Machinery & Energetics
Vol. 16, No. 4. 2025

Journal homepage:
https://technicalscience.com.ua/en

UDC 004.738.5:004.042
DOI: 10.31548/machinery/4.2025.21

Khrystyna Terletska*
Bachelor
Lviv Polytechnic National University
79000, 12 Stepan Bandera Str., Lviv, Ukraine
https://orcid.org/0009-0002-7302-2625

Performance trade-offs between predictive and reactive autoscaling in stateful microservices

Abstract. The relevance of studying the trade-offs between reactive and predictive autoscaling of stateful microservices was due to the need to improve stability, resource performance and reliability of cloud systems in dynamic conditions. The aim of the study was to assess the effectiveness of reactive, predictive and hybrid autoscaling strategies in stateful microservice architectures. The methodological basis of the research relied on comparative, content, systems and structural-functional analyses, as well as modelling, which made it possible to comprehensively investigate the performance and stability of different approaches to autoscaling in stateful microservices. Theoretical analysis showed that autoscaling in microservice systems provides adaptive performance, and the choice of strategy depends on the nature of the workload, latency sensitivity and consistency requirements. The review of models demonstrated that reactive autoscaling responds quickly to changes but is accompanied by high latency, performance fluctuations and the risk of state divergence between replicas, whereas predictive autoscaling reduces scaling delay and increases stability and resource efficiency but requires accurate models and additional computational resources. Analysis of the results revealed that hybrid strategies integrate the advantages of both approaches, providing both flexibility and stability, and that system efficiency is determined by a trade-off between latency, stability, state consistency, resource efficiency and financial costs. The results of the study may be useful for developers and architects of cloud microservice systems, IT engineers and DevOps teams for optimising scaling strategies, increasing service stability and ensuring efficient resource use in different cloud environments

Keywords: Kubernetes; resource optimisation; cloud performance; dynamic resource allocation; system stability; observability metrics

INTRODUCTION

Modern cloud microservice architectures require efficient autoscaling mechanisms, especially for stateful services with critical state consistency and high latency sensitivity. Traditional reactive approaches provide basic stability but suffer from reaction delay and inefficient resource use, whereas predictive models make it possible to anticipate load peaks but require complex analytics and precise tuning. In this regard, the study of performance trade-offs between reactive and predictive approaches and the development of a conceptual model for choosing a scaling strategy for stateful environments are relevant and make it possible to improve stability, resource efficiency and reliability of cloud systems in dynamic conditions.

In the modern academic discourse, research on microservice autoscaling increasingly focuses on adaptability, load prediction and resource optimisation. Thus, in the work of I. Varzhanskiy *et al.* (2025), approaches to load forecasting and strategic resource analysis were considered, demonstrating the potential of applying predictive and hybrid methods to increase system resilience and efficiency. The study showed that the integration of analytical methods and forecasting makes it possible to improve scaling planning and reduce the risks of peak loads. In the work of O. Romanov *et al.* (2024), dynamic scaling of containerised applications using Kubernetes Cluster Autoscaler and Auto Scaling Group was analysed. The authors

Article's History: Received: 10.07.2025; Revised: 15.10.2025; Accepted: 27.11.2025.

Suggested Citation:
Terletska, Kh. (2025). Performance trade-offs between predictive and reactive autoscaling in stateful microservices. *Machinery & Energetics,* 16(4), 21-33. doi: 10.31548/machinery/4.2025.21.

*Corresponding author

demonstrated that the combined use of horizontal and vertical scaling reduces latency and increases the performance of stateful microservices in multicluster environments. The results of the work confirmed the relevance of combining reactive, predictive and hybrid strategies to achieve an optimal balance between stability, reaction speed and resource efficiency. At the same time, the study of T. Babenko *et al.* (2025) focused on automated data collection and risk assessment of digital assets using AI methods. The analytical and predictive approaches used demonstrated the possibilities of integrating machine learning to optimise the management of complex systems. This highlighted the prospects of applying intelligent algorithms to support scalable and resilient microservice architectures. In the study, B. Guntupalli & S. Vamshi (2023) analysed the design of microservices for processing high data volumes. The results showed that the correct combination of horizontal and vertical scaling has a critical impact on system performance and economic efficiency. The authors emphasised the importance of a structured approach to microservice architecture in large cloud environments.

In studies of autoscaling for stateful microservices, attention was also paid to increasing system stability and resource efficiency. M. Mekki *et al*. (2025) proposed an approach to proactive lifecycle management in multi-cluster containerised environments, which envisages load forecasting and adaptive resource deployment. The study showed that this approach reduces the risk of state divergence between replicas and increases system resilience to load fluctuations, demonstrating practical value in complex distributed architectures. C. Meng *et al.* (2023) developed Deepscaler – a comprehensive horizontal and vertical scaling model based on spatio-temporal graph neural networks with adaptive graph learning. The review showed that the approach makes it possible to effectively forecast load peaks and optimise resource distribution, which reduces system reaction latency and increases the stability of stateful microservices. The results of the study demonstrated the advantages of using deep learning for comprehensive scaling, particularly in scenarios with dynamic and unpredictable workloads. In turn, M. Thompson *et al*. (2023) focused on intelligent workload balancing in multi-cloud environments, applying adaptive algorithms for resource distribution between providers. The study showed that such an approach makes it possible to increase system performance and fault tolerance while reducing the risk of overloading individual components. The literature review demonstrates that the integration of intelligent balancing strategies with predictive scaling provides comprehensive optimisation of stateful microservice operation in complex and distributed cloud infrastructures. T. Prasad (2023) proposed an AI-driven predictive scaling approach that uses machine-learning algorithms to forecast workload and automatically allocate resources. The study showed that the use of AI reduces scaling latency and increases system stability, and the results demonstrated the effectiveness of the predictive approach in complex cloud environments with dynamic workloads.

Previous studies focused on the application of reactive and predictive autoscaling models for cloud and cloud-native architectures, paying attention to forecasting efficiency, system stability and resource optimisation. At the same time, most works leave the performance trade-offs in stateful microservices without detailed analysis, in particular the relationship between reaction latency, state consistency, stability and financial costs, and do not offer a comprehensive assessment of hybrid strategies. The aim of the study was to examine the effectiveness of reactive, predictive and hybrid autoscaling strategies in stateful microservice architectures, taking into account performance trade-offs and financial feasibility. To achieve this aim, the following objectives were set: to analyse the theoretical aspects of these strategies in terms of the impact on latency, stability, state consistency and efficiency of resource provision, as well as to develop a generalised comparison model for strategies in stateful microservices, enabling the identification of key trade-offs and contextual factors for choosing the optimal model.

MATERIALS AND METHODS

The study was an interdisciplinary analytical investigation in which elements of theoretical, comparative, modelling and predictive analysis were combined. The theoretical-analytical group of methods included systems analysis and the method of analytical generalisation, which made it possible to identify the main trends in the development of scaling strategies in stateful environments and to classify approaches according to adaptability, type of computational load, consistency requirements and latency level. A comprehensive source base was formed in the study, which included 34 peer-reviewed journal articles selected as a result of searches in international scientometric databases, in particular Scopus, Web of Science and Google Scholar. The criteria for selecting sources were the relevance of publications (2021-2025), the direct relevance to the research topic, scientific novelty and practical significance for assessing the effectiveness of reactive, predictive and hybrid autoscaling strategies in stateful microservices.

The systems method assumed a comprehensive view of the research object as an integrated structure in which all elements were interconnected and interdependent. Its application in this study was conditioned by the need to consider the scaling process not in isolation but as a result of the interaction of several infrastructure levels – the orchestrator, service mesh, autoscaler and load balancer. Within the systems analysis, key links between these components were identified, which ensured the stability and performance of stateful microservices. The use of this method made it possible to reveal critical factors influencing scaling efficiency and to form a generalised logical model of interaction between infrastructure components. The use of the method of analytical generalisation was determined by the need to integrate disparate approaches to autoscaling presented in contemporary academic and applied sources. Within this method, typical decision-making scenarios in

the scaling process were analysed and common features among reactive, predictive and hybrid approaches were distinguished. The application of analytical generalisation made it possible to construct a generalised classification of scaling strategies that reflected the logic, strengths, and limitations in the context of stateful environments.

Comparative-structural methods involved the assessment of practical scaling implementations in leading cloud ecosystems (Amazon Web Services, Google Cloud Platform and Azure), taking into account the architectural features of each. Comparative analysis provided an opportunity to compare mechanisms for managing container state (stateful sets, persistent volumes, checkpointing) and to assess the impact on system stability. In turn, structural-functional analysis made it possible to trace the logic of interaction between infrastructure components and to determine the features of performance monitoring, load forecasting and balancing between nodes. The application of this method resulted in the identification of patterns in the choice of scaling strategies depending on Service Level Agreement parameters, which created the basis for forming generalised conclusions regarding the effectiveness of different scaling models.

Conceptual modelling methods ensured the formalisation of complex systemic dependencies through the construction of logical, structural and hierarchical models. The use in the study was driven by the need to generalise the results of the analytical and comparative stages and to present the process of choosing a scaling strategy as an integrated conceptual scheme. The application of the systems-hierarchy method made it possible to form a multilevel structure of criteria within which reactive, predictive and hybrid approaches were compared according to the type of workload, latency sensitivity and data-consistency requirements. As a result, conceptual modelling made it possible to integrate the disparate analysis results into a single formalised model that visually represented the decision-making logic and the relationship between the technical characteristics of the system and the effectiveness of the choice of scaling strategy. Analytical-predictive methods were justified by the need to forecast the effectiveness of scaling strategies in stateful microservices and to assess the potential for integrating intelligent autoscaling solutions based on machine learning. The use of analytical-predictive methods made it possible to extrapolate the identified patterns to prospective scenarios, to evaluate the impact of predictive strategies on resource optimisation and to determine the contribution to increasing the stability of stateful microservices.

RESULTS

Theoretical foundations of autoscaling in microservice architecture

Autoscaling in microservice systems acts as a key tool for maintaining adaptive performance under dynamically changing workloads. Theoretically, this process is based on the principles of self-organisation and feedback control, which allow the infrastructure to adapt its resources automatically to current and forecast needs without operator intervention. According to dynamic load control models, autoscaling is a mechanism that minimises the difference between the actual and target level of resource utilisation, ensuring stability and efficiency in the operation of cloud services. In a microservice architecture, each component of the system performs a narrowly specialised function, so scaling is carried out granularly, i.e., at the level of individual services rather than the whole application (Quattrocchi *et al.*, 2024). This creates conditions for local optimisation of performance but at the same time generates complex interaction dynamics between services, which requires theoretical modelling based on the principles of discrete control systems and distributed coherence. In the context of Kubernetes, autoscaling is implemented as a multilevel control system that combines monitoring, analytics and orchestration. The main theoretical foundation is the control loop, in which three components (observation, analysis, and action) interact cyclically, forming a mechanism of continuous adaptation. This model is consistent with the concept of cybernetic feedback control, where the system not only reacts to changes but also predicts future trends on the basis of historical data (which becomes the basis for the transition from reactive to predictive autoscaling) (Tabilo Alvarez & Ramírez-Correa, 2023). In cloud environments such as Amazon Web Services, Google Cloud Platform or Azure, similar mechanisms are integrated into global Service Level Objectives policies, which set permissible limits for stability and performance metrics.

Horizontal scaling consists in increasing the number of homogeneous service instances to distribute the load. In Kubernetes, this approach is implemented through the Horizontal Pod Autoscaler, which continuously monitors Central Processing Unit (CPU) metrics or custom metrics (for example, latency or the number of requests per second). Theoretically, horizontal scaling is based on the model of massive parallelism, where system performance is directly proportional to the number of available instances but is limited by Amdahl's Law. Its main advantage is linear scalability for stateless services, whereas the main challenge is the coordination of load balancing and inter-service dependencies. Vertical scaling, in contrast, focuses on optimising the parameters of each instance, i.e. increasing computing power (CPU, Random Access Memory (RAM), storage) without changing the number of pods. In Kubernetes, this approach is implemented in the form of the Vertical Pod Autoscaler, which uses statistical methods to assess average resource consumption and dynamically adjusts requests and limits. From a theoretical point of view, the Vertical Pod Autoscaler is a manifestation of the model of adaptive parameter control, within which the system reduces the imbalance between node capacity and current workload. The main drawback is that pod restarts may temporarily reduce service availability (Razavi *et al.*, 2024).

Hybrid scaling combines the advantages of both previous approaches and uses multilevel analytics to control

both the number of pods and the resource parameters simultaneously. Theoretically, it relies on the model of multiloop control, where one loop is responsible for reactive regulation and the other for predictive optimisation of resources. This approach is particularly effective in scenarios where there is high workload variability and system sensitivity to latency. Comparative analysis presented in Table 1 shows that the choice of scaling type should be context-dependent – based on the nature of the workload, latency sensitivity, consistency requirements and the capabilities of the Kubernetes infrastructure.

Table 1. Comparative characteristics of scaling types in Kubernetes microservice environments

Type of scaling	Theoretical model	Main purpose	Advantages	Disadvantages
Horizontal Pod Autoscaler	Parallelism model	Load distribution	High flexibility, simplicity	Not suitable for stateful
Vertical Pod Autoscaler	Adaptive parameter control	Optimisation of pod capacity	Efficient use of resources	Requires restart
Hybrid	Multiloop control	Performance balancing	High stability, adaptability	Implementation complexity

Source: created by the author on the basis of analysis of data from F. Almeida & B. Bálint (2024), K. Razavi *et al*. (2024)

As shown in the table, each type of scaling has its own theoretical basis, purpose, and trade-offs. Horizontal scaling is the simplest to implement and provides high flexibility, but it has limitations for stateful services, where state coordination between replicas complicates parallel operation. Vertical scaling uses resources efficiently but requires container restarts, which may lead to short-term system interruptions. Hybrid scaling is considered the optimal strategy for complex workloads, as it combines the adaptability of reactive solutions with the stability of predictive ones, although it requires complex orchestration and precise tuning of interaction parameters between control layers. Scaling stateful services constitutes a separate theoretical and applied challenge, as it violates the classical assumption of instance independence. In systems with internal state (databases, caches, message queues, streaming data stores), key issues become data consistency, availability, and fault tolerance. From a theoretical point of view, this means that the autoscaling process must be aligned with the constraints of Consistency, Availability, Partition tolerance (the CAP theorem), which proves the impossibility of simultaneously ensuring full consistency, availability, and partition tolerance. In Kubernetes, the management of stateful components is carried out through the StatefulSet object, which ensures persistent pod identity, a guaranteed order of start and termination, as well as stable data storage. Unlike Deployment for stateless components, StatefulSet preserves a pod → persistent volume map, which makes it possible to implement localised state coherence. However, theoretically, effective scaling of such systems requires state-synchronisation models that include replication, latency control and cluster-stability forecasting. Thus, autoscaling in stateful environments is not only resource management but a complex task of dynamically maintaining system invariants, where a scaling decision must take into account the topology of connections between nodes, the volume of data transferred and synchronisation delays (Kumari *et al*., 2023).

The theoretical foundations of autoscaling in microservice architecture presuppose a synthesis of concepts of feedback control, resource optimisation and coherent state management. For Kubernetes and similar systems, autoscaling appears as a cybernetic regulation system operating at the intersection of three axes – type of scaling, nature of service and observability metrics. In theoretical terms, this creates a basis for further differentiation between reactive and predictive approaches to scaling, which are considered the next level in the evolution of automated control systems in cloud environments.

Review of principles, algorithms, and architecture of reactive and predictive autoscaling models

Autoscaling models in cloud microservice architectures have evolved from simple reactive mechanisms to complex predictive systems capable of taking into account temporal patterns of load, correlations between components and user behaviour patterns. In theoretical terms, these models are based on different approaches to control – reactive, which involves a direct response to current system metrics, and predictive, focused on forecasting load changes using time-series prediction models or neural networks. The comparison makes it possible to reveal the fundamental principles of building adaptive control systems that underlie Kubernetes, Amazon Web Services Auto Scaling or Azure Autoscale.

Reactive scaling is the most widespread paradigm of automatic resource regulation in cloud environments. Theoretically, it relies on a feedback-control model, within which the system observes current metrics (for example, CPU utilisation, memory usage, latency, throughput) and performs scaling in response to deviations from predetermined thresholds (Chilakala *et al*., 2024). Mathematically, such scaling can be described in the form of a Proportional-Integral-Derivative (PID-controller) type regulator, which responds to the error between the desired and actual metric value. The advantage of reactive models lies in the implementation simplicity, deterministic behaviour and high predictability. However, the main drawback is response latency, which may lead to short-term overloads or

excessive resource use. In the context of Kubernetes, the implementation of such an approach is the Horizontal Pod Autoscaler, which regularly (for example, every 15 seconds) analyses the average workload of pods and increases or decreases the number according to the scaling formula (1):

$$N_{desired} = N_{current} \times \frac{U_{current}}{U_{target}}, \quad (1)$$

where $N_{desired}$ – the new number of pods; $U_{current}$ – the current average workload; U_{target} – the target value of the metric.

The formula is a clear example of the mathematical representation of the principle of operation of the reactive model: from the measured error to resource correction, and explains how the Horizontal Pod Autoscaler system transforms observed workload metrics into specific actions – changing the number of pods, which ensures the maintenance of the desired performance level. Thus, reactive scaling is a post factum correction of the system state that supports short-term stability but does not provide adaptive forecasting. Its application is appropriate in systems with stable or weakly variable workloads, where the risk of peak spikes is minimal.

Threshold models are the basic form of reactive scaling (Fig. 1). The principle lies in defining hard limits of permissible metric values: if the average CPU workload exceeds, for example, 80%, the system adds a new pod when the threshold is crossed. The advantage of threshold models lies in the low computational complexity and ease of parameterisation, which makes the mentioned models popular in industrial systems. However, these models are prone to oscillations – frequent changes in the number of instances when metrics fluctuate near boundary values. To mitigate this effect, hysteresis or moving averages are used.

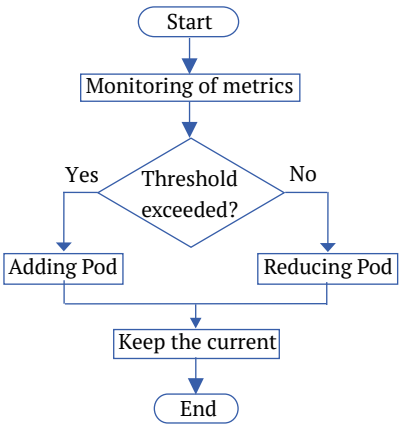

Figure 1. Logic of threshold scaling in a reactive model
Source: created by the author

According to the analytical model presented in the figure, the key advantage of this approach is responsiveness – the system quickly reacts to peak loads without the need for complex forecasting. At the same time, such logic has a number of structural limitations, in particular: a delay between the moment the threshold is exceeded and the actual scaling, the risk of performance fluctuations due to frequent metric oscillations, as well as increased resource consumption in cases of excessive scaling. Thus, the presented scheme reflects not only the mechanism of operation of the reactive model, but also the basis for the subsequent transition to predictive and hybrid strategies designed to reduce the delay between detection and system response.

The transition from reactive to predictive models has become a key point in the development of autoscaling theory. The main idea of the predictive approach lies in forecasting future workload on the basis of historical data and temporal patterns. Theoretically, these models are based on time-series forecasting methods or machine learning, which makes it possible to reduce reaction lag and prevent overloads. Among classical statistical methods, the following are used: AutoRegressive Integrated Moving Average (ARIMA) – for forecasting short-term workload changes on the basis of autocorrelation; Exponential Smoothing – for identifying trends and seasonal fluctuations in resource use; Kalman Filter – for adaptive correction of forecasts in real time (Table 2). More modern approaches are based on neural networks, in particular Long Short-Term Memory (LSTM) or Temporal Convolutional Networks (TCN), which are capable of capturing non-linear dependencies and long-term workload patterns. In such models, scaling decisions are based not only on current metrics but also on a probabilistic assessment of the future system state. For example, if the forecast workload exceeds current capacity by 20% over the next five minutes, the system may pre-emptively activate new pods, avoiding performance degradation.

As shown in the table, ARIMA models time series and is effective for short-term, linear processes but is not suitable for complex dynamic systems with non-linear dependencies. LSTM, based on deep learning, can identify non-linear workload patterns, providing high accuracy, but requires significant computational resources and lengthy training. Kalman Filter, in turn, combines stochastic principles of filtering and adaptive control, ensuring flexible response to changes in real time, but is sensitive to noise and errors in input metrics. Thus, the choice of forecasting model should be based on the nature of the workload: ARIMA – for stable and periodic systems, LSTM – for highly volatile and dynamic environments, and Kalman Filter – for scenarios where the speed of adaptation to metric changes in real time is critical. In real cloud environments, reactive and predictive mechanisms are often combined in a hybrid autoscaling architecture, where short-term fluctuations are handled reactively and medium- and long-term trends – predictively. Architecturally, such a system consists of three layers: monitoring and observability – collection of metrics, logs and tracing via Prometheus, Grafana, OpenTelemetry; analytical layer – metric processing, time-series modelling, workload forecasting; orchestration layer – implementation of decisions via the Kubernetes Application Programming Interface (API), Horizontal Pod Autoscaler/Vertical Pod Autoscaler controllers or custom webhooks (Fig. 2).

Table 2. Comparative characteristics of predictive autoscaling models in cloud environments

Type of model	Algorithmic basis	Main purpose	Advantages	Limitations
ARIMA	Linear time series	Short-term forecasting	Simplicity, interpretability	Does not work with nonlinear data
LSTM	Deep learning	Nonlinear patterns	High accuracy	High computational costs
Kalman Filter	Stochastic control	Real-time adaptability	Fast response	Sensitivity to data noise

Source: created by the author based on data analysis by H.S. Chilakala *et al.* (2024)

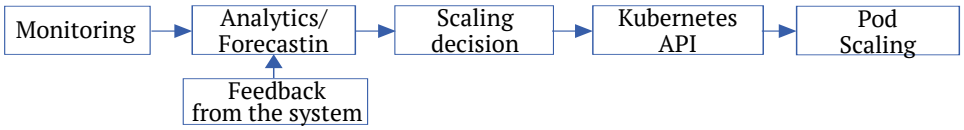

Figure 2. Architecture of a combined autoscaling approach
Source: created by the author on the basis of analysis of data from V. Rampérez *et al.* (2021)

This architecture corresponds to the theoretical model of adaptive control with a double loop, in which the outer loop performs long-term optimisation and the inner loop short-term stabilisation. In summary, it can be argued that the evolution of autoscaling models from reactive to predictive reflects the transition from deviation-based regulation to forecast-based regulation, i.e. from the classical cybernetic control model to an intelligent self-tuning system. Reactive models provide basic stability but are limited in dynamic environments; predictive models, due to the use of machine-learning methods and time-series models, form the basis of cognitive autoscaling capable of self-learning and resource optimisation in real time.

Theoretical analysis of performance trade-offs

In theoretical terms, the effectiveness of autoscaling systems in microservice architectures is always assessed through the prism of performance trade-offs. The term performance trade-offs refers to the balance between a number of interrelated parameters: latency, system stability, state consistency, resource efficiency and scaling cost. The application of different autoscaling models (reactive or predictive) determines which aspects of performance will be prioritised and where certain compromises are possible.

One of the priority parameters is scaling latency, i.e., the time from the moment overload is detected to the actual addition of new resources. In reactive models, this latency usually includes: the time taken to collect metrics, compute a decision based on thresholds, initiate scaling via the Kubernetes API and start a new pod. Theoretically, even with optimal configuration, reactive systems demonstrate higher latency, which may lead to short-term overload and deterioration of Quality of Service. In predictive models, scaling latency is compensated by anticipatory decision-making, when new resources are activated before critical workloads are reached. However, forecast accuracy directly affects the effectiveness of this approach: inaccurate forecasts may lead to unjustified resource allocation, which increases the cost of scaling without real benefit for latency.

Stability is defined as the absence of frequent and sharp fluctuations in the number of pods or resource use. In reactive systems, high sensitivity to thresholds often causes "flipping": a short-term threshold exceedance leads to the addition of pods, and a decrease in workload to the removal almost immediately afterwards. Such behaviour creates performance oscillations and may worsen the stability of stateful services, especially when state consistency between replicas is important. Predictive autoscaling, by contrast, makes it possible to smooth out fluctuations, since scaling decisions are made on the basis of forecast trends rather than instantaneous spikes. However, incorrect model configuration or failure to take sudden peaks into account may lead to systematic instability, for example when the system allocates resources in advance, but the peak does not occur (Kanwal *et al.*, 2024). For stateful microservices, a critical characteristic is state consistency between replicas, since any changes in the scaling configuration – both scale-out (adding pods) and scale-in (removal) – may disrupt data synchronisation. In the absence of effective replication mechanisms such as Raft or Paxos, the system may temporarily lose consistency, which is particularly dangerous for transactional or analytical services. Within reactive autoscaling, such problems occur more often, since scaling takes place post factum, i.e. after an increase in workload has been detected. At this point, new pods have not yet had time to fully synchronise state with existing instances, which leads to asynchronous state divergence and potential data-consistency errors. Predictive autoscaling, on the other hand, makes it possible to plan scaling in advance, ensuring data replication before the system reaches peak workload. This reduces the risk of state divergence but requires additional computing resources and time costs for preliminary synchronisation. Thus, there is a clear trade-off between response speed and data consistency that must be taken into account when designing the system.

Another aspect is resource efficiency and scaling cost. In the reactive approach, the system often operates in a state of under- or over-provisioning, which leads to unstable performance and increased financial costs. Due to the delay in responding to workload, resources are added with a lag, and after the activity decreases the resources remain unused, generating economic losses – especially in cloud environments where payment is made for allocated resources. The predictive approach, by contrast, makes it

possible to optimise resource provisioning, activating pods only when necessary. However, forecast accuracy becomes a critical factor: overly conservative or inaccurate models may lead to suboptimal reservation and, accordingly, additional costs (Maliakel *et al.*, 2025). Overall, the effectiveness of predictive scaling is determined by the balance between model accuracy, performance stability and economic feasibility. The reactive approach prevails in scenarios where rapid response is required, but temporary instability is acceptable, whereas the predictive approach is preferable in environments that require stability, state consistency and optimal resource use. Table 3 presents a structured comparison of the characteristics of each approach, taking into account the impact on latency, stability, state consistency, resource efficiency and financial costs. This format makes it possible to clearly trace in which conditions one approach shows advantages over the other and where trade-offs arise between accuracy, reaction speed and economic feasibility.

Table 3. Comparative characteristics of reactive and predictive autoscaling for stateful microservices

Performance aspect	Reactive autoscaling	Predictive autoscaling	Theoretical trade-off
Latency	High due to post factum response	Reduced thanks to anticipatory forecasting	Forecast accuracy vs. reaction speed
Stability	Prone to oscillations and "flipping"	Higher stability due to trend smoothing	Flexibility of response vs. stability
State consistency	Possible state divergence during scaling	Planning of scaling with regard to replication	Synchronisation time vs. scaling speed
Resource efficiency	Often under- or overprovisioned	Higher with accurate forecasting	Forecast accuracy vs. resource savings
Scaling cost	Higher due to excessive or delayed resource allocation	Optimised but dependent on forecast accuracy	Forecasting vs. financial costs

Source: created by the author on the basis of analysis of data from J. Karol Santos Nunes *et al.* (2024), A. Khaleq & I. Ra (2021)

The results obtained indicate that reactive autoscaling remains more flexible in unpredictable conditions but is characterised by increased latency and a risk of destabilising state. Predictive autoscaling, in turn, provides better stability and resource-use efficiency, especially for stateful microservices, but requires accurate forecasting models and additional computational costs. Thus, the choice of optimal scaling strategy should be based on a balance between consistency requirements, Service Level Agreement, cost constraints and workload dynamics of the system.

Conceptual model for choosing a scaling strategy for stateful environments

In microservice systems with internal state, the choice of an optimal scaling strategy is a complex task that requires balancing between response speed, system stability, state consistency and efficient use of resources. Theoretically, this is linked to performance trade-offs, where none of the autoscaling models – reactive or predictive – can optimise all parameters simultaneously. The choice of approach depends on the characteristics of the system: type of workload, permissible latency, state-consistency requirements, resource availability and the duration of workload trends.

Reactive autoscaling is appropriate in environments with stable or moderately variable workloads where short-term deviations from target metrics are acceptable and response latency is not critical. Theoretically, this approach relies on a feedback-control model, in which scaling takes place post factum, allowing rapid reaction to local changes but not preventing peak overloads. Predictive autoscaling, by contrast, presupposes anticipatory resource management on the basis of forecasts of future workload. This approach is optimal for environments with highly variable workload, critical latency and strict requirements for state consistency. The use of time-series methods, ARIMA models, the Kalman Filter or neural networks (LSTM, TCN) makes it possible to activate resources in advance, reducing the risk of performance degradation and consistency violations. Hybrid autoscaling combines the advantages of both approaches and implements a multi-loop architecture: the inner loop provides short-term reaction to local changes, while the outer loop forecasts medium- and long-term trends. This approach is particularly effective in stateful environments with mixed workloads, where stability, consistency, and resource savings must be maintained simultaneously. Table 4 shows the key criteria for choosing an autoscaling strategy for stateful environments.

Table 4. Comparison of scaling strategies

Criterion	Reactive autoscaling	Predictive autoscaling	Hybrid autoscaling	Trade-off/notes
Type of workload	Stable or moderately variable	Highly variable, seasonal spikes	Any, especially medium dynamics	Reactive quickly reacts to local changes, predictive anticipates peaks
Latency tolerance	Medium/high latency tolerance	Low tolerance of latency	Configurable dynamically	Reactive may miss peaks, predictive acts in advance
Consistency requirements	Short-term inconsistencies are acceptable	High criticality of consistency	Balance between speed and consistency	Hybrid makes it possible to stabilise state and react to local fluctuations
Resource efficiency	Often under- or over-provisioning	Optimisation given accurate forecasting	Medium – depends on configuration	Reactive is less economical, predictive precise, hybrid combines both

Continued Table 4.

Criterion	Reactive autoscaling	Predictive autoscaling	Hybrid autoscaling	Trade-off/notes
Implementation complexity	Low	High (requires ML/ analytics)	Very high	Hybrid requires dual-loop architecture and monitoring
Scaling cost	High due to delayed or excessive allocation	Depends on forecast accuracy	Optimised through a combined approach	Trade-off between finance and productivity

Source: created by the author on the basis of analysis of data from D. Zou *et al.* (2024)

Table 4 summarises the key characteristics of the three scaling approaches – reactive, predictive and hybrid – in stateful microservices. Overall, reactive autoscaling provides rapid reaction to current workload, but is less effective with highly variable peaks and may allow short-term state inconsistencies. The predictive approach anticipates workload in advance, improving stability and consistency, but requires accurate forecasting and additional resources. Hybrid autoscaling combines the advantages of both approaches, allowing simultaneous reaction to local changes and forecasting of peaks, although its implementation is more complex and requires greater analytical support. Thus, the choice of strategy depends on workload dynamics, latency and consistency requirements, as well as on the system's resource and financial constraints. Different parameters, such as workload variability, latency criticality, data-consistency requirements and resource availability, determine which approach – reactive, predictive or hybrid – will be more appropriate. Table 5 summarises these dependencies, demonstrating the trade-off between performance, stability and resource efficiency.

Table 5. Influence of stateful-system characteristics on the choice of autoscaling strategy

System characteristic	Reactive	Predictive	Hybrid	Comment
Workload	Stable	Dynamic, spikes	Medium/mixed	Choice depends on load variability
Latency criticality	Low	High	Medium/depends on scenario	If latency is critical, predictive has the advantage
Consistency requirements	Moderate	High	High + reactive smoothing	For stateful services, it is important to plan replication
Resource/budget availability	Limited	Moderate	Sufficient for dual-loop	Hybrid requires more resources for analytics and synchronisation
Trend duration	Short	Long or seasonal	Any	Hybrid allows a combination of short- and long-term strategies

Source: created by the author on the basis of analysis of data from T. Betti Pillippuge *et al.* (2025)

Table 5 shows that reactive autoscaling is effective for stable systems with low latency criticality and moderate consistency requirements, but does not provide high stability under sharp workload spikes. The predictive approach demonstrates advantages in systems with dynamic or seasonal workload and high requirements for latency and state consistency, but demands significant resources for accurate forecasting. Hybrid autoscaling makes it possible to combine reactive speed and anticipatory planning, which makes it universal for systems with medium or mixed workload dynamics and long-term trends, although its implementation requires dual-loop architecture and additional monitoring. Overall, the choice of strategy should be based on a balance between performance, consistency and resource-use efficiency.

Choosing an optimal scaling strategy for stateful microservice environments requires a systematic analysis of a set of interrelated factors that affect performance, state consistency and resource-use efficiency. A hierarchical model of factors makes it possible to structure the key decision-making criteria, from general requirements for throughput and latency to specific requirements for consistency and resource availability. Such a conceptual approach provides a scientifically grounded basis for assessing trade-offs between response speed, system stability and economic efficiency, which is critical when choosing between reactive, predictive and hybrid autoscaling models (Rao, 2021). Schematically, the conceptual model can be presented as a hierarchy of factors that determine the optimal scaling approach (Fig. 3).

On the basis of this hierarchy, the decision-making process can be formalised as follows: if most characteristics point to stable workload and acceptable latency, reactive is chosen; if consistency and latency are critical, predictive is chosen; and for mixed scenarios with sufficient resources, hybrid is chosen. The conceptual model of choosing a scaling strategy shows that there is no universal approach for stateful microservices. The choice between reactive, predictive and hybrid approaches is a compromise and depends on workload characteristics, latency and consistency requirements, resource availability and trend duration. Integration of a dual-loop architecture in hybrid autoscaling makes it possible to maintain stability, optimal resource use and state consistency simultaneously, which makes it the most flexible and promising option for complex stateful environments.

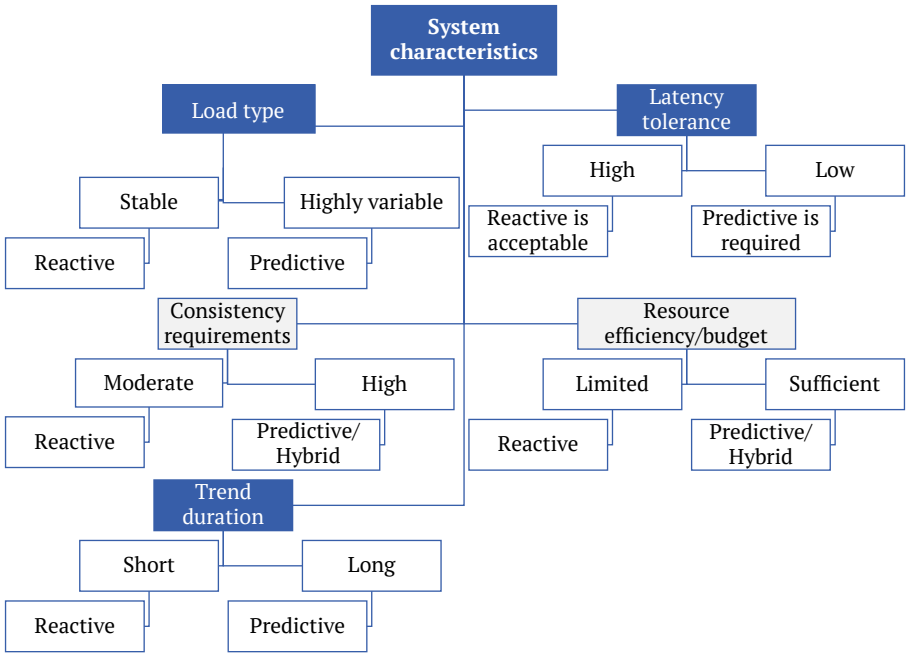

Figure 3. Hierarchy of factors for strategy selection

Source: created by the author

DISCUSSION

The review of scientific works shows a gradual shift of focus from localised technical solutions (simple horizontal/vertical scaling) to complex, context-oriented approaches: self-adaptive architectures, vertically oriented resizing algorithms, solutions for serverless/ETL processes and assessments of the impact of frameworks on scalability. For example, S. Boyapati & C. Szabo (2022) present a detailed case study of self-tuning microservice architectures, in which the authors focus on mechanisms of dynamic adaptation to workload changes in real time. The results of the study confirmed the effectiveness of reactive approaches for environments with highly dynamic requests but revealed a significant dependence on the accuracy of metric monitoring. In the present study, similar conclusions were obtained regarding the high sensitivity of the system to threshold values in the reactive model. Methods of optimising this sensitivity through combination with elements of predictive scaling were also proposed, providing greater stability during peak loads. In addition, the findings of Yu. Palamarchuk (2022) emphasise that the effectiveness of microservice-based systems is strongly dependent on the architectural decomposition of services and the precision of interaction patterns between components. It is highlighted that well-structured microservice architectures enable more predictable scaling behaviour and support improved resource utilisation, particularly in environments where dynamic load variations are present. These observations align with the patterns identified in the present study, where architectural clarity and service modularity are shown to influence the performance and stability of both reactive and predictive autoscaling approaches in stateful systems. The work further reinforces the notion that autoscaling mechanisms cannot be fully effective without a coherent underlying architectural framework, which remains consistent with the broader conclusions of the current research regarding the context-dependent nature of scaling strategies. The publication by A. Pavlenko *et al.* (2024) presents the CaaSPER algorithm, aimed at vertical autoscaling of monolithic applications in a containerised environment. The authors showed that even for non-microservice architectures, the vertical approach can significantly reduce downtime and optimise resource use. These results partially align with the current observations: in stateful environments vertical scaling is appropriate only if mechanisms of state coordination are in place, whereas in the work of A. Pavlenko *et al*. the main focus was on isolated monolithic applications. This difference explains the divergence in conclusions regarding the effectiveness of the Vertical Pod Autoscaler – in the present study, its application is limited by data-consistency issues.

In the study by D. Rodrigues *et al*. (2025), the trade-off between execution latency and computation cost in serverless environments is examined, where scaling is performed on the basis of optimisation rules for ETL processes. The authors formalised a trade-off model for decision-making, which shows how scaling policies affect performance and costs. This approach is close to the theoretical analysis carried out here of trade-offs between latency, stability, and costs. At the same time, the concept of adaptive selection of a scaling policy depending on the dominant metric confirms the validity of the proposed model of hybrid load control. M. Islam (2025) reveals the impact of data-driven web frameworks on the scalability and performance of modern enterprise systems. The author found that framework features – in particular the ORM layer, asynchronous request

handling and caching mechanisms – directly determine the behaviour of autoscaler algorithms. This conclusion is consistent with the results obtained here regarding the role of RAM and CPU as the main triggers for scaling and the importance of taking into account characteristics of the software stack when choosing the Horizontal Pod Autoscaler or Vertical Pod Autoscaler.

B. Tutuncuoglu (2025) proposed the "Phoenix Stack" architecture for self-healing microservice web applications in real time, where the key elements are automatic failure detection and localised state recovery. The author showed that integrating self-healing mechanisms significantly increases availability and reduces recovery time in the event of component failures, which directly correlates with the current conclusion about the need to consider fault tolerance when designing scaling strategies for stateful services. At the same time, the "Phoenix Stack" strongly emphasises local recovery and restart patterns. In turn, J. Prodanov *et al.* (2025) investigated an in-place scaling mechanism for edge-cloud environments, implemented through multi-agent reinforcement learning (MARL), where agents coordinate decisions on local increase/decrease of replicas, taking into account network latency and energy constraints. The results showed that the MARL approach significantly improved latency and energy-consumption balance compared with static policies; this correlates with the present conclusions about the promise of adaptive, intelligent controllers for hybrid autoscaling. However, the application of MARL by the authors is oriented towards edge-class scenarios with strict resource constraints and a different network topology, so direct transfer of these results to traditional stateful Kubernetes clusters requires adjustments. I. Vasireddy *et al*. (2025) presented a non-linear regression predictive model of replicas combined with the ORLE algorithm to improve the scalability, energy efficiency and economy of Kubernetes clusters. In the experiments, the authors demonstrated a reduction in redundant replicas and improved CPU/RAM use compared with the standard Horizontal Pod Autoscaler, which directly correlates with the current conclusion that more accurate predictive models increase the efficiency of resource provisioning. The difference is that I. Vasireddy *et al*. focused on specific regression models and an energy metric, whereas the present study considered a broader set of trade-offs (latency, consistency, costs). Thus, these results complement but do not fully replace the recommendations proposed here regarding strategy selection.

M. Yang *et al.* (2025) conducted a systematic review on the optimisation of full-stack microservice architecture and identified research gaps in the integration of network, container, and application levels for global performance optimisation. The review confirmed that comprehensive, cross-stack approaches (combining optimisation at the code, container, and orchestration levels) are the most promising for achieving both cost-efficiency and low latency, which corresponds to the current recommendation in favour of hybrid and multi-layer strategies. At the same time, M. Yang *et al.* emphasised that there is still a lack of large-scale empirical research under production conditions, which reinforces the present limitation due to the absence of full experimental validation and indicates directions for further research. In the study by A. Chaudhari & P. Charate (2025), an approach to dynamic orchestration of microservices in multicloud environments is proposed, aimed at minimising latency in applications with high performance requirements. The authors demonstrated that combining container orchestration with adaptive routing algorithms can significantly reduce latency between cloud nodes and increase resource-use efficiency. This result partially corresponds to the current conclusions on the importance of adaptive scaling strategies for stateful systems, but the present study focuses on intra-cluster optimisation mechanisms, which explains the difference in focus and practical recommendations.

In turn, the dissertation research by A. Baarzi (2021) analyses the effectiveness of service deployment in public clouds from the perspective of the cost-performance-security ratio. The author proved that optimising service deployments by combining automated scaling and security policies can reduce operational costs without significant loss of performance. Compared with the present study, the work of A. Baarzi is more oriented towards the macro-level of economic parameters and overall architectural trade-offs, while the current analysis focuses on the technical detail of reactive and predictive scaling models. At the same time, C. Oleti (2023) considered the construction of secure scalable microservice architectures based on Spring Boot and Amazon Web Services, paying particular attention to the integration of artificial-intelligence modules in corporate systems. The author showed that the use of AI components in orchestration and load-distribution processes significantly improves request-processing efficiency and allows peak loads to be predicted with high accuracy. This directly correlates with the present conclusions regarding the feasibility of the predictive approach and hybrid strategies that combine reactive mechanisms with intelligent forecasting. However, the work of C. Oleti focuses mainly on the corporate context (enterprise-scale AI), whereas the present study covers more universal scenarios of containerised stateful services, which explains the difference in the scope of application and the target efficiency metrics.

Thus, the synthesis of the analysed sources has confirmed the current conclusions regarding the context-dependence of scaling-strategy choice and the feasibility of using hybrid models that combine reactive and predictive principles. Differences in conclusions between the works are primarily due to differences in research subject (monoliths, microservices, serverless), the level of empirical verification and the specificity of the metrics used (latency, cost, uptime). The present study extends the existing theoretical basis by integrating these approaches into a generalised decision-making model for stateful environments, which forms a foundation for further experiments in the field of adaptive scaling.

CONCLUSIONS

Autoscaling in microservice systems provides adaptive performance through a feedback loop, while the choice of strategy depends on the nature of the workload, latency sensitivity and consistency requirements. Horizontal scaling provides flexibility for stateless services, vertical scaling optimises pod resources, and hybrid scaling combines both approaches for complex scenarios. Reactive autoscaling reacts quickly to changes but may lead to state inconsistencies and inefficient resource use, whereas predictive autoscaling forecasts needs and increases stability, requiring additional computing resources. The choice of an optimal strategy is always a compromise and context-dependent.

The study has shown that autoscaling models in cloud microservice architectures have evolved from simple reactive mechanisms to complex predictive systems capable of taking into account temporal workload patterns and user behaviour. Reactive models implemented via the Horizontal Pod Autoscaler provide simplicity and predictability, but are limited in reaction speed and resource-use efficiency. Predictive models based on time-series methods (ARIMA, Exponential Smoothing, Kalman Filter) or deep learning (LSTM, TCN) make it possible to reduce reaction lag, predict workload peaks and increase system stability, although such models require significant computing resources and more complex implementation. Hybrid approaches combine the advantages of both strategies, providing short-term stabilisation and long-term forecasting through a dual-loop adaptive-control architecture. Overall, the choice of an optimal autoscaling strategy is context-dependent and involves balancing between reaction speed, stability, state consistency, resource efficiency and implementation complexity.

The theoretical analysis of performance trade-offs has shown that the effectiveness of autoscaling in stateful microservice architectures is determined by the balance between reaction latency, system stability, state consistency, resource-use efficiency and financial costs. Reactive autoscaling provides rapid adaptation to unpredictable workload changes, but is characterised by high latency, performance fluctuations and a risk of replica-state inconsistencies. Predictive autoscaling reduces scaling latency and increases stability and resource provisioning efficiency through anticipatory forecasting, but requires accurate prediction models and additional computing resources. Hybrid strategies combine the advantages of both approaches, making it possible to ensure short-term flexibility and long-term stability simultaneously, which makes the choice of optimal model context-dependent and determined by workload dynamics, Service Level Agreement and financial constraints.

The analysis of autoscaling models in stateful microservice architectures has shown that system effectiveness is defined by the trade-off between reaction latency, stability, state consistency, resource efficiency and costs. Reactive autoscaling provides rapid adaptation to unpredictable workloads but is accompanied by high latency and a risk of replica-state inconsistencies. Predictive autoscaling increases stability and optimises resource use owing to anticipatory forecasting, but requires accurate models and additional computing resources. Hybrid strategies integrate the advantages of both approaches, providing flexibility and stability at the same time, and the choice of optimal model depends on workload dynamics, Service Level Agreement requirements and financial constraints.

The limitations of the study lie in the theoretical nature of the analysis, which was based on conceptual models of reactive, predictive and hybrid scaling without experimental validation in real stateful microservice environments, as well as on a limited set of forecasting algorithms and the absence of a detailed examination of economic and cloud-provider aspects of implementation. Prospects for further research include empirical testing of strategies in different cloud infrastructures, integration of modern machine-learning methods to increase forecasting accuracy and the development of dynamic combined scaling models that take into account Service Level Agreement, latency, consistency and financial efficiency.

ACKNOWLEDGEMENTS

None.

FUNDING

None.

CONFLICT OF INTEREST

None.

REFERENCES

[1] Almeida, F., & Bálint, B. (2024). Approaches for hybrid scaling of agile in the IT industry: A systematic literature review and research agenda. *Information*, 15(10), article number 592. doi: 10.3390/info15100592.

[2] Baarzi, A.F. (2021). *Efficient service deployment on public cloud: A cost, performance, and security perspective*. University Park: Pennsylvania State University.

[3] Babenko, T., Kolesnikova, K., Abramkina, O., & Vitulyova, Y. (2025). Automated OSINT techniques for digital asset discovery and cyber risk assessment. *Computers,* 14(10), article number 430. doi: 10.3390/computers14100430.

[4] Betti Pillippuge, T.L., Khan, Z., & Munir, K. (2025). Horizontal autoscaling of virtual machines in hybrid cloud infrastructures: Current status, challenges, and opportunities. *Encyclopedia*, 5(1), article number 37. doi: 10.3390/encyclopedia5010037.

[5] Boyapati, S., & Szabo, C. (2022). Self-adaptation in microservice architectures: A case study. In *Proceedings of the 26th international conference on engineering of complex computer systems* (pp. 42-51). Hiroshima: IEEE. doi: 10.1109/ICECCS54210.2022.00014.

[6] Chaudhari, A.V., & Charate, P.A. (2025). Dynamic orchestration of microservices across multi-cloud environments for low-latency applications. *International Research Journal of Advanced Engineering and Science*, 10(2), 165-172.
[7] Chilakala, H.S., Preeti, N., Marri, S.P., & Murali, K. (2024). Kalman integration with ARIMA & LSTM. In A. Kumar, G. Sharma, A. Sharma, P. Chopra & P. Rattan (Eds.), *Advances in networks, intelligence and computing* (pp. 562-569). London: CRC Press. doi: 10.1201/9781003430421.
[8] Guntupalli, B., & Vamshi, S. (2023). Designing microservices that handle high-volume data loads. *International Journal of AI, BigData, Computational and Management Studies*, 4(4), 76-87. doi: 10.63282/3050-9416.IJAIBDCMS-V4I4P109.
[9] Islam, M. (2025). The impact of data-driven web frameworks on performance and scalability of US enterprise applications. *International Journal of Business and Economics Insights*, 5(3), 523-558. doi: 10.63125/f07n4p12.
[10] Kanwal, N., Khoraminia, F., Kiraz, U., Mosquera-Zamudio, A., Monteagudo, C., Janssen, E., Zuiverloon, T., Rong, C., & Engan, K. (2024). Equipping computational pathology systems with artifact processing pipelines: A showcase for computation and performance trade-offs. *BMC Medical Informatics and Decision Making*, 24(1), article number 288. doi: 10.1186/s12911-024-02676-z.
[11] Karol Santos Nunes, J.P., Nejati, S., Sabetzadeh, M., & Nakagawa, E.Y. (2024). Self-adaptive, requirements-driven autoscaling of microservices. In L. Baresi (Ed.), *Proceedings of the 19th international symposium on software engineering for adaptive and self-managing systems* (pp. 168-174). New York: Association for Computing Machinery. doi: 10.1145/3643915.3644094.
[12] Khaleq, A., & Ra, I. (2021). Intelligent autoscaling of microservices in the cloud for real-time applications. *IEEE Access*, 9, 35464-35476. doi: 10.1109/ACCESS.2021.3061890.
[13] Kumari, A., Anand, A.K., & Sahoo, B. (2023). From stateless to stateful: a comparative analysis of stateful serverless computing frameworks. In R. Tiwari, M. Saraswat & M. Pavone (Eds.), *International conference on computational intelligence* (pp. 223-237). Singapore: Springer. doi: 10.1007/978-981-97-3526-6_19.
[14] Maliakel, P.J., Ilager, S., & Brandic, I. (2025). Investigating energy efficiency and performance trade-offs in LLM inference across tasks and DVFS settings. *arXiv*. doi: 10.48550/arXiv.2501.08219.
[15] Mekki, M., Ksentini, A., & Meliani, A.E. (2025). Resiliency focused proactive lifecycle management for stateful microservices in multi-cluster containerized environments. *Computer Communications*, 236, article number 108111. doi: 10.1016/j.comcom.2025.108111.
[16] Meng, C., Song, S., Tong, H., Pan, M., & Yu, Y. (2023). Deepscaler: Holistic autoscaling for microservices based on spatiotemporal GNN with adaptive graph learning. In *Proceedings of the 38th IEEE/ACM international conference on automated software engineering* (pp. 53-65). Luxemburg: IEEE. doi: 10.1109/ASE56229.2023.00038.
[17] Oleti, C.S. (2023). Enterprise AI at scale: Architecting secure microservices with spring boot and AWS. *International Journal of Research in Computer Applications and Information Technology*, 6(1), 133-154. doi: 10.34218/IJRCAIT_06_01_011.
[18] Palamarchuk, Yu. (2022). Methods of building microservice architecture of e-learning systems. *Information Technologies and Computer Engineering*, 19(2), 43-54. doi: 10.31649/1999-9941-2022-53-1-43-54
[19] Pavlenko, A., Cahoon, J., Zhu, Y., Kroth, B., Nelson, M., Carter, A., Liao, D., Wright, T., Camacho-Rodriguez, J., & Saur, K. (2024). Vertically autoscaling monolithic applications with CaaSPER: Scalable container-a s-a-service performance enhanced resizing algorithm for the cloud. In P. Barcelo & N. Sanchez-Pi (Eds.), *Companion of the 2024 international conference on management of data* (pp. 241-254). New York: Association for Computing Machinery. doi: 10.1145/3626246.3653378.
[20] Prasad, T.A. (2023). AI-driven predictive scaling for performance optimization in cloud-native architectures. *Journal of Electrical Systems*, 19(4), 607-617.
[21] Prodanov, J., Bertalanič, B., Fortuna, C., Chou, S.K., Jurič, M.B., Sanchez-Iborra, R., & Hribar, J. (2025). Multi-agent reinforcement learning-based in-place scaling engine for edge-cloud systems. In *Proceedings of the 18th international conference on cloud computing* (pp. 32-42). Helsinki: IEEE. doi: 10.1109/CLOUD67622.2025.00014.
[22] Quattrocchi, G., Incerto, E., Pinciroli, R., Trubiani, C., & Baresi, L. (2024). Autoscaling solutions for cloud applications under dynamic workloads. *IEEE Transactions on Services Computing*, 17(3), 804-820. doi: 10.1109/TSC.2024.3354062.
[23] Rampérez, V., Soriano, J., Lizcano, D., & Lara, J.A. (2021). FLAS: A combination of proactive and reactive auto-scaling architecture for distributed services. *Future Generation Computer Systems*, 118, 56-72. doi: 10.1016/j.future.2020.12.025.
[24] Rao, A. (2021). Architectural trade-offs between stateless and stateful microservices in large-scale cloud systems. *International Journal of Innovation Studies*, 5(1), 135-141.
[25] Razavi, K., Salmani, M., Mühlhäuser, M., Koldehofe, B., & Wang, L. (2024). A tale of two scales: Reconciling horizontal and vertical scaling for inference serving systems. *arXiv*. doi: 10.48550/arXiv.2407.14843.
[26] Rodrigues, S., Rosas, M., Delbianco, N., Cláudia, C., Peterson, B., & Grácio, M.C. (2025). *Latency vs. cost trade-offs in serverless ETL: A decision-theoretic framework for architecture design.* Retrieved from https://www.researchgate.net/publication/393003364_Latency_vs_Cost_Trade-offs_in_Serverless_ETL_A_Decision-Theoretic_Framework_for_Architecture_Design.

[27] Romanov, O., Mankivskyi, V., Romanov, A., Nesterenko, M., & Pidpaly, O. (2024). Performance assessing of dynamic scaling of containerized applications when using kubernetes cluster autoscaler and auto scaling group together. In M. Ilchenko, L. Uryvsky & L. Globa (Eds.), *Advanced smart information and communication technology and systems* (pp. 111-136). Cham: Springer. doi: 10.1007/978-3-031-94799-5_7.
[28] Tabilo Alvarez, J., & Ramírez-Correa, P. (2023). A brief review of systems, cybernetics, and complexity. *Complexity*, 2023(1), article number 8205320. doi: 10.1155/2023/8205320.
[29] Thompson, M.A., Carter, J.L., Nguyen, D.R., Bennett, S.K., Miller, R.J., & James, A. (2023). *Intelligent workload balancing across multi-cloud providers*. Retrieved from https://tinyurl.com/4x7sebkh.
[30] Tutuncuoglu, B.T. (2025). *Phoenix stack: A self-healing microservice architecture for real-time web applications*. Retrieved from https://papers.ssrn.com/sol3/papers.cfm?abstract_id=5343101.
[31] Varzhanskiy, I., Melnychenko, A., Naiev, S., & Kapustynskyi, O. (2025). Foresight and active forecast potential for improving defense capability. *European Journal of Interdisciplinary Studies*, 17(1). doi: 10.24818/ejis.2025.01.
[32] Vasireddy, I., Wankar, R., & Chillarige, R. (2025). Improving scalability, energy efficiency, and cost-effectiveness in Kubernetes clusters using a nonlinear regression-based predictive replica model and ORLE algorithm. *Egyptian Informatics Journal*, 31, article number 100732. doi: 10.1016/j.eij.2025.100732.
[33] Yang, M., Wang, X., Cai, B., Wang, Y., & Guo, Y. (2025). Full stack optimization of microservice architecture: Systematic review and research opportunity. *Cluster Computing*, 28, article number 1005. doi: 10.1007/s10586-025-05690-6.
[34] Zou, D., Lu, W., Zhu, Z., Lu, X., Zhou, J., Wang, X., Liu, K., Sun, R., & Wang, H. (2024). OptScaler: A collaborative framework for robust autoscaling in the cloud. *Proceedings of the VLDB Endowment*, 17(12), 4090-4103. doi: 10.14778/3685800.3685829.

Христина Терлецька
Бакалавр
Національний університет «Львівська політехніка»
79000, вул. Степана Бандери, 12, м. Львів, Україна
https://orcid.org/0009-0002-7302-2625

Компроміси продуктивності між прогнозним та реактивним автомасштабуванням у станозалежних мікросервісах

Анотація. Актуальність дослідження компромісів між реактивним та прогнозним автоматичним масштабуванням станозалежних мікросервісів зумовлена необхідністю підвищити стабільність, продуктивність ресурсів і надійність хмарних систем у динамічних умовах. Метою дослідження було оцінити ефективність реактивних, прогнозних та гібридних стратегій автоматичного масштабування у станозалежних мікросервісних архітектурах. Методологічна основа дослідження спиралася на порівняльний, контентний, системний і структурно-функціональний аналізи, а також на моделювання, що дозволило комплексно дослідити продуктивність і стабільність різних підходів до автоматичного масштабування у станозалежних мікросервісах. Теоретичний аналіз показав, що автоматичне масштабування в мікросервісних системах забезпечує адаптивну продуктивність, а вибір стратегії залежить від характеру навантаження, чутливості до затримок та вимог до консистентності. Огляд моделей продемонстрував, що реактивне автоматичне масштабування швидко реагує на зміни, але супроводжується високою латентністю, коливанням продуктивності та ризиком розривів стану реплік, тоді як прогнозне автоматичне масштабування знижує затримку масштабування та підвищує стабільність і ефективність ресурсів, проте потребує точних моделей і додаткових обчислювальних ресурсів. Аналіз результатів виявив, що гібридні стратегії інтегрують переваги обох підходів, забезпечуючи одночасно гнучкість і стабільність, а ефективність систем визначається компромісом між затримкою, стабільністю, узгодженістю стану, ефективністю ресурсів і фінансовими витратами. Результати дослідження можуть бути корисними розробникам і архітекторам хмарних мікросервісних систем, IT-інженерам і DevOps-командам для оптимізації стратегій масштабування, підвищення стабільності сервісів і ефективного використання ресурсів у різних хмарних середовищах

Ключові слова: Kubernetes; оптимізація ресурсів; продуктивність хмар; динамічний розподіл ресурсів; стабільність системи; метрики спостереження