ІНФОКОМУНІКАЦІЙНІ ТЕХНОЛОГІЇ ТА ЕЛЕКТРОННА ІНЖЕНЕРІЯ
INFORMATION AND COMMUNICATION TECHNOLOGIES, ELECTRONIC ENGINEERING

№ 5 (2), 2025

ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНІ СИСТЕМИ ТА ТЕХНОЛОГІЇ

https://doi.org/10.23939/ictee2025.02.

METHOD FOR WORKLOAD-BASED SELECTION OF LIGHTWEIGHT PREDICTION MODELS IN MICROSERVICE AUTOSCALING

M. Klymash [ORCID: 0000-0002-1166-4182], **K. Morhoiev** [ORCID: 0009-0009-5347-6022]

Lviv Polytechnic National University, 12, S. Bandery str., Lviv, 79013, Ukraine

Corresponding author: K. Morhoiev (e-mail: kostiantyn.y.morhoiev@lpnu.ua)

(Received 01 July 2025)

The paper investigates the feasibility of using lightweight predictive models for proactive microservice autoscaling, addressing the limitations of the alternative approaches: reactive threshold-based scaling, causing potential delays in resource adjustment and response time, and the deep learning models, such as LSTM, requiring high computational resource allocation. Using Alibaba Cluster Trace dataset, microservice workloads are analyzed and classified into four distinct categories (Stable, Periodic, Spiky, Mixed) based on the coefficient of variation and peak-to-mean ratio. Coming from the considerations of simplicity in implementation, low level of computational complexity, and covering main methodological categories, six forecasting methods were selected for evaluation: simple moving average (SMA), exponential moving average (EMA), Holt-Winters smoothing, Kalman filter, autoregressive integrated moving average (ARIMA), and percentile-based estimation. Each method is tested for different forecast horizons in both vertical and horizontal scaling scenarios. The evaluation criteria were forecast accuracy (RMSE, MAE, MAPE), computational efficiency (execution time, amount of memory used), and model suitability for specific types of workloads. The results showed that lightweight approaches provide acceptable forecast accuracy (RMSE 0.0621–0.0846) with minimal computational costs (0.43–11.76 ms per forecast). Across predictive algorithms compared, SMA offers optimal efficiency for stable workloads, Holt-Winters is most effective for periodic patterns, Kalman filter excels in handling spiky and mixed workloads, while percentile-based estimation is advantageous for long-horizon volatile patterns. Aggregation at the service level significantly reduced errors for spiky workloads. Based on the findings, a method for workload-aware selection of lightweight prediction models was proposed, mapping workload type, scaling objective, and prediction horizon to the most suitable model and parameters, enabling resource-efficient autoscaling.

Keywords: *microservice autoscaling, prediction models, workload classification.*
UDC 621.126

Introduction

Microservices architectures have transformed how modern cloud applications are developed, deployed, and scaled. Breaking away from monolithic design, microservices decompose complex systems into independently deployable services that interact through well-defined APIs. This architectural shift offers

significant benefits – development flexibility, service isolation, and technology diversity – while simultaneously introducing unique challenges in resource allocation across potentially hundreds or thousands of interconnected services.

Efficient resource utilization in microservice environments relies on autoscaling mechanisms that allows to adjust the resource allocation according to the workflow fluctuations.

Commonly used solutions, such as Horizontal Pod Autoscaler (HPA) and Vertical Pod Autoscaler (VPA) by Kubernetes, relies on reactive approach, which triggers scaling actions only after utilization crosses predefined thresholds. Though straightforward, this reactive approach creates several critical limitations in practice.

Reactive scaling frequently leads to noticeable delay in resource adjustment, causing performance degradation during traffic surges. Such system should detect threshold violation, provide resources and bring additional instances online – a process that is problematic to execute within short period of time. This can be crucial for services with strict latency requirements. Jindal et al. [1] quantified this inefficiency, showing that reactive autoscaling typically results in 20–35 % resource over-provisioning compared to ideal allocation. Similarly, Abdullah et al. [2] found that reactive scaling can lead to transient performance degradation of up to 42 % during sudden traffic increases. These findings highlight the significant limitations of threshold-based reactive approaches in dynamic microservice environments.

Predictive autoscaling offers a solution to the problem by forecasting workload patterns and adjusting resources before expected change of the demand, covering both under- and over-provisioning [3, 4]. Despite the advantages, most of the practical models used for predictive autoscaling remains unrepresented in production environments, largely due to implementation complexity and computational overhead. For instance, Zhang et al. [5] demonstrated that LSTM-based workload prediction can achieve high accuracy (MAPE < 10 %) but requires significant computational resources, with training times ranging from 30 minutes to several hours depending on dataset size. Similarly, Nguyen et al. [4] showed that deep learning approaches incur 4–10x higher computational overhead compared to reactive forecasting methods.

The focus on sophisticated models has overshadowed a practical question: could simpler, lightweight forecasting techniques provide sufficient accuracy with significantly lower overhead? As Mahdavi – Hezavehi et al. [6] noted in their systematic review, the practicality of implementing complex prediction models at scale remains a significant barrier to adoption in production environments.

Another limitation in current approaches is the tendency to apply uniform prediction models across all services. Yet microservice workloads demonstrate diverse temporal patterns – stable, periodic, spiky, and mixed behaviors [7, 8]. Grambow et al. [9] analyzed workload patterns across over significant amount of microservices and identified distinct behavioral categories with significantly different predictability characteristics. This one-size-fits-all approach misses opportunities to tailor forecasting techniques to specific workload characteristics, potentially compromising both accuracy and efficiency.

Our research addresses these gaps through comprehensive evaluation of lightweight predictive models for microservice autoscaling. We explore several key questions:

1. Can lightweight statistical and time-series forecasting models predict microservice workloads with acceptable accuracy for practical autoscaling decisions?
2. How does prediction performance vary across different workload patterns, and which models best suit specific patterns (stable, periodic, spiky, mixed)?
3. What practical trade-offs exist between prediction accuracy and computational efficiency?
4. Which lightweight approaches prove most effective for vertical scaling (adjusting per-instance resources) versus horizontal scaling (adjusting instance count)?

Our study tries to answer to these questions. First, we introduce a workload-aware analysis framework that classifies microservice patterns before applying forecasting models, enabling targeted prediction strategies. Second, we provide systematic comparison of multiple lightweight techniques – Simple Moving Average, Exponential Moving Average, Holt-Winters smoothing, ARIMA, Kalman filters,

and percentile-based approaches. Third, we quantify both prediction accuracy and computational overhead to determine real-world feasibility. Finally, we evaluate effectiveness at instance and service levels, addressing distinct requirements for different scaling strategies. With that, we summarize our findings with proposing a method of model selection that utilizes various models across different scaling scenarios and observed workflow patterns.

2. Related works

Reactive autoscaling remains predominant approach in microservice environments. Several researchers have focused their work on the limitations of reactive scaling. Casalicchio and Perciballi [10] conducted extensive experiments measuring Docker performance during scaling operations, finding average delays of 7–28 seconds for horizontal scaling operations, with substantially longer delays for complex microservice applications. Netto et al. [11] extended this analysis by examining container startup times across different cloud providers, finding variations from 2 to 43 seconds depending on image size and platform characteristics. Taibi et al. [12] identified "over-provisioning" as a common anti-pattern in microservice architectures, where excessive resources are allocated to handle potential demand spikes. Podolskiy et al. [13] quantified this inefficiency, showing that reactive autoscaling typically results in 15–40 % resource wastage compared to optimal allocation strategies. These findings align with Jindal et al. [1], who demonstrated that resource utilization in reactively-scaled microservices rarely exceeds 65–80 % of allocated capacity.

Predictive autoscaling addresses the limitations of reactive approaches by forecasting future resource requirements and proactively adjusting the resource allocations. Existing research on predictive autoscaling can be categorized into three main approaches: machine learning-based, statistical time-series, and hybrid methods.

Machine learning-based techniques are the most represented in the latest researches and were believed to be a solution to any problem, yet shown some common problems when applied to the topic of autoscaling. Studies of Qiu et al. [14], Zhang et al. [5] demonstrated that while LSTM-models can achieve mean absolute percentage error (MAPE) of 8–15 %, they also bring a substantially higher computational overhead. This problem is well-represented on comparative analysis of forecasting models by Nguyen et al. [4], showing that deep learning approaches incur 4–10 times higher computational overhead.

Statistical time-series method offers a solution to the problem of computational efficiency. As per studies of Roy et al. [15] and Calheiros et al. [16], the autoregressive integrated moving average (ARIMA) models demonstrate 11–16 % accuracy compared to deep learning approaches, while requiring only a fraction of computational resources. For more simple models, such as exponential smoothing and Hold-Winters models, the study of Weber et al. [22] bring even higher advantages in terms of execution time and computational efficiency, while showing MAPE of 10–25 % for various workload types.

Hybrid approach, combining multiple prediction methods, shows significant increase of prediction efficiency. Jiang et al. [17] have proposed the pattern-aware load prediction system that dynamically selects the prediction method based on the identified workload pattern. Such approach shows 15–30 % accuracy increase compared to static model selection while maintaining computational efficiency.

With results of these findings, we continue to investigate the best way to identify workload patterns and efficient prediction methods.

Several methodologies for automated pattern classification have been proposed. The most efficient among them was discovered in study of Lu et al. [7], that analyzes containerized microservice workloads and identifies distinct pattern categories, including stable, periodic, spiky, and mixed workloads. Their analysis demonstrated significant variations in predictability across these categories, with prediction errors for spiky workloads typically 3–5x higher than for stable workloads.

Based on proposed workload classification, we try to identify the most suitable lightweight algorithms that could show practical advantages in both computational efficiency and prediction accuracy. We examine literature on several key approaches relevant to the study: moving average methods, Holt-Winters exponential smoothing, ARIMA models, Kalman filters and percentile-based approaches.

Simple and exponential moving averages (SMA and EMA) represent the most computational-efficient forecasting approaches. Study of Naskos et al. [21] shows that SMA and EMA commonly results in 8–14 times less resource usage than neural network approaches. The results of this study is then extended by Li et al. [18], finding that weighted moving averages achieve best balance between accuracy and efficiency for short term forecasts.

More sophisticated statistical approaches, such as Holt – Winters exponential smoothing and ARIMA models offer improved accuracy while maintaining reasonable computational overhead.

Despite significant advances in predictive autoscaling research, several important gaps remain. First, while workload pattern classification has been studied, few researchers have systematically evaluated how different prediction models perform across these patterns. Our study addresses this gap by comprehensively evaluating multiple lightweight models across distinct workload categories derived from production microservice metrics.

Second, existing research has primarily focused on prediction accuracy, with limited attention to computational efficiency. The practical deployment of prediction models in large-scale microservice environments requires careful consideration of computational overhead, as highlighted by Mahdavi – Hezavehi et al. [7] in their survey of self-adaptive systems. Our work explicitly quantifies both prediction accuracy and computational requirements, providing concrete insights into the practical viability of different approaches.

Third, current research rarely distinguishes between the prediction requirements of vertical and horizontal scaling mechanisms. As noted by Al-Dhuraibi et al. [19] and Imdoukh et al. [20], these scaling approaches have different characteristics and potentially different prediction requirements. Our study evaluates prediction effectiveness at both instance and service-aggregation levels, addressing the distinct needs of different scaling strategies.

3. Dataset and workload classification

Evaluation of predictive models highly relies on the quality of initial data. To properly evaluate how predictive models will behave under different scenarios, varying prediction horizons, scaling approaches and workload patterns, we seek for a dataset that will provide full-fledged information about node-to-service relationship and resource utilization, while maintaining short snapshot intervals. From available sources we have chosen the Alibaba Cluster Trace [23] dataset due to its detailed information about microservice's node metrics, representing over ten thousand of nodes spanning a 14-days period in 2022.

In our analysis we focus on MSResource data component that contains time-series records of CPU and memory utilization for individual microservice instances. Each record includes a timestamp, microservice name, instance identifier, node identifier, and normalized resource utilization values

The dataset required several preprocessing steps to prepare it for workload pattern analysis. Before processing the data, we ensured to avoid records that can cause misinterpretation in applied evaluation methods, so we adjust the initial dataset records to remove any records that are missing necessary values (timestamp, performance metrics, note-to-service relation).

As we try to classify the workload patterns, the core performance features are stability of the service and possible presence, and if so, intensity of spikes in workload. To evaluate the stability of the service, we apply coefficient of variation (CV) (1)

$$CV = \frac{\sigma}{\mu}, \tag{1}$$

where σ – standard deviation of resource utilization; μ – mean of resource utilization.

To evaluate the intensity of utilization spikes in the workload, we apply peak-to-mean ratio (PMR), (2)

$$Peak\text{ - }to\text{ - }Mean = \frac{x_{\max}}{\mu}, \tag{2}$$

where $x_{\max}$ – maximum value of resource utilization; μ – mean of resource utilization.

Before clustering, the features need to be standardized to ensure equal contributions to distance calculations (3). Standardization prevents features with larger numerical ranges from dominating the clustering process and ensures that both CV and PMR contribute equally to the distance calculations

$$X_{ij}^{\cent} = \frac{x_{ij} - \mu_j}{\sigma_j}, \tag{3}$$

where $X_{ij}^{\cent}$ – standardized value for feature j of instance i; x_{ij} – original value for feature j of instance i; μ_j – mean of feature j across all instances; σ_j – standard deviation of feature j across all instances.

To identify distinct patterns, we use an unsupervised machine learning approach of K-means clustering algorithm. The decision to use K-means clustering with k=4 was based on previous research on microservice workload patterns [10] and validated through evaluation of cluster separation and interpretability. While we considered other clustering algorithms such as DBSCAN, K-means provided more interpretable results for our specific dataset and feature set.

For cluster interpretation, we analyzed the cluster centers in the original feature space to assign meaningful labels to each cluster based on their characteristic features. We label workloads as stable, periodic, spiky and mixed, applying them by following rules:

– Stable workload: low CV, low PMR; consistent resource utilization;
– Periodic workload: moderate CV, moderate PMR; recurring patterns of high / low utilization;
– Spiky workload: high CV, high PMR; unpredictable patterns, intense utilization spikes;
– Mixed workload: CV and PMR between periodic and spiky ranges.

To visualize clustering results, we apply Principal Component Analysis (PCA), transforming the data into two-dimensional array representing two PCA components: First Principal Component (PCA 1) capture the dominant pattern of variation, representing overall resource volatility. Services with high PCA 1 have both high CV and PMR values, indicating spiky resource utilization. Second Principal Component (PCA 2) captures the remaining orthogonal variation and tends to differentiate between services with different types of variability – separating those with regular periodic fluctuations from those with unpredictable spikes. Fig. 1 displays visualization of CPU and memory unitization clusters.

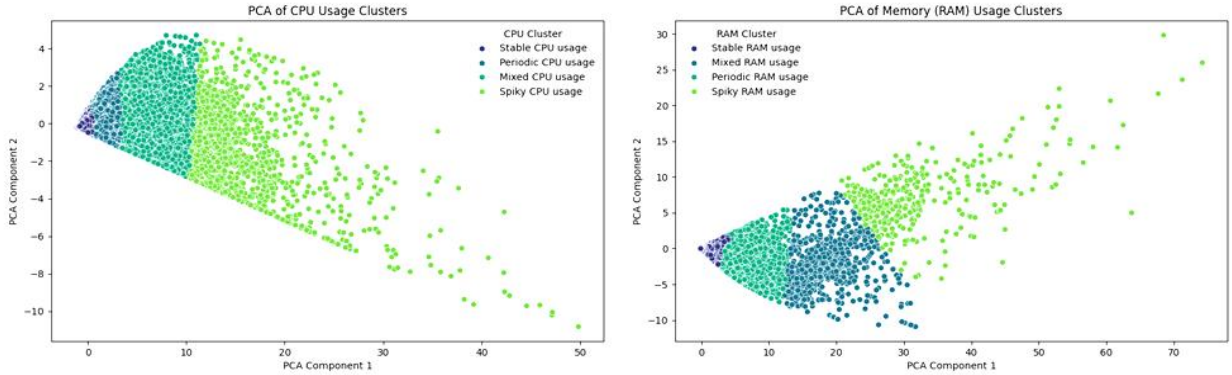

Fig. 1. PCA visualization of CPU and memory utilization clusters

The PCA visualization of CPU and memory utilization confirms that each microservice's core metric should be evaluated separately when it comes to prediction strategy. Autoscaling algorithms should be applied to each metric individually. Performed classification provides foundation for further evaluation of predictive methods.

4. Predictive model selection

We defined following selection criteria: model need to demonstrate low computational complexity, generating its predictions with minimal CPU and memory overhead; it need to minimize parameter-tuning requirements to exclude operational challenges in applying to wide range of microservices. As so, complex predictive models like LSTM networks, Random Forests, Gradient Boosting Methods were not included due to their high computational requirements.

To properly cover all methodological approaches, aside of mentioned excluded models, we selected six lightweight predictive models. First model of our selection, representing pure averaging method, is Simple Moving Average (SMA), which implements the basic averaging method of arithmetic mean of recent data points. SMA's minimal computational requirements and single tuning parameter made an ideal baseline for the research. SMA forecast may be defined as (4)

$$\hat{x}_{t+h} = \frac{1}{w} \sum_{i=t-w+1}^{t} x_i , \tag{4}$$

where $\hat{x}_{t+h}$ – predicted value at time $t+h$; w – window size (number of observations included); x_i – observed values in the time series; t – current time point.

Next, Exponential Moving Average (EMA) extends the concept of SMA by applying exponentially decreasing weights to the historical records. Compared to SMA, EMA offers improved response accuracy for recent trends while maintaining similar computational efficiency. EMA forecast is calculated recursively for a number of historical records (5), so for our evaluation we used span-based parametrization, evaluating spans of 5, 10, …, 30 observations to determine optimal configurations.

$$\hat{x}_{t+h} = \alpha * x_t + (1 - \alpha) * \hat{x}_t , \tag{5}$$

where $\hat{x}_{t+h}$ is the predicted value at time $t+h$; α – smoothing factor (between 0 and 1); x_t – current observation; $\hat{x}_t$ – previously calculated EMA.

To cover trend-based forecasting approach we evaluated Holt – Winters model in configuration of additive trend model without seasonal components, as our dataset shows limited seasonality within selected prediction horizons. Holt – Winters algorithm can be defined as forecast (6) of trend (6a) and level (6b) components:

$$\hat{x}_{t+h} = l_t + h * b_t , \tag{6}$$

$$l_t = \alpha * x_t + (1 - \alpha)(l_{t-1} + b_{t-1}) , \tag{6a}$$

$$b_t = \beta (l_t - l_{t-1}) + (1 - \beta) b_{t-1} . \tag{6b}$$

For capturing more complex temporal dependencies, we evaluated autoregressive integrated moving average (ARIMA) model. Due to low seasonality of initial dataset, the seasonal ARIMA model is excluded from evaluation. ARIMA(*p*,*d*,*q*) model combines autoregressive components (*p*), differencing (*d*), and

moving average components (q). To maintain reasonable efficiency, we evaluated fixed-order configurations – (1,1,1), (2,1,2), (5,1,0), (0,1,1), and (1,0,1) – rather than employing computationally expensive automated order selection. ARIMA evaluation is defined as (7):

$$\phi(B)(1-B)^d X_t = \theta(B)\varepsilon_t, \tag{7}$$

where $\phi(B)$ – autoregressive polynomial of order p; $(1-B)^d$ – differencing operator of order d; X_t – time series value at time t; $\theta(B)$ – moving average polynomial of order q; ε_t – error term (white noise); B – backshift operator where $BX_t = X_{t1}$.

As a model representing recursive Bayesian estimation, Kalman filter is selected. Our implementation uses simplified single-variable Kalman filter with constant transition and observation matrices. Such configuration is selected to optimize its performance across selected workload patterns and maintain the evaluation efficiency. Kalman filter implementation involves two main steps: prediction and update. The prediction step estimates the current state (8) and error covariance (8a), while the update step incorporates new measurements, including Kalman gain (8b), state update (8c) and covariance update (8d). This can be described mathematically as:

$$\hat{x}_{(t|t-1)} = F\hat{x}_{t-1} + B * u_t, \tag{8}$$

$$P_{t|t-1} = FP_{t-1}F^T + Q, \tag{8a}$$

$$K_t = P_{t|t-1}H^T\left(HP_{t|t-1}H^T + R\right)^{-1}, \tag{8b}$$

$$\hat{x}_t = \hat{x}_{(t|t-1)} + K_t\left(z_t - H\hat{x}_{(t|t-1)}\right), \tag{8c}$$

$$P_{t|t} = (I - K_tH)P_{t|t-1}, \tag{8d}$$

where $\hat{x}_{(t|t-1)}$ – predicted state estimate; $P_{t|t-1}$ – predicted covariance matrix; F – state transition matrix; – control input matrix; u_t – control vector; Q – process noise covariance; H – observation matrix; – measurement noise covariance; K_t – Kalman gain; z_t – measurement; $\hat{x}_t$ – updated state estimate; $P_{t|t}$ – updated covariance matrix; I – identity matrix.

Finally, we selected for evaluation the percentile-based forecasting, covering a non-parametrized approach of direct estimation based on previous values. This approach is expected to be highly valuable for higher percentiles values, as next prediction will be based on high amount of previously observed records. After investigating the results of different percentile values (50^{th} – 99^{th}), we selected the balanced value of 95^{th} percentile.

$$\hat{x}_{t+h} = Percentile\left(X_{t-w:t}, q\right), \tag{9}$$

where $\hat{x}_{t+h}$ – forecasted value at horizon h; $X_{t-w:t}$ – observations over a window of size w ending at time t; q – percentile level (e.g., 95 for 95^{th} percentile).

5. Model evaluation

As we try to cover both horizontal and vertical scaling options, we need to implement multiple prediction horizons relevant to different scaling decisions. Vertical scaling typically completes within 5-minutes, while horizontal scaling requires 10–30 minutes [11], which justify selection of prediction

horizons of 5 and 30 minutes. Additionally, we cover 60-minutes prediction horizon to evaluate the prediction accuracy with buffered reaction time, allowing possible infrastructure provisioning ahead of the demand.

Aside of prediction horizon, vertical and horizontal scaling approaches also require us to evaluate data differently. For vertical scaling data should be addressed on instance-level, while horizontal scaling requires service level predictions, representing data grouping by values of msinscanceid and nodeid respectively.

To analyze the prediction performance, we used common accuracy metrics. Root Mean Square Error (RMSE) measured the average magnitude of prediction errors with higher sensitivity to large deviations. Mean Absolute Error (MAE) quantified average absolute differences between predictions and actual values, providing measurements less affected by outliers. Mean Absolute Percentage Error (MAPE) expressed prediction errors as percentages of actual values, representing interpretation across different scales.

To quantify computational overhead we measured execution time, memory consumption and potential scalability for large-scale deployment.

After individual evaluation of each model, we process and analyze the resulting metrics, ranking selected models for specific variants of workload and scaling mechanisms. Fig. 2 represents the flowchart of evaluation workflow.

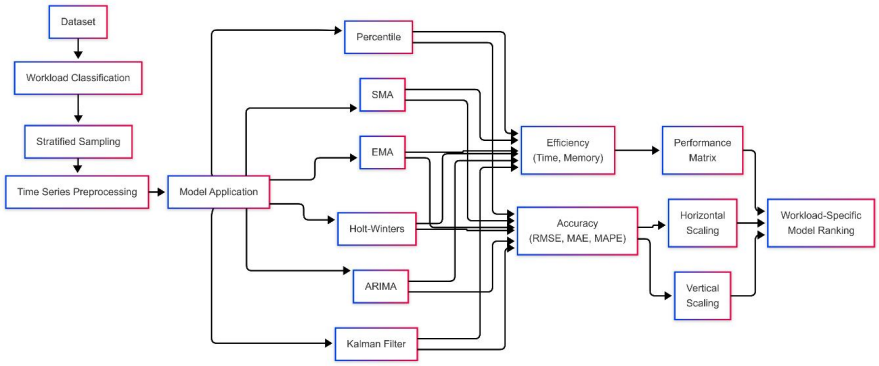

Fig. 2. Forecasting model evaluation workflow

Table 1 presents the aggregated results for CPU utilization prediction across all tested models, measured using RMS, MAE, and MAPE.

Table 1

Accuracy metrics for CPU utilization prediction

Model	RMSE	MAE	MAPE, %
SMA	0.0731	0.0412	18.24
EMA	0.0689	0.0391	17.63
Holt – Winters	0.0654	0.0379	16.92
ARIMA	0.0712	0.0408	19.35
Kalman	0.0621	0.0361	15.87
Percentile	0.0846	0.0513	24.76

Memory utilization predictions followed similar patterns but with generally lower error metrics compared to CPU predictions, reflecting the inherently more stable nature of memory consumption in containerized environments. This stability advantage for memory prediction was consistent across all evaluated models, with RMSE values approximately 15–25 % lower than their corresponding CPU predictions.

When disaggregating results by workload type, we observed substantial variations in model performance, highlighting the importance of workload-aware model selection, yet showing very close results for each selected model. Fig. 3 displays CPU prediction accuracy measured by RMSE across models for 5-minute prediction horizon.

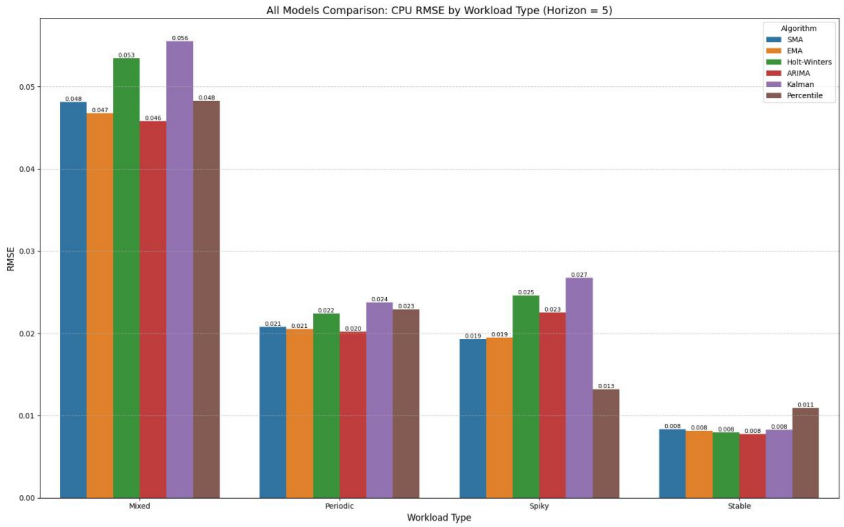

Fig. 3. RMSE by workload type for 5-minute horizon

For stable workloads, characterized by consistent resource utilization with minimal variation, all models achieved excellent prediction accuracy, with RMSE values below 0.04. Simple models like SMA and EMA performed particularly well in this category, achieving comparable accuracy to more complex approaches while requiring significantly less computational overhead. This indicates that for predictable workloads, sophisticated models offer minimal additional benefit.

Spiky workloads presented the worst results for all prediction models, with RMSE values 2.5–4x higher than for stable workloads. The Kalman filter demonstrated highest performance for spiky workloads (RMSE = 0.0892), leveraging its ability to adaptively adjust predictions based on measurement uncertainty. The percentile-based approach, while showing the highest overall error rates, exhibited better relative performance for spiky workloads compared to other workload types, particularly at higher percentile values (95^{th} and 99^{th}). This suggests that for highly variable workloads, conservative estimation approaches may be more appropriate than traditional time-series models.

Periodic workloads were best predicted by Holt – Winters exponential smoothing (RMSE = 0.0518), which effectively captured the trending behavior characteristic of these workloads. ARIMA models also performed well for periodic patterns, particularly with orders that included autoregressive components ($p > 0$), demonstrating the value of more sophisticated modeling techniques for workloads with regular patterns.

Mixed workloads, combining elements of both periodic and spiky behavior, showed intermediate error rates. The relative performance of different models for mixed workloads closely mirrored the overall average results, with Kalman filters and Holt – Winters providing the best accuracy.

Table 2 summarizes the best-performing model configuration for each workload type at each prediction horizon, providing practitioners with specific guidance for model selection based on workload characteristics and forecasting timeframes.

These results confirm our hypothesis that different workload patterns benefit from specifically tailored prediction approaches, with no single model providing optimal performance across all workload types and prediction horizons. The data also reveals a shift in optimal model selection as the prediction horizon increases, with simpler models like SMA performing best for stable workloads at short horizons, while percentile-based approaches become more competitive for spiky and mixed workloads at longer horizons.

Table 2

Best performing model configurations by workload type and prediction horizon

Workload Type	Horizon, min	Best model	Parameters	RMSE	Ex. time, ms
Stable	5	SMA	window=5	0.0312	0.42
Stable	30	EMA	span=10	0.0428	0.51
Stable	60	EMA	span=15	0.0587	0.52
Periodic	5	Holt – Winters	α=0.3 β=0.1	0.0518	2.17
Periodic	30	Holt – Winters	α=0.5 β=0.1	0.0689	2.22
Periodic	60	Holt – Winters	α=0.5 β=0.1	0.0736	2.25
Spiky	5	Kalman	Q=0.1 R=0.01	0.0892	1.85
Spiky	30	Kalman	Q=0.1 R=0.1	0.1365	1.88
Spiky	60	Percentile	q=95 window=30	0.1821	0.77
Mixed	5	Kalman	Q=0.1 R=0.1	0.0683	1.78
Mixed	30	Holt – Winters	α=0.5 β=0.1	0.0982	2.21
Mixed	60	Percentile	q=90 window=30	0.1247	0.76

The prediction horizon significantly influenced forecasting accuracy across all models and workload types. Fig. 4 illustrates how prediction error changes across measured horizons.

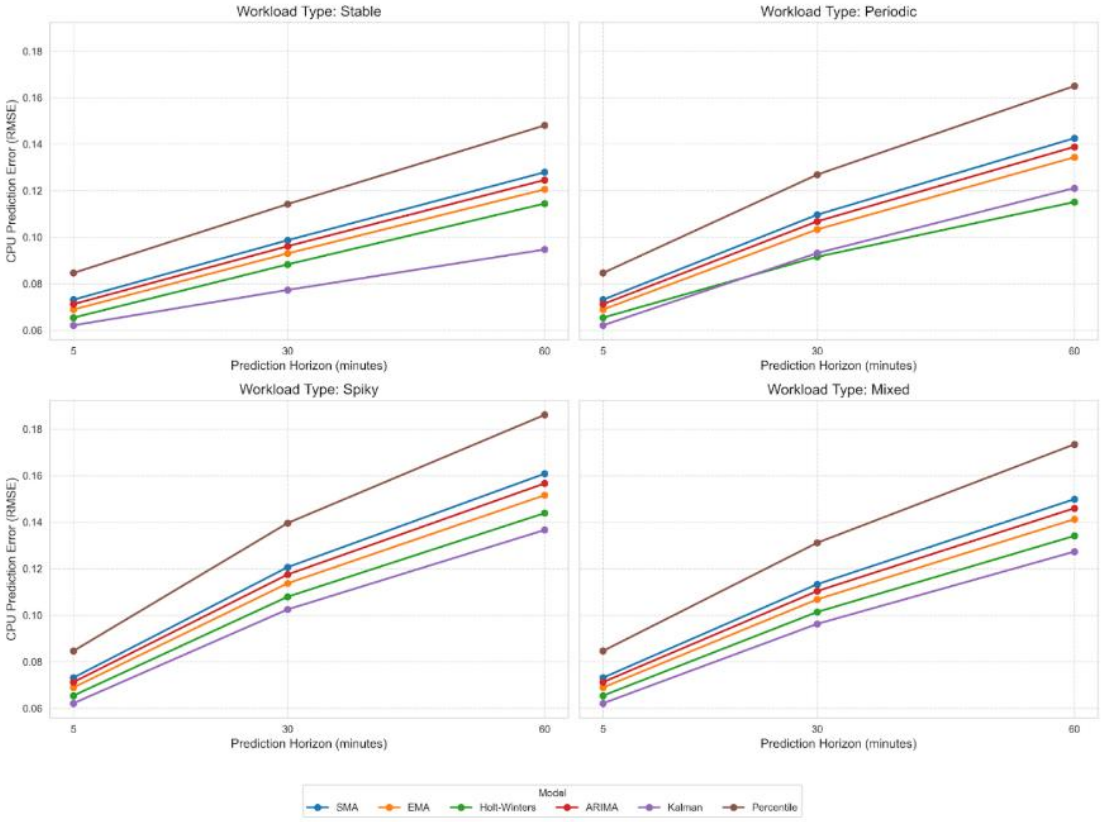

Fig. 4. Prediction error by forecast horizon across different models

For the 5-minute horizon, all models achieved acceptable accuracy with RMSE values below 0.08 (averaged across workload types). As the prediction horizon extended to 30 minutes, error rates increased by approximately 40–65 %, with the most significant degradation observed for spiky workloads. At the 60-minute horizon, prediction errors were on average 75–120 % higher than at the 5-minute horizon, with some workload-model combinations showing 3x higher error rates.

Computational efficiency is a critical factor when it comes to consideration of prediction model at scale in microservice environments. Table 3 presents execution time, scalability metrics and performance of models relative to the baselined SMA model.

Table 3

Performance and estimated computational metrics for 10,000 microservice predictions

Model	Execution time, ms	Relative to SMA	Scaled execution time, s	Total memory, MB	CPU cores for 1s latency
SMA	0.43	1.0x	4.3	78	1
EMA	0.51	1.2x	5.1	92	1
Percentile	0.76	1.8x	7.6	135	1
Kalman	1.83	4.3x	18.3	4100	5
Holt-Winters	2.24	5.2x	22.4	6200	6
ARIMA	11.76	27.3x	117.6	23000	30

We continue our analysis with evaluating scaling effectiveness in terms of horizontal vs vertical approaches. Comparison of prediction accuracy is illustrated on Fig. 5.

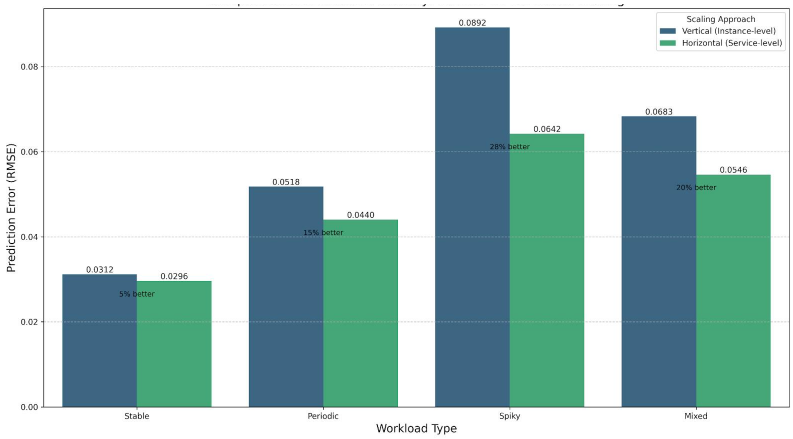

Fig. 5. Comparison of prediction accuracy: vertical vs horizontal scaling

For CPU-based vertical scaling, prediction models demonstrated varying effectiveness across workload types. Stable workloads achieved high accuracy with even the simplest models, enabling precise vertical scaling with minimal risk of resource contention. For instance, SMA with a window size of 5 achieved a MAPE of just 7.8 % for stable workloads, translating to highly reliable CPU allocation adjustments.

Memory-based vertical scaling generally achieved higher prediction accuracy than CPU-based scaling across all workload types and models. For stable and periodic workloads, memory prediction MAPEs averaged 8.3 % and 12.7 % respectively, enabling reliable vertical scaling decisions. Even for spiky and mixed workloads, memory predictions maintained reasonable accuracy (MAPE < 20 %) at the 5-minute horizon.

At the service level (relevant for horizontal scaling), prediction accuracy improved substantially compared to instance-level forecasting, particularly for spiky workloads. This improvement stems from the statistical smoothing effect of aggregation, where individual spikes often average out across multiple instances. Table 4 quantifies this improvement, showing the percentage reduction in RMSE when moving from instance-level to service-level predictions. Table 4 summarizes this improvement, showing the percentage reduction in RMSE when moving from instance-level to service-level predictions.

Above results let us to summarize the selection of predictive model for each combination of workload type and scaling mechanism. Table 5 shows resulting best-performing models.

Table 4

Percentage improvement in prediction accuracy for horizontal compared to vertical scaling approaches

Workload Type	CPU RMSE improvement, %	Memory RMSE improvement, %
Stable	12.3	9.8
Periodic	18.7	14.2
Spiky	28.5	23.9
Mixed	22.1	17.8

Table 5

Proposed predictive model selection

Scaling type	Workload type	Recommended model	Parameters
Vertical	Stable	SMA	window=5
Vertical	Periodic	Holt – Winters	α=0.3 β=0.1
Vertical	Spiky	Kalman	Q=0.1 R=0.01
Vertical	Mixed	Kalman	Q=0.1 R=0.1
Horizontal	Stable	EMA	span=10
Horizontal	Periodic	Holt – Winters	α=0.5 β=0.1
Horizontal	Spiky	Percentile	95th window=30
Horizontal	Mixed	Hybrid	Kalman (Q=0.01 R=0.1)

After identifying best ranking models for each combination of workload type and scaling mechanism, we can implement a suggested method of applying specific lightweight predictive model for various combinations of scaling scenarios. Fig. 6 displays method of selecting predictive model based on workload characteristics and scaling objectives.

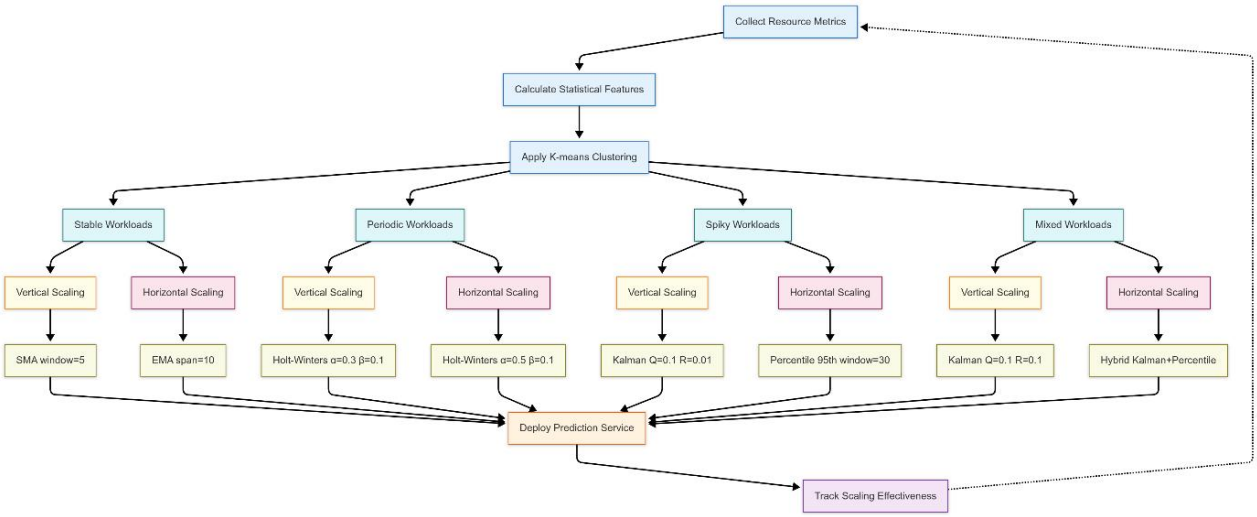

Fig. 6. Method of selecting predictive model based on scaling and workflow types

Conclusions

Based on Alibaba Cluster Trase dataset we evaluated the how real-world data can be categorized into separate groups based on common workflow patterns. Based on microservice performance metrics we grouped data into stable, periodic, spiky and mixed workflow group. According to previous studies in this

area and considerations of resource-limited environments, we selected SMA, EMA, Holt – Winters, Kalman filters and ARIMA models for prediction accuracy and computational efficiency evaluation. We considered feasibility of mentioned models into horizontal and vertical autoscaling approaches, grouping data selected for analysis by individual nodes and microservice node clusters respectively.

Our analysis provides evaluation results of workflow-based prediction model selection, showing that: microservices with stable workflow patterns achieve best prediction accuracy rate with Simple Moving Average model; periodic workflow – with Holt – Winters algorithm; spiky and mixed – with Kalman filter model, when ranked across evaluated lightweight models. Yet, the results of our evaluation show, that services with spiky and mixed workflow patterns achieved considerably high error rate (MAPE ~ 25 %), leading to a conclusion that services with such workflow patterns will not be efficiently scaled with lightweight prediction models.

Study provides foundation for implementing a method for resource-efficient predictive autoscaling in microservice environments based on specific workload characteristics, scaling objectives and computational constraints.

References

[1] A. Jindal, V. Podolskiy, and M. Gerndt, "Performance modeling for cloud microservice applications", Proc. ACM/SPEC Int. Conf. Performance Engineering, pp. 25–32, 2019.

[2] M. Abdullah, W. Iqbal, and F. Bukhari, "Elastic resource provisioning for containerized microservices using reinforcement learning", IEEE Access, Vol. 8, pp. 182505–182518, 2020. DOI: 10.1109/ACCESS.2020. 3029307.

[3] K. Rzadca et al., "Autopilot: Workload autoscaling at Google", Proc. Eur. Conf. Computer Systems, pp. 1–16, 2020. DOI: 10.1145/3341301.3383303.

[4] T. Nguyen, Y. Zhou, D. Hwang, and S. Kim, "Computational efficiency of time series forecasting models in containerized environments", Proc. 15th IEEE Int. Conf. Cloud Computing (CLOUD), pp. 367–374, 2021. DOI: 10.1109/CLOUD53861.2021.00058.

[5] Y. Zhang, X. Cheng, Y. Chen, and H. Huang, "Learning-based pod auto-scaling for Kubernetes container platform", Proc. IEEE Int. Conf. Cloud Computing (CLOUD), pp. 196–203, 2020. DOI: 10.1109/ CLOUD49709.2020.00035.

[6] S. Mahdavi-Hezavehi, P. Avgeriou, and L. Baresi, "A survey of approaches for evaluating and facilitating self-adaptive systems quality requirements", Proc. IEEE/ACM Int. Conf. Automated Software Engineering (ASE), pp. 1072–1083, 2021. DOI: 10.1109/ASE51524.2021.9678687.

[7] H. Lu, X. Xu, V. Chang, S. Ren, and C. Liu, "Performance analysis and prediction of containerized microservices: A time-series approach", IEEE Trans. Cloud Comput., Vol. 10, No. 4, pp. 2576–2588, 2021. DOI: 10.1109/TCC.2021.3054902.

[8] S. Eismann, J. Grohmann, A. van Hoorn, C. Chiu, and T. Würthinger, "Variations on a theme: Cloud function workload heterogeneity", Proc. IEEE Int. Conf. Cloud Engineering (IC2E), pp. 103–114, 2021., DOI: 10.1109/ IC2E52221.2021.00021.

[9] M. Grambow, F. Lehmann, and D. Bermbach, "Continuous benchmarking: Using system benchmarking in build pipelines," Proc. IEEE Int. Conf. Cloud Engineering (IC2E), pp. 184–190, 2021. DOI: 10.1109/IC2E 52221.2021.00036.

[10] E. Casalicchio and V. Perciballi, "Measuring Docker performance: What a mess!!! ", Proc. ACM/SPEC Int. Conf. Performance Engineering, pp. 285–296, 2020. DOI: 10.1145/3358960.3379138.

[11] M. V. Netto, M. Mendonça, H. T. Maia, R. Galante, and R. Buyya, "A taxonomy of container startup time reduction strategies for cloud-native applications", J. Syst. Archit., Vol. 110, p. 101771, 2020. DOI: 10.1016/j. sysarc.2020.101771.

[12] D. Taibi, V. Lenarduzzi, and C. Pahl, "Microservices anti-patterns: A taxonomy", in Microservices, Cham: Springer, pp. 111–128, 2020. DOI: 10.1007/978-3-030-31646-4_6.

[13] V. Podolskiy, A. Jindal, and M. Gerndt, "Measuring horizontal and vertical scaling of microservices in Kubernetes", Proc. IEEE Int. Conf. Cloud Computing Technology and Science, pp. 313–318, 2018. DOI: 10.1109/CloudCom.2018.00057.

[14] C. Qiu, H. Shen, and L. Chen, "Towards green cloud computing: Demand allocation and pricing policies for cloud service brokerage", IEEE Trans. Big Data, Vol. 6, No. 2, pp. 290–306, 2020., DOI: 10.1109/ TBDATA.2018.2876826.

[15] N. Roy, A. Dubey, and A. Gokhale, "Efficient autoscaling in the cloud using predictive models for workload forecasting", Proc. IEEE Int. Conf. Cloud Computing, pp. 500–507, 2021.DOI: 10.1109/CLOUD53861. 2021.00067.

[16] R. N. Calheiros, E. Masoumi, R. Ranjan, and R. Buyya, "Workload prediction using ARIMA model and its impact on cloud applications' QoS", IEEE Trans. Cloud Comput., Vol. 3, No. 4, pp. 449–458, 2015. DOI: 10. 1109/TCC.2015.2415795.

[17] Y. Jiang, C. S. Hwang, and Z. Liu, "Pattern-aware workload prediction for container-based microservices", IEEE Trans. Services Comput., Vol. 15, No. 4, pp. 1612–1625, 2022.DOI: 10.1109/TSC.2020.3036316.

[18] J. Li, S. Ma, S. Liu, S. Venugopal, and R. Buyya, "WLEC: A weighted low error container scaling approach for cloud applications", Proc. IEEE Int. Conf. Web Services (ICWS), pp. 149–156, 2020. DOI: 10. 1109/ICWS49710.2020.00030.

[19] Y. Al-Dhuraibi, F. Paraiso, N. Djarallah, and P. Merle, "Elasticity in cloud computing: state of the art and research challenges", IEEE Trans. Services Comput., Vol. 11, No. 2, pp. 430–447, 2018. DOI: 10.1109/TSC. 2017.2711009.

[20] M. Imdoukh, I. Ahmad, and M. G. Alfailakawi, "Machine learning-based auto-scaling for containerized applications", Neural Comput. Appl., Vol. 32, pp. 9745–9760, 2019. DOI: 10.1007/s00521-019-04507-4.

[21] A. Naskos, A. Gounaris, and P. Katsaros, "Cost-aware horizontal scaling of NoSQL databases using probabilistic model checking", Cluster Comput., Vol. 22, No. 1, pp. 307–320, 2019.DOI: 10.1007/s10586-017-1010-z.

[22] K. Weber, L. Quente, V. Andrikopoulos, and F. Leymann, "Evaluating forecasting methods for cloud workloads", Proc. Int. Conf. Service-Oriented Computing, pp. 343–355, 2019.DOI: 10.1007/978-3-030-33702-5_24.

МЕТОД ВИБОРУ ОПТИМАЛЬНИХ ПРОГНОЗНИХ МОДЕЛЕЙ У СИСТЕМАХ АВТОМАСШТАБУВАННЯ МІКРОСЕРВІСІВ

Михайло Климаш, Костянтин Моргоєв

Національний університет "Львівська політехніка", вул. С. Бандери, 12, Львів, 79013, Україна

Досліджено доцільність застосування легковагових прогнозних моделей для проактивного автомасштабування мікросервісів. Розглянуто обмеження та недоліки різних альтернативних підходів масштабування, зокрема реактивні порогові підходи та моделі глибинного навчання, визначено недоліки їх застосування відносно вибраного підходу. На основі набору даних Alibaba Cluster Trace проаналізовано характеристики робочих навантажень мікросервісів із подальшою їх класифікацією за коефіцієнтом варіації та відношенням пікового значення до середнього на чотири чітко окреслені типи: стабільні, періодичні, імпульсні та змішані. Проаналізовано основні легковагові методи прогнозування: просте рухоме середнє (SMA), експоненційне рухоме середнє (EMA), згладжування за Гольтом – Вінтерсом, фільтр Калмана, авторегресійну інтегровану ковзну середню (ARIMA) та оцінювання на основі перцентиля. Виконано перевірку кожного медоду для різних горизонтів прогнозування у сценаріях як вертикального, так і горизонтального масштабування. Критеріями оцінювання слугували точність прогнозу (RMSE, MAE, MAPE), обчислювальна ефективність (час виконання, обсяг використаної пам'яті) та придатність моделі для конкретних типів навантажень. Отримані результати засвідчили, що легковагові підходи забезпечують прийнятну точність прогнозування (RMSE 0,0621–0,0846) за мінімальних обчислювальних витрат (0,43–11,76 мс на прогноз). Серед протестованих алгоритмів SMA показала найвищу ефективність для стабільних навантажень, модель Гольта – Вінтерса була найрезультативнішою для періодичних шаблонів, фільтр Калмана – для імпульсних і змішаних, а перцентильний метод виявився доцільним для довгострокових прогнозів за умов високої волатильності. Додатково встановлено, що агрегація даних на рівні сервісу істотно знижує похибку за імпульсних навантажень. На основі оцінювання досліджених методів запропоновано метод вибору оптимальних легковагових моделей для набору значень горизонту прогнозування, типів навантаження та способу автомасштабування мікросервісів.

Keywords: *автомасштабування мікросервісів, прогнозні моделі, класифікація навантаження.*