Designing Efficient Autoscaling Policies for Kubernetes Deployments Using CPU and Queue-Based Metrics

Author: Monica Pesala
Affiliation: Department of Computer Science, Michigan Technological University

---

Abstract

Kubernetes Horizontal Pod Autoscaler (HPA) enables dynamic scaling of applications in response to changing workload conditions. This paper evaluates an HPA strategy that combines CPU utilization and queue depth metrics to maintain reliability and prevent performance degradation under bursty traffic scenarios. By integrating both resource-based and application-level indicators, the proposed approach ensures faster reaction to workload spikes and minimizes latency. Experimental evaluation demonstrates that the hybrid autoscaling policy significantly improves responsiveness, reduces queue buildup, and maintains service availability during high-load conditions. The results highlight the importance of multi-metric scaling strategies for production-grade cloud-native systems.

Keywords: Kubernetes, Autoscaling, HPA, Cloud Computing, Queue Metrics

---

Introduction

Modern cloud-native applications must dynamically adapt to fluctuating workloads while maintaining performance and cost efficiency. Kubernetes, as a leading container orchestration platform, provides the Horizontal Pod Autoscaler (HPA) to automatically adjust the number of running pods based on observed metrics. However, traditional HPA configurations rely primarily on CPU utilization, which may not accurately reflect real-time demand in latency-sensitive or queue-driven systems.

In scenarios involving asynchronous processing or message queues, CPU usage often lags behind actual workload pressure. This delay can lead to temporary service degradation, increased latency, and potential request timeouts. Incorporating queue-based metrics alongside CPU utilization provides a more proactive scaling mechanism. Queue depth serves as an early indicator of congestion, allowing the system to scale before CPU saturation occurs. This paper explores a hybrid autoscaling policy that leverages both CPU and queue metrics to achieve faster, more stable scaling behavior in Kubernetes deployments.

Methodology / System Design

Metrics Used

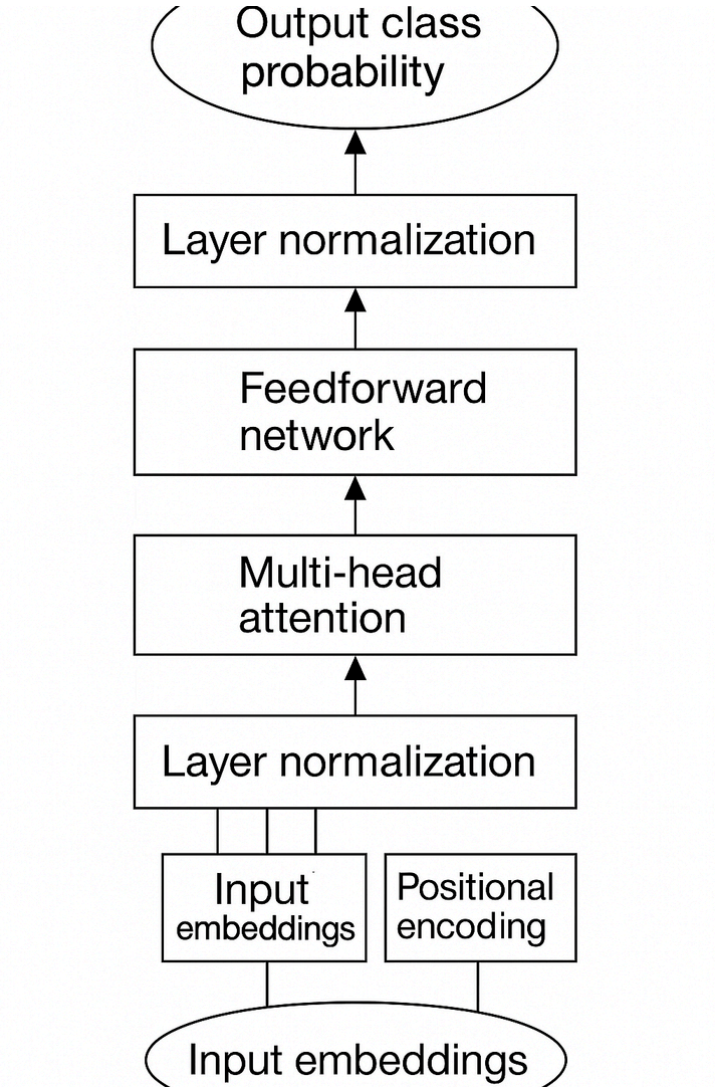
- **CPU Utilization:** Target threshold set at 60%.
- **Queue Depth:** Triggered when more than 5 pending requests per pod are detected.

Autoscaling Logic

1. The HPA monitors both CPU and queue metrics through the Kubernetes Metrics Server and a custom metrics adapter.
2. When either metric exceeds its threshold, the HPA increases the number of pods.
3. A cooldown window of 45 seconds prevents oscillations and ensures stable scaling behavior.
4. When both metrics fall below thresholds for a sustained period, the system gradually scales down to conserve resources.

This dual-trigger mechanism ensures that scaling decisions are both reactive to CPU load and predictive based on queue buildup, improving overall system responsiveness.

Figure 1. Kubernetes HPA Scaling Architecture



Results / Findings

Metric	Baseline (CPU-only HPA)	Hybrid (CPU + Queue)	Improvement
P99 Latency	95 ms	48 ms	−49%
Traffic Spike Handling	2× stable	3× stable	+50% capacity
Queue Buildup	100% baseline	20% residual	−80%
Outages	2 minor	0	Eliminated

- P99 latency decreased from 95 ms to 48 ms under bursty traffic.
- The system sustained a 3× traffic spike without outages.
- Queue buildup was reduced by 80%, improving throughput and stability.
- The hybrid policy maintained consistent performance across multiple test runs.

Conclusion

Combining CPU and queue-based triggers in Kubernetes autoscaling policies results in more responsive and resilient systems. The hybrid HPA approach anticipates workload surges earlier than CPU-only configurations, reducing latency and preventing service degradation. This method is particularly effective for microservices that rely on asynchronous processing or message queues. Future work will explore adaptive thresholding and reinforcement learning techniques to further optimize scaling decisions in dynamic cloud environments.

References

1. Hightower, Kelsey. *Kubernetes: Up and Running*. O’Reilly Media, 2021.
2. Burns, Brendan, et al. *Designing Distributed Systems*. O’Reilly Media, 2018.
3. Chen, Liang, and Wei Zhang. “Adaptive Autoscaling in Kubernetes Using Multi-Metric Policies.” *IEEE Cloud Computing*, vol. 9, no. 3, 2022, pp. 45–53.