ISSN 2320 - 2602
Volume 14 No. 6, June 2025
International Journal of Advances in Computer Science and Technology
Available Online at http://www.warse.org/IJACST/static/pdf/file/ijacst011462025.pdf
https://doi.org/10.30534/ijacst/2025/011462025

Efficient Microservices Scaling: Kubernetes, Autoscaling, and Load Balancing

Smirnov Andrei
Master's degree, Perm National Research Polytechnic University, Russia
drvanhelcing@rambler.ru

Received Date : April 28 , 2025 Accepted Date : May 29, 2025 Published Date : June 07, 2025

ABSTRACT
This article addresses the issue of efficient scaling of microservice architecture under variable load conditions and limited resources. Special attention is given to the use of automatic scaling mechanisms in Kubernetes, such as the Horizontal Pod Autoscaler and Vertical Pod Autoscaler, as well as the selection and interpretation of load metrics that form the basis for scaling decisions. It explores how different scaling approaches affect system performance and infrastructure operating costs. It also investigates how scaling strategies, the degree of automation, and load balancing configurations influence service stability, resource utilization, and the overall quality of distributed system operation.

Key words: microservices, scaling, Kubernetes, autoscaling, load metrics, Horizontal Pod Autoscaler, Vertical Pod Autoscaler, load balancing.

1. INTRODUCTION

Modern web applications stand in need of high degrees of fault tolerance, in addition to the ability to adapt to changing workloads. With the expanding sizes of data volumes and user populations, regular resource management approaches become more and more inadequate, often resulting in either extensive overprovisioning or severe resource shortages under peak loads. In this context, dynamic scaling emerges as a vital mechanism for maintaining system stability and ensuring efficient utilization of infrastructure.
Microservice architecture is widely adopted in contemporary application development and enables granular resource management across individual system components. With the use of the Kubernetes platform and its built-in autoscaling capabilities, namely the Horizontal Pod Autoscaler (HPA) and Vertical Pod Autoscaler (VPA), services can effectively cope with changing workloads without leaving the system idle for long. This is why the selection of a scaling approach has a great impact on system performance and operational costs, making achieving balance between the two a vital application in design.

The goal of this research to analyze present-day methods of dynamic microservices scaling based on load metrics, with a particular focus on horizontal and vertical scaling approaches. The relevance of the study is underscored by the growing demand for flexibility and efficiency in distributed systems, especially in cloud-native environments. The methodology includes a comparative analysis of existing Kubernetes autoscaling mechanisms and an assessment of their impact on performance and infrastructure expenses.

2. MAIN PART. DYNAMIC MICROSERVICES SCALING METHODS: APPROACHES AND LOAD METRICS

Microservices' scalability is essential for maximizing the performance, fault tolerance, and cost efficiency of the system as a whole. Modern software systems cannot rely solely on static resource allocation because of the dynamic behavior of application workloads that constantly change in real time. Dynamic scaling supports the elastic adjustment of the resources for processing in line with the current operational needs of a running service to minimize infrastructure costs and avoid system overload risk. This is made possible by load metrics that offer insights on whether to raise or lower resource allocation.
Dynamic scaling involves the automatic adjustment of service replica counts or their allocated resources in response to observed load patterns. Unlike static approaches that provision resources with a fixed overhead, dynamic scaling allows the system to align more closely with actual demand [1]. This capability is mostly crucial for high-throughput and cloud-native applications, in which responsiveness to changing workloads directly impacts operational efficacy.
The core principles of dynamic scaling encompass both **reactive and predictive resource management.** Reactive scaling relies on monitoring the state of the system. When predefined boundaries are exceeded, additional resources are brought online and vice versa, when demand tapers off, the resource allocation is scaled down. Predictive scaling, by contrast, makes use of machine learning algorithms as well as learned data to anticipate spikes in demand and allow the system to advance-plan adjustments in capacity in line with expected increases in workload.

Effective automated scaling requires the use of objective indicators to inform decision-making. The primary metrics used for this purpose can include CPU utilization, latency and others (table 1).

Table 1: Load metrics for autoscaling in microservices [2, 3]

Metric	Description
CPU utilization	One of the most common indicators used for scaling containerized microservices. High CPU utilization may indicate a lack of computational resources, while low utilization may suggest overprovisioning.
Memory utilization	A crucial indicator, especially for services performing intensive data processing. Memory shortages may cause crashes, whereas excess usage increases infrastructure costs.
Requests per second (RPS)	Helps assess current load on application programming interface and web services. Used to scale services handling HTTP requests and to respond appropriately to traffic spikes.
Latency	Indicates delays in request processing. High latency may signal system overload or insufficient network throughput.
Disk usage and IOPS	A key metric for services working actively with file systems and databases. Scaling may improve performance when disk I/O becomes a bottleneck.
Custom metrics	Specific indicators that reflect business logic, such as the number of concurrent transactions or active users.

Which metrics are selected is a matter of the kind of workload and architectural characteristics of the application. Such data can be gathered and processed automatically through APIs like Prometheus, the Metrics Server, and other tools for observability.

Kubernetes has two primary automatic scaling tools: the HPA and the VPA. The first one performs horizontal scaling by increasing or decreasing the number of pods based on predefined metrics. This method is commonly employed for scaling microservices that operate under highly variable loads. For instance, when average CPU utilization exceeds a certain threshold, such as 80%, HPA can automatically create additional pods to redistribute the load across more instances [4].

Conversely, VPA enables vertical scaling through the adjustment of computational resources allocated to individual pods. While the HPA mostly deals with the number of pod replicas, the VPA focuses on the resource allocation of individual pods. The adjustment of pod configurations through the VPA typically needs the restarting of pods, which can cause temporary service disruptions during the scaling process.

The combined use of HPA and VPA offers maximum efficiency in managing resource allocation. In services experiencing highly volatile traffic, HPA entitle rapid horizontal scaling by increasing the number of pods on demand, while VPA boosts resource utilization under more stable workloads through adjusting the resource limits of individual pods.

The choice of a suitable scaling approach is largely determined by the unique nature of the application and its resource availability needs. Applications that are subject to sudden traffic spikes, which is the case with e-commerce websites during special promotions, significantly benefit from immediate horizontal scaling. Applications that require large computational capacities, as an instance, video processing tasks, are better supported through vertical scaling, as increasing the resource allocation of a single pod reduces the overhead associated with inter-process communication. A hybrid strategy combining both types of scaling may be especially beneficial in scenarios characterized by extended but intermittent load patterns.

Dynamic scaling of microservices is a critical component of their efficient operation. Employing both horizontal and vertical scaling enables systems to adapt seamlessly to changing runtime conditions. Load metrics form the basis of making informed scaling decisions, and Kubernetes mechanisms like HPA and VPA allow the automatic implementation of those decisions. Choosing the appropriate scaling strategy involves careful deliberation of the nature of the application to balance between optimizing the performance and managing infrastructure costs.

3. THE IMPACT OF SCALING ON PERFORMANCE AND INFRASTRUCTURE COST

Once an appropriate scaling strategy has been selected, the key question becomes its effectiveness under real-world operating conditions. One of the primary criteria for evaluating a scaling approach is not only the system's performance but also the cost of the infrastructure required to support it [5]. Achieving a well-calibrated balance between these two factors ensures both the technical resilience of a microservice architecture and the economic viability of its maintenance.

Horizontal scaling offers substantial benefits in terms of service availability and the ability to handle multiple concurrent requests. This approach can be quite valuable for distributed systems, where microservices are packaged in containers that can be easily replicated. Yes, the growth in pods is accompanied by a rising need for resources, as each instance requires its own memory and CPU. In addition, horizontal scaling presents some obstacles in traffic management, state consistency, and inter-service communication. These hurdles can negate the overall benefits of scaling, particularly in situations where there is poor decoupling between microservices or in situations with high interdependence.

Vertical scaling is accomplished by increasing more computing resources to each individual instance. This approach is often less complex to implement and does not entail radical changes to application design. It also comes with physical as well as financial limitations. It is often more expensive to scale up a single node than scaling out several instances that are less powerful but spread out. Further, vertical scaling can be inefficient if microservices don't fully utilize the provisioned resources, leading to resource wastage

and underutilization. It should be noted that vertical scaling in Kubernetes using the VPA may need pod restarts, which cause temporary service unavailability.

A comparative testing work reveals that HPA consistently maintains lower and more stable response times under increasing load, with median latencies ranging from 2,5 to 2,7 ms and standard deviation values remaining below 0,4 ms [6]. In contrast, VPA shows both higher average response times (up to 3,5 ms) and significantly greater variability in the 99th percentile, with deviations exceeding 240 ms in some scenarios (figure 1).

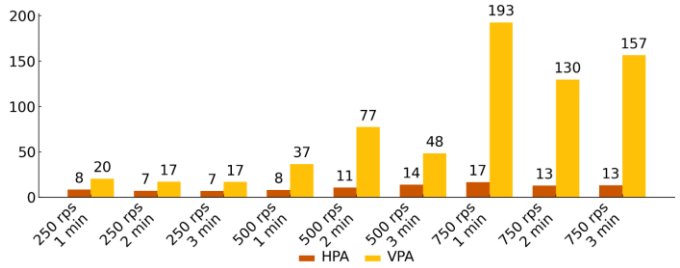

Figure 1: Comparative 99th percentile latency for HPA and VPA across load scenarios, ms

From an economic perspective, scaling is inherently a trade-off. In cloud environments, where billing is based on consumed resources, over-scaling can lead to significant increases in operational costs, while under-scaling may result in performance degradation and a diminished user experience [7]. Therefore, selecting an appropriate level of automation and properly configuring scaling triggers is essential. Monitoring and telemetry systems not only enable real-time observation of microservice behavior but also reveal usage patterns that can inform predictive scaling strategies. This is particularly valuable in scenarios with anticipated traffic surges, such as promotional campaigns or consistent peaks in user activity during specific hours.

An additional factor influencing the cost-efficiency of scaling is the effectiveness of load balancers. These components distribute incoming traffic across pods and nodes, and their configuration directly affects the evenness of resource utilization. Poorly optimized load balancing strategies can lead to imbalances where some resources remain underutilized while others become overloaded, potentially triggering unnecessary cascading scaling actions. Modern solutions such as Ingress controllers and service meshes enable more precise traffic distribution, allowing for a reduction in the total number of pods by maximizing their utilization without compromising performance [8].Thus, the impact of scaling on both performance and infrastructure cost depends not only on the choice between horizontal and vertical scaling mechanisms, but also on the degree of automation, the precision of metric configuration, and the quality of load balancing. An effective scaling strategy is not merely a reactive response to current load levels, it is a carefully designed model of system-wide coordination, informed by business objectives, traffic profiles, and economic constraints.

4. CONCLUSION

Microservice scaling is an essential component of the effective operation of distributed systems under variable workloads. The use of autoscaling mechanisms in Kubernetes, such as the HPA and VPA, enables real-time resource adaptation based on load metrics and application behavior. A flexible scaling approach enhances service resilience, maintains high availability, and ensures timely responses to increases in user demand.

However, the selection of a scaling strategy requires careful analysis. It must take into account workload characteristics, architectural specifics, fault tolerance requirements, and cost-efficiency considerations. Achieving an optimal balance between horizontal and vertical scaling, fine-tuning metrics, and employing advanced load balancing solutions not only improves system performance but also reduces infrastructure costs. In the context of rapidly evolving cloud technologies, scaling is no longer merely a technical concern, it becomes a strategic aspect of IT infrastructure management.

REFERENCES

1. A. Dudak A. **Microservice architecture in frontend development**, *Norwegian Journal of development of the International Science*, no. 145, pp. 99-102, 2024. DOI: 10.5281/zenodo
2. S. Bolgov. **Optimizing microservices architecture performance in fintech projects**, *Bulletin of the Voronezh Institute of High Technologies*, vol. 19, no. 1, 2025.
3. M.P. Kuranage, E. Hanser, A. Bouabdallah, L. Nuaymi, P. Bertin. **CPU throttling-aware AI-based autoscaling for Kubernetes.** *In2024 IEEE 35th International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*, pp. 1-7, 2024. DOI: 10.1109/PIMRC59610.2024.10817283
4. N. Bjørndal, L.J. de Araújo, A. Bucchiarone, N. Dragoni, M. Mazzara, S. Dustdar. **Benchmarks and performance metrics for assessing the migration to microservice-based architectures.** *J. Object Technol*, vol. 20, no. 2, pp. 3-1, 2021.
5. A. Blazhkovskii. **Optimization of mobile application performance: modern approaches and methods.** *ISJ Theoretical & Applied Science*, vol. 140, no. 12, pp. 290-294, 2024. DOI: 10.15863/TAS.2024.12.140.35
6. J. Nilsen. **Performance Evaluation of Kubernetes Autoscaling strategies on GKE clusters**. 103 pp. 2023.
7. A. Shahidinejad, M. Ghobaei-Arani, M. Masdari. **Resource provisioning using workload clustering in cloud computing environment: a hybrid approach.** *Cluster Computing*, vol. 24, no. 1, pp. 319-42, 2021. DOI: 10.1007/s10586-020-03107-0
8. G. Rathi, S. Amin, S. Jain, M. Yachawad, K. Digholkar. **Performance Analysis of Different Ingress Controllers Within the Kubernetes Cluster.** *In2024 IEEE International Conference on Information Technology, Electronics and Intelligent Communication Systems (ICITEICS)*, pp. 1-6, 2024. DOI: 10.1109/ICITEICS61368.2024.10625280