Smarter Scaling: Predictive Autoscaling for Kubernetes

Maurizio Giacobbe*, Sarah Zanafi†, Jiregna Abdissa Olana†, Antonio Puliafito†‡

*Dept. of Math., Computer, Physical and Earth Sciences, University of Messina, Messina, Italy

†Department of Engineering, University of Messina, Messina, Italy

‡Consorzio Interuniversitario Nazionale per l'Informatica (CINI), Rome, Italy

{mgiacobbe, szanafi, jiolana, apuliafito}@unime.it
0000-0001-6178-7132 (Giacobbe), 0000-0002-0126-7837 (Zanafi)
0009-0002-2220-1359 (Olana) 0000-0003-0385-2711 (Puliafito)

***Abstract*—Ensuring efficient and scalable computing is critical for real-time data processing, resource optimization, and service reliability. Effective load balancing mechanisms are necessary to manage workload distribution, minimize latency, and maintain system performance in highly dynamic edge computing environments. This paper proposes a hybrid autoscaling approach that enhances the default HorizontalPodAutoscaler (HPA) by integrating predictive scaling techniques. Results demonstrate the effectiveness of the predictive autoscaling model compared with fixed threshold-based and traditional HPA.**

***Index Terms*—Edge-cloud continuum, IoT, kubernetes, load balancing, predictive autoscaling, scalability**

I. Introduction

The consolidation of *IoT-as-a-Service (IoTaaS)* [1] is transforming the landscape of smart cities, enabling advanced urban services through the seamless integration of cloud computing and edge computing. **Kubernetes** has emerged as a *de facto* standard for orchestrating containerized workloads in such distributed scenarios. In Kubernetes, a *HorizontalPodAutoscaler (HPA)* (Figure 1) automatically updates a workload resource in which a **pod** is the fundamental unit as a set of running containers in a cluster.

II. Background and Related work

Load balancing is a critical challenge that ensures the optimal allocation of computational tasks between edge and cloud resources, especially for latency sensitive applications operating in Kubernetes-based edge cloud environments [2].

Traditional cloud-centric models face limitations in handling the dynamic and heterogeneous nature of computational workloads. Therefore, it is essential to explore adaptive load balancing mechanisms [3] that account for factors such as network bandwidth, computational capacity, and geographical distribution of devices [4].

Starting from the native HPA mechanism, we propose a *hybrid autoscaling approach* that integrates *predictive scaling techniques* alongside HPA to anticipate workload fluctuations and improve system responsiveness. Through an experimental use case, we assess the effectiveness of this combined strategy in managing workload distribution. The *Pearson correlation coefficient* is used as a key metric to evaluate the alignment between predicted and observed use of resources, offering insight into the model's reliability and the overall performance of the system.

This work was partially supported by the European Union - Next Generation EU under the Italian National Recovery and Resilience Plan (NRRP), Mission 4, Component 2, Investment 1.3, project SecCO, CUP D33C22001300002, and project 3D-SEECSDE, CUP J33C22002810001, partnership on "SEcurity and RIghts in the CyberSpace" (PE00000014 - program "SERICS").

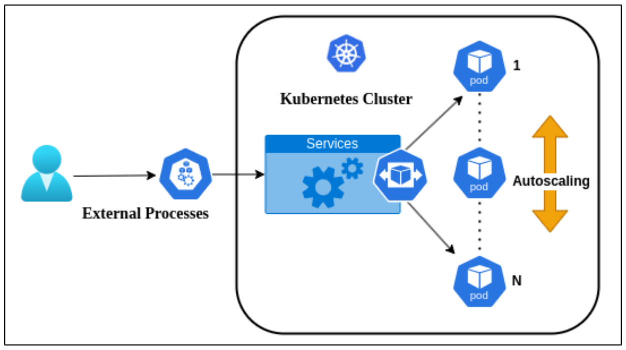

Fig. 1. Overview of the typical autoscaling of pods in Kubernetes.

III. Use Case

We consider a typical scenario of a distributed application in a Kubernetes cluster of nodes.

A. Key Challenges

- **Predictive autoscaling**
- **Gradual scaling process to avoid over-reaction**

To address these challenges, we propose an approach that combines predictive autoscaling with traditional HPA.

Figure 2 shows its operative positioning at the Workload Autoscaling Layer in a Kubernetes stack.

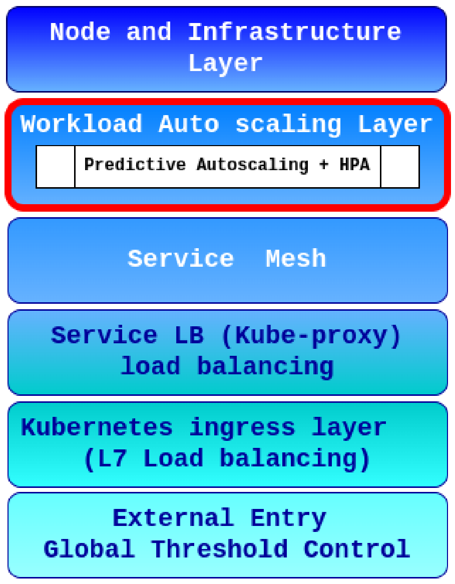
Node and Infrastructure Layer
Workload Auto scaling Layer
Predictive Autoscaling + HPA
Service Mesh
Service LB (Kube-proxy) load balancing
Kubernetes ingress layer (L7 Load balancing)
External Entry Global Threshold Control

Fig. 2. Predictive Autoscaling with HPA at the Workload Autoscaling Layer in a Kubernetes stack

CPU load is used as a key parameter to control resource allocation. The key idea is to predict future CPU load based on historical CPU usage trends and adjust the number of resources (e.g., pods) in advance to meet anticipated demand.

The predictive autoscaling process follows these steps: *(i) data collection, (ii) future load prediction, (iii) gradual scaling, (iv) resource adjustment.*

The number of pods is updated as:

$$\text{Pods}_{\text{new}} = \max(1, \min(\text{max\_pods}, \text{Pods}_{\text{current}} + \Delta)) \quad (1)$$

The relationship between CPU load and pod count is modeled linearly:

$$N_{\text{POD}}(t) = \alpha \cdot L_{\text{CPU}}(t) + \beta \quad (2)$$

where α is the proportionality coefficient determining how much each unit of CPU load affects the number of PODs, and β is the constant term representing the base number of PODs when the CPU load is zero. A threshold-based approach adjusts pods based on load limits:

$$N_{\text{POD}}(t) = \begin{cases} N(t-1) + \Delta, & L(t) > L_{\text{thresh}} \\ N(t-1) - \Delta, & L(t) < L_{\text{low}} \\ N(t-1), & \text{otherwise} \end{cases} \quad (3)$$

The correlation is quantified with the Pearson coefficient r, indicating how closely pod count tracks CPU load.

CPU evolution is computed as:

$$\text{CPU}_{\text{new}} = \max(0, \min(100, \text{CPU}_{\text{previous}} + \Delta)) \quad (4)$$

We implemented computational flows in **Node-RED**, as it is commonly used on edge or gateway devices, enabling quick adaptation from prototyping to deployment.

Figure 3 shows a 1-minute time slot of the real-time CPU load (%), the predicted CPU load (%), the number of pods allocated over time, and the respective Pearson coefficient trend.

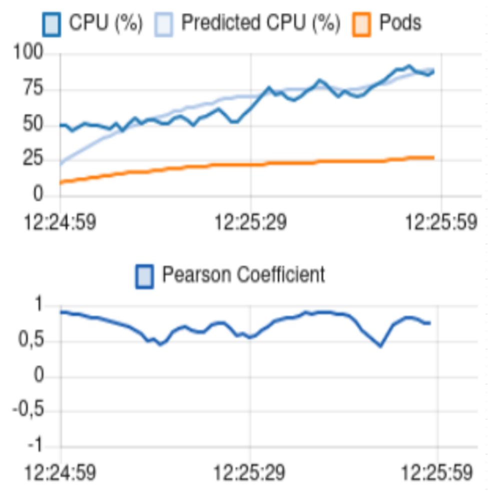

Fig. 3. Predictive Autoscaling with HPA in a Node-RED Dashboard

The analysis validates the effectiveness of predictive autoscaling models, which achieve a desirable trade-off between responsiveness and stability by maintaining a strong but not overly reactive correlation with system load.

IV. Conclusion and Future Work

This study emphasizes a new autoscaling strategy that improves Kubernetes' *HorizontalPodAutoscaler (HPA)* by adding predictive features to overcome the shortcomings of fixed threshold-based methods in edge-cloud settings. The proposed system ensures more stable, responsive, and efficient resource allocation by gradually adjusting the number of pods. Future developments will focus on validating the approach in real-world, multi-cluster Kubernetes deployments. Additionally, we plan to investigate more advanced forecasting techniques to further improve accuracy and adaptability in dynamic edge-cloud settings.

References

[1] S. Zanafi, N. Aknin, M. Giacobbe, M. Scarpa and A. Puliafito, "Enabling Sustainable Smart Environments Using Fog Computing," 2018 International Conference on Electronics, Control, Optimization and Computer Science (ICECOCS), Kenitra, Morocco, 2018, pp. 1-6, doi: 10.1109/ICECOCS.2018.8610509.

[2] P. P S, R. K, A. Sharma, A. Temura, R. Shah and P. Tammana, "DL3: Adaptive Load Balancing for Latency-critical Edge Cloud Applications," 2024 20th International Conference on Network and Service Management (CNSM), Prague, Czech Republic, 2024, pp. 1-5, doi: 10.23919/CNSM62983.2024.10814405.

[3] J. Santos, T. Wauters, F. D. Turck and P. Steenkiste, "Towards Optimal Load Balancing in Multi-Zone Kubernetes Clusters via Reinforcement Learning," 2024 33rd International Conference on Computer Communications and Networks (ICCCN), Kailua-Kona, HI, USA, 2024, pp. 1-9, doi: 10.1109/ICCCN61486.2024.10637606.

[4] Q. Liu, E. Haihong and M. Song, "The Design of Multi-Metric Load Balancer for Kubernetes," 2020 International Conference on Inventive Computation Technologies (ICICT), Coimbatore, India, 2020, pp. 1114-1117, doi: 10.1109/ICICT48043.2020.9112373.