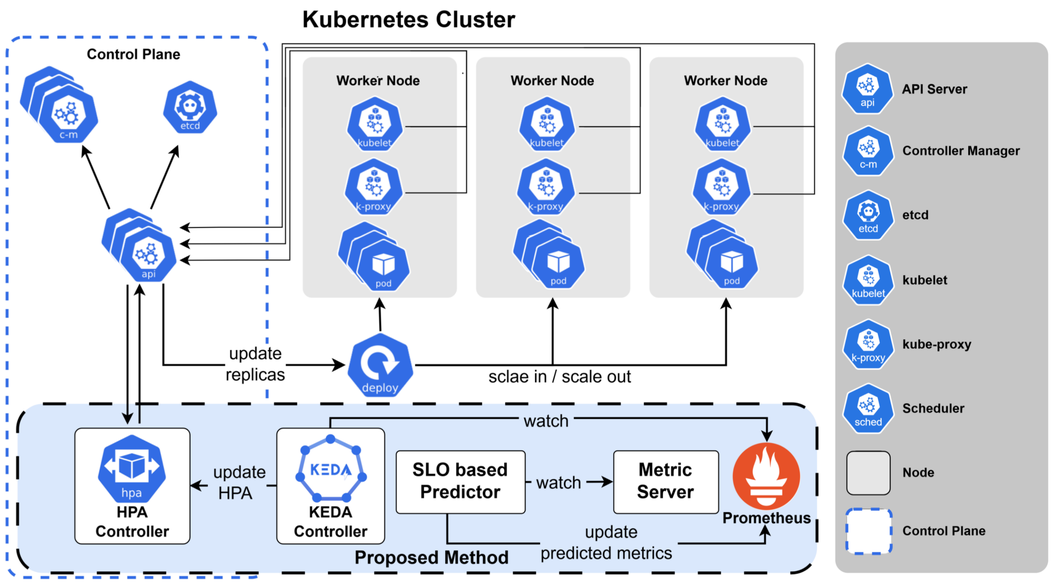

Figure 1. Proposed autoscaling architecture in Kubernetes.

III. Evaluation

We compare three autoscaling strategies under the same microservice workload:

1. Baseline HPA
   Uses current CPU utilization with different target values such as 80 percent
2. Predictive scaling with LSTM only
   Uses LSTM based CPU prediction as the scaling metric without explicit SLO feedback
3. Proposed SLO aware predictive scaling
   Uses the composite load metric that combines predicted CPU utilization and the latency aware weight for several values of ω = 0.05, 0.1

- In addition, we evaluate these three strategies with a cost model summarized in Equations (1) to (5).
- For each request, we define an indicator V_i that becomes zero when the request latency L_i satisfies the SLO target and one when the latency exceeds the SLO target
- The total number of SLO violating requests V_{total} is obtained by summing all V_i, and the SLO related cost $Cost_{SLO}$ is computed as W_{SLO} multiplied by V_{total} divided by the total number of requests N.
- Resource cost $Cost_{res}$ is calculated from the number of pods of each service over time and the resource cost R_s of each service, weighted by W_{res}. The overall cost $Cost_{total}$ is then defined as the sum of $Cost_{SLO}$ and $Cost_{res}$.

$$V_i = \begin{cases} 0, & \text{if } L_i \le SLO_{target} \\ 1, & \text{if } L_i > SLO_{target} \end{cases} \quad (1)$$

$$V_{total} = \sum_{i=1}^{N} V_i \quad (2)$$

$$Cost_{SLO} = W_{SLO} \times \frac{V_{total}}{N} \quad (3)$$

$$Cost_{res} = W_{res} \times \sum_{t=1}^{t_{end}} \sum_{s \in S} pod_{s,t} \times R_s \quad (4)$$

$$Cost_{total} = Cost_{SLO} + Cost_{res} \quad (5)$$

Figures 2 and 3 illustrate the relationship between total pod usage and latency across different autoscaling strategies, where Figure 2 focuses on mean latency and Figure 3 highlights tail latency (p95), revealing clear performance differences among the strategies.

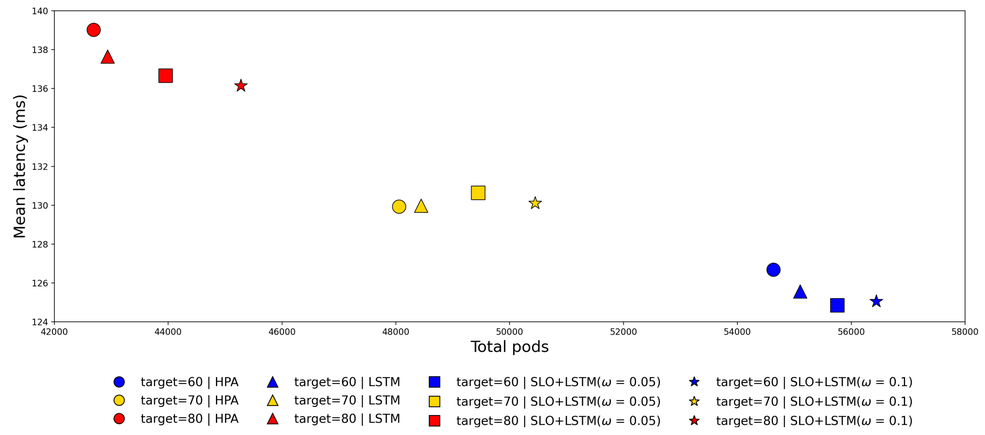

Figure 2. Resource usage versus mean latency.

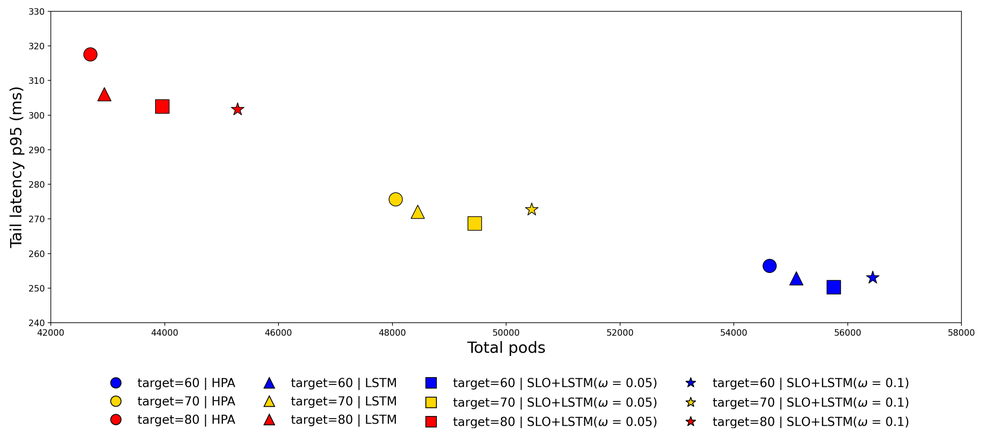

Figure 3. Resource usage versus tail latency.

Table 1. Total cost of autoscaling schemes under different SLO priority settings.

Autoscaling scheme	$Cost_{SLO}$ (W_{SLO} = 0.1)	$Cost_{SLO}$ (W_{SLO} = 1)	$Cost_{res}$	$Cost_{total}$ (W_{SLO} = 0.1)	$Cost_{total}$ (W_{SLO} = 1)
HPA (target value=80)	720.37	7,203.7	1,328.81	2,049.18	8,532.51
LSTM (target value=80)	263.27	2,632.77	1,338.80	1,602.07	3,971.57
LSTM+SLO(ω=0.05) (target value=80)	87.49	874.9	1,368.50	1,455.99	2,243.4
LSTM+SLO(ω=0.1) (target value=80)	74	740.07	1,413.30	814.07	2,153.37

- Table 1 summarizes the cost analysis under different SLO weight settings $Cost_{SLO}$, $Cost_{\text{res}}$, and $total_{cost}$ across the evaluated autoscaling schemes.
  In our experiments, an SLO violation is defined as a p95 latency exceeding 300ms, and the analysis focuses on the violation-prone scenario with a target value of 80.
- As shown in Table 1, the LSTM-based approach substantially reduces the SLO related cost compared with the baseline HPA while keeping resource cost at a similar level. Moreover, the SLO-aware LSTM+SLO methods further lower $Cost_{SLO}$ with only a modest increase in $Cost_{res}$, resulting in the lowest overall cost. This advantage becomes more pronounced as W_{SLO} increases, indicating that the proposed method is especially beneficial when SLO violations are heavily penalized.

IV. Conclusion

In summary, this research demonstrates that incorporating SLO awareness into predictive autoscaling improves tail latency and SLO compliance.
Experiments on the Google Online Boutique application with workloads derived from the Alibaba Cluster Trace show the following implications.

- The proposed method reduces p95 tail latency compared with standard CPU-based HPA and prediction only scaling
- It lowers the SLO violation ratio while keeping pod usage at a similar or slightly higher level
- Under a cost model where SLO violations are heavily weighted, the proposed method achieves the lowest total cost among the evaluated strategies

Acknowledgement

This research was supported by the Regional Innovation System & Education (RISE) program through the Jeju RISE Center, funded by the Ministry of Education (MOE) and the Jeju Special Self-Governing Province, Republic of Korea (2025-RISE-17-001). It was also supported by the Basic Science Research Program to the Research Institute for Basic Sciences (RIBS) of Jeju National University through the National Research Foundation of Korea (NRF), funded by the Ministry of Education (RS-2019-NR040080).

References

[1] K. Aykurt et al.et al., “HyPA: Hybrid Horizontal Pod Autoscaling with Automated Model Updates,” 2023 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN), Dresden, Germany, 2023, pp. 8-14.

[2] S. Xie, J. Wang, B. Li, Z. Zhang, D. Li and P. C. K. Hung, “PBScaler: A Bottleneck-Aware Autoscaling Framework for Microservice-Based Applications,” IEEE Transactions on Services Computing, vol. 17, no. 2, pp. 604-616, 2024.

* Corresponding author: Joon-Min Gil