Workloads Prediction Methods for Proactive Resource Scaling in Kubernetes

Oleksandr Rolik and Vitalii Omelchenko

National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute", Kyiv, Ukraine

Abstract
The article considers the issue of predicting workloads in a cluster for use in the proactive scaling of computing resources. Although classical prediction methods have a sufficient level of accuracy, their use on the scale of hundreds of different workloads requires manual data preprocessing and model tuning. The new generation of prediction methods is more versatile, including those capable of independently detecting seasonalities, trends, and anomalies. The paper considers applying these methods to provide accurate predictions for workloads without significant manual intervention. Given the current trend of using microservice architecture, where there are many unique workloads, this attribute can be helpful. Numerous research papers focus on the subject of proactive scaling, exploring statistical approaches and artificial intelligence-based methods. However, most of these studies primarily assess the accuracy of the models while overlooking an essential aspect, which is universality. Universality refers to a model's capacity to handle diverse workload patterns without requiring manual adjustments. The primary focus is to investigate the feasibility of using these methods as a comprehensive solution for automated scaling.

Keywords [1]
Network architecture, deep learning, training, small data, hardware

1. Introduction

The emergence of orchestrators such as Kubernetes, Nomad, and others has dramatically simplified many aspects of computing resource management and made significant changes in approaches to infrastructure development, mainly through the introduction of the containerization paradigm [1].

Containers can significantly reduce the application's availability time compared to virtualization and optimize resource utilization while improving application performance [2]. This is achieved due to the absence of a guest operating system and using cgroups to manage allocated resources.

Orchestration and containerization gave impetus to the development and popularization of microservice architecture. A microservice is a lightweight application whose functionality follows the principle of single responsibility. This separation of functions makes it possible to scale system components separately and allocate cluster resources more granularly.

In particular, these solutions, together with the decomposition of the system into microservices, provide significant opportunities for automating resource management, including computing. Scaling, in particular automatic scaling, is one of the most effective tools for managing the cluster's computing resources and maintaining the required level of service quality. Individual applications, groups of applications, or the entire cluster can be scaled horizontally and vertically. Scaling approaches can be reactive, proactive, and hybrid, which includes components of both previous approaches [3].

2. Related work

Proactive scaling approaches can be broadly divided into time series based and machine learning based. Time series approaches are easier to interpret and do not require a lot of time and computational

International Scientific Symposium "Intelligent Solutions" (IntSol-2023), September 27-28, 2023, Kyiv-Uzhhorod, Ukraine
EMAIL: arolick@gmail.com (A.1); vitaly.om25@gmail.com (A.2)
ORCID: 0000-0001-8829-4645 (A.1); 0000-0002-3850-6555 (A.2)

resources. The main drawback of the approach is the prediction accuracy, that relies heavily on the selected metric and how well historical data is pre-processed.

In the work [4], the authors proposed a solution for proactive scaling based on ARIMA, using the Hyndman-Khandakar algorithm for more accurate selection of model parameters. The accuracy of the model was tested on the example of web requests to the Wikipedia server, obtaining an accuracy rate of 91%. It is worth noting that ARIMA does not support work with complex seasonality and the authors evaluated the model on the weekly load pattern.

In another paper [5], the authors presented a solution that allows combining multiple time series forecasting algorithms (Simple Exponential Smoothing, Moving Average, ARMA, Holt-Winters) using a genetic algorithm. This work shows that there is no best algorithm for all existing time series and that a combination of such methods can be more accurate than each of them individually.

ML-based are able to detect nonlinear features of systems, but require a lot of time and data to train the model. In the work devoted to Google Autopilot [6], it is noted that the predictions are produced with the help of several ML models, which, in addition to historical data of computing resources usage, are able to include such events as OOM and CPU throttling, in the forecasting process. The authors also point out that one of the problems of this approach is the interpretation and the possibility of explaining the predictions.

3. Predictive scaling and Kubernetes

Firstly, we must consider the disadvantages of reactive scaling, as discussed in our previous work [7]. With reactive scaling, there is a delay between when the load increases and when additional resources become available. During this delay, the application may experience performance degradation or unavailability. In addition, reactive scaling does not work effectively with sudden or unpredictable load peaks. Also, this type of scaling can be resource-intensive and inefficient if the load changes rapidly. Suppose the limits for reactive scaling are set too sensitive. In that case, you may end up with a system constantly scaling up and down, resulting in suboptimal resource utilization and degraded QoS. Predictive scaling can compensate for these shortcomings by predicting peak loads [8]. However, it should be considered that in the case of atypical loads, reactive scaling can be more effective [9]. A predictive approach is practical when a seasonal load pattern persists for a long time. It makes it possible to scale up the application in time and maintain a high level of QoS, and on the other hand, to free up resources when they are no longer needed.

3.1. Requirements to prediction methods

This paper is devoted to studying the feasibility of using the selected models for prediction in conditions close to working in a Kubernetes cluster, considering all its limitations and capabilities.

In this paper, it is assumed that the load pattern of any component has a clear seasonality or trend. Otherwise, there is no point in applying this approach. Therefore, the first requirement for the predictive models considered in this paper is the ability to work with one or more seasonality. Also, the models considered in this paper should be accurate since the more accurate the forecast, the more optimally the amount of computing resources is calculated. Given the current trend of microservices, scaling all possible cluster components makes sense. Also, given that each component of the microservice architecture has its unique features and functionality, the load pattern is individual for each component, and there can be an unlimited number of such components. This leads to the conclusion that processing historical metrics and manually adjusting model parameters for each component is a process that cannot be scaled. The main advantage and feature of predictive models is that it is possible to estimate the load at any point in the future. This means that we can adapt the target subsystem to future loads in terms of performance and saving computing resources. However, predictions in the context of workloads are only sometimes accurate for many reasons. The workload on a subsystem depends on many technical and non-technical factors, such as network stability, data center availability, holidays, and even political and economic situations. No model can account for all of these factors, but it can be made adaptive and resilient to such things. If a model is very accurate but takes too long to train, its effectiveness also decreases. It is necessary to formulate the requirements for predictive models taking into account all of the above:

1. Versatility
2. High accuracy of predictions
3. Work with complex seasonality

3.1.1. Architecture of Kubernetes

Kubernetes, as an open-source platform, facilitates the management of workloads and applications. It offers automation for load balancing, application deployment, scaling, data storage management, and access control. A cluster in Kubernetes refers to a collection of virtual or physical machines connected within a single network. This coherence is achieved through specialized software on each machine called kubelet agent. In the context of a Kubernetes cluster, each machine is considered a node. Each node is allocated specific computing resources as part of the resource management subsystem.

During application deployment, each instance is assigned to a node with sufficient computing resources to ensure proper functionality. The deployment specification includes minimum resource requirements, known as requests and limits. While a containerized application instance can utilize more resources than the specified requests if available on the node, it is restricted from using more resources than the limits configuration defines. This ensures efficient resource utilization and management within the Kubernetes cluster. In this paper, we consider scaling automation in two stages. The first stage involves obtaining historical data, processing it, obtaining resource or workload predictions, and validating them. The second stage involves determining the point in time when applying new resource constraints will have the most negligible impact on quality indicators. This paper considers only a part of the first stage, namely obtaining predictions.

3.1.2. Kubernetes limitations

Kubernetes uses a ready-made solution for resource monitoring - Prometheus, which has its specifics. In this paper, we will use this solution as a data source. First, it has a relatively short storage time for historical metrics, usually several weeks. Second, metrics can be lost due to system failures. Third, the frequency of metrics collection is approximately one minute due to the limitations of kubelet [10]. Predicting memory values has its specifics because if the limit is exceeded, a denial of service due to OOM may occur. This paper does not cover this feature.

3.2. Prediction methods

There are several predictive models, but we have chosen to analyze relatively new approaches that meet the above requirements in this paper – TBATS, Prophet and NeuralProphet.

3.2.1. TBATS

The first forecasting method considered in this article is TBATS [11], whose name is an acronym for the main components of the model: trigonometric seasonality, Box-Cox transformation, ARIMA, trend, and seasonality components. TBATS is designed to forecast complex time series with several seasonalities of different lengths. In TBATS, the original time series is subjected to the Box-Cox transformation [12], the primary purpose of which is to make the variance of the data stable, which is a crucial assumption for many statistical models, especially for linear regression and time series models. After that, the transformed time series is modeled as a linear combination of an exponentially smoothed trend, seasonal components, and ARMA components. Seasonal components are modeled by trigonometric functions using a Fourier series. TBATS is capable of adjusting some parameters independently using AIC [13]. This method was chosen for this work because of its versatility and ability to adapt to time series of varying complexity, which means there is no need to adapt the model to the load of each existing component in the system.

$$y_t^{(\lambda)} = l_{t-1} + \phi b_{t-1} + \sum_{i=1}^{T} s_{t-m_i}^{(i)} + d_t \ , \qquad (1)$$

where $y_t^{(\lambda)}$ – time series at moment *t*, $s_{t-m_i}^{(i)}$ - seasonal components, l_{t-1} - local level, b_{t-1} - trend with damping, d_t - ARMA(p, q) process for residuals. Each seasonal component described with the following dependence:

$$s_{j,t}^{(i)} = s_{j,t-1}^{(i)} \cos(\omega_i) + s_{j,t}^{*(i)} \sin(\omega_i) + \gamma_1^{(i)} d_t \ , \qquad (2)$$

where $\omega_j = 2\pi j / m_i$, m_i - length of i-th seasonal period, $\gamma_1^{(i)}$- seasonal smoothing of i-th seasonality.

3.2.2. Prophet

Prophet is a time series forecasting library developed at Facebook [14]. The main goal of the development was to create a simple, transparent, and understandable model-generation algorithm that would allow for quick and reliable predictions.

This algorithm is based on an additive regression model [15] with several components.

$$y(t) = g(t) + s(t) + h(t) + e(t), \qquad (3)$$

where $g(t)$ - trend component, $s(t)$ - seasonal component, $h(t)$ - anomaly component and $e(t)$ - an error function. In addition to the additive regression model, Prophet also uses the Fourier transform.

Among the advantages of this model are the ability to work with any time series, the ability to work efficiently with large data sets and missing data, and flexibility in customization.

3.2.3. NeuralProphet

NeuralProphet [16] is a time series prediction library developed on top of the PyTorch library. This library develops Facebook's Prophet model but uses neural networks. The main difference of this library is the ability to use the power of deep learning to predict time series with different trends and seasonality.The basis of this library is an autoregressive neural network [17], which combines classical autoregressive time series prediction methods with modern approaches based on artificial intelligence.

This library consists of several components: trend, seasonality, regression of future variables, autoregression of historical variables, and regression of lagged variables. The following equation can describe their dependence:

$$\hat{y}_t = T(t) + S(t) + E(t) + F(t) + A(t) + L(t), \qquad (4)$$

where $T(t)$ - trend function, $S(t)$- seasonal function, $E(t)$ - event and holiday function, $F(t)$ - regression effects for future-known exogenous variables, $A(t)$- auto-regression effects based on past observations, $L(t)$- regression effects for lagged observations of exogenous variables [16].

This library lies at the intersection of static and neural network methods and has the advantages of both approaches. In addition, NeuralProphet includes the ability to select component parameters automatically, has the ability to work with time series of different seasonality, and allows the use of side variables to improve the results of predictions.

3.2.4. Other methods

In this paper, we do not consider SARIMA since this method does not support the work with several seasonalities in one time series. Also, finding the optimal model parameters is a non-trivial task. Also, this paper does not consider the LTSM method, as it has been considered in many papers. In addition, approaches that rely entirely on neural networks are difficult to understand and require a significant understanding of neural network architecture.

4. Experiments

This section is devoted to conducting practical experiments to evaluate the accuracy of the models under certain conditions. The purpose of these experiments is not only to compare the selected methods with each other and determine the most accurate ones but also to assess the feasibility of using these methods in general.

4.1. Accuracy evaluation

To evaluate the accuracy of time series prediction models, two accuracy metrics are suitable:

- Root mean square error (RMSE);
- Mean absolute percentage error (MAPE).

RMSE allows you to compare the deviations in the initial values and helps you assess the prediction's overall accuracy.

$$RMSE = \sqrt{\frac{\sum_{t=1}^{n}(\hat{x}_t - x_t)^2}{n}}, \tag{5}$$

where $x(t)$ – an actual value at the moment of time t, $\hat{x}_t$ – prediction at the moment of time t, and n – a number of datapoints in the dataset.

MAPE allows you to compare the predictions of different models at different scales or data.

$$MAPE = \frac{100}{n}\sum_{t=1}^{n}\left|\frac{\hat{x}_t - x_t}{x_t}\right|, \tag{6}$$

where $x(t)$ – an actual value at the moment of time t, $\hat{x}_t$ – prediction at the moment of time t, and n – a number of datapoints in the dataset.

The assessment of the universality of the method is determined based on the assessment of accuracy without additional adjustment of the models. Determining how the model can adapt to various load patterns is essential.

4.2. Typical workloads

To compare the selected time series prediction methods, it is necessary first to identify typical application load patterns [18]. Figure 1 shows graphs of typical loads:

- Monotonically increasing;
- on/off patters;
- bursty pattern;
- random pattern;
- mix of several patterns;

The random pattern is not considered in this paper, as predictions of this type are impossible. All other patterns have either a steady trend or some seasonality. This is the basis for the load patterns selected for training.

4.1. Dataset

To verify the accuracy and versatility of the models, we chose the metrics of an artificially generated load that is stationary with daily and weekly seasonality. The data for training and validation were obtained from Prometheus. This combination of seasonality was chosen because it reflects the cyclical nature of human behavior in society, which is driven by social norms and habits. In a typical week, especially in a business environment, a distinct rhythm is caused by working days and days off. Daytime seasonality, in turn, reflects the repetition of people's actions throughout the day: working hours, time for rest, sleep, and so on. Generally, any other seasonality can be selected, and the goal is to test the models on complex seasonality.

The frequency of the values in the generated time series is one hour and has a minimal impact on the accuracy of the models compared to lower data frequencies. Anomalies and distortions are introduced by artificially distorting the data.

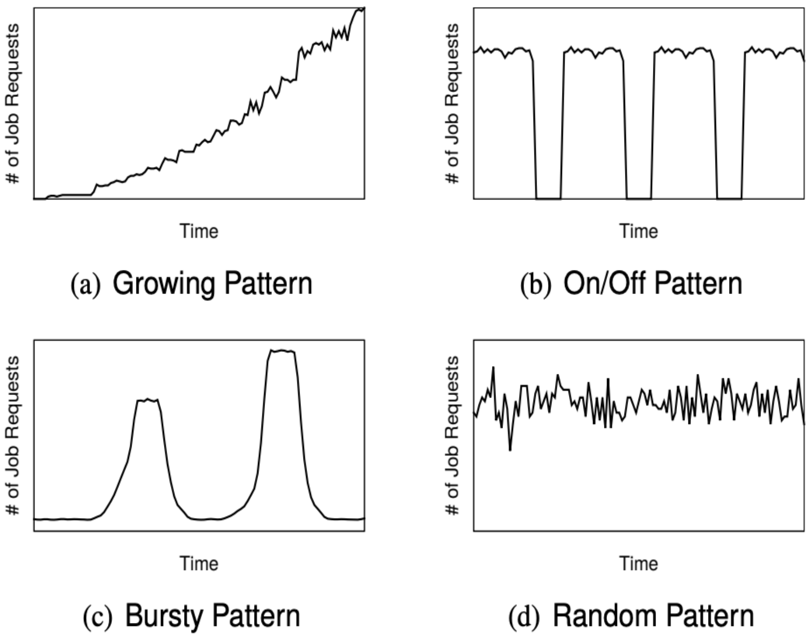

Figure 1: Classification of typical load patterns

4.2. Experiment: daily and weekly fluctuations without data distortion

In the first experiment, we compare the selected models on the example of the above-described time series with two periodicities of different lengths - daily and weekly. The data were not pre-processed. The purpose of this experiment is to investigate the prediction capabilities of the selected models on complex load patterns without any data distortion and to investigate the effect of the size of historical data during training on prediction accuracy. The models are trained on datasets of different lengths, including 1, 2, and 3-week periods.

Based on the graphs and accuracy values, we can conclude that each model accurately predicts the load under these conditions.

However, it is worth noting that TBATS is 6% more accurate than the other models that showed the same result. It is worth noting that the minimum number of periods of historical data to detect seasonality and, accordingly, anomalies are two periods. Therefore, it makes sense to test the behavior of these methods on short historical data. Reducing the duration of training data had almost no effect on accuracy in the case of TBATS and Prophet, but NeuralProphet's accuracy deteriorated by 44%, although it is still quite high.

Table 1
Prediction of daily and weekly fluctuations without data distortion based on a three-week dataset

Model	RMSE	MAPE
TBATS	32.464	0.081
Prophet	46.16	0.0856
NeuralProphet	53.705	0.0891

This experiment is also conducted on the basis of a non-stationary time series with an existing upward trend.

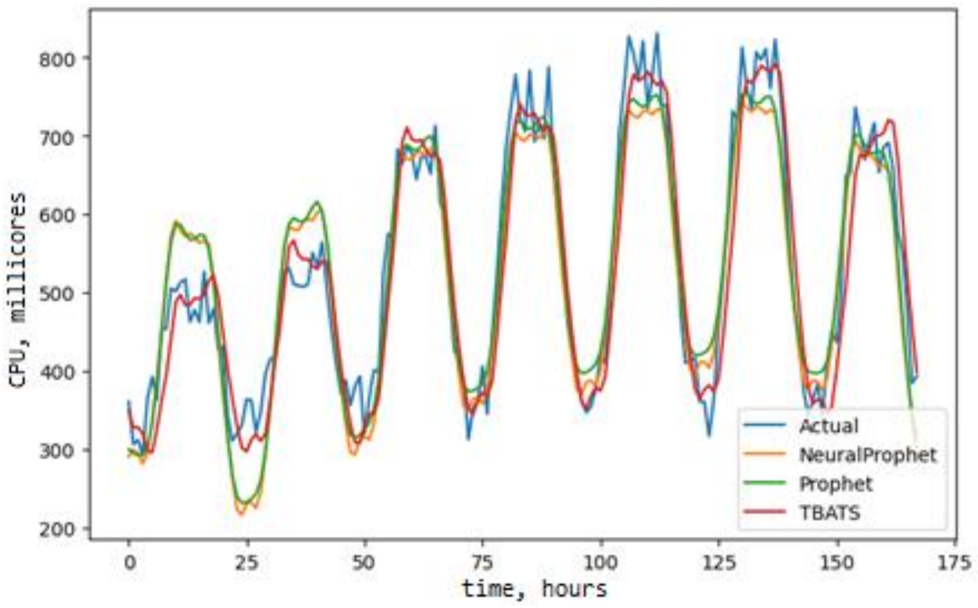

Figure 2: Prediction of daily and weekly fluctuations without data distortion based on a three-week dataset

Table 2. Prediction of daily and weekly fluctuations without data distortion based on a two-week dataset

Model	RMSE	MAPE
TBATS	32.943	0.084
Prophet	51.236	0.089
NeuralProphet	71.378	0.130

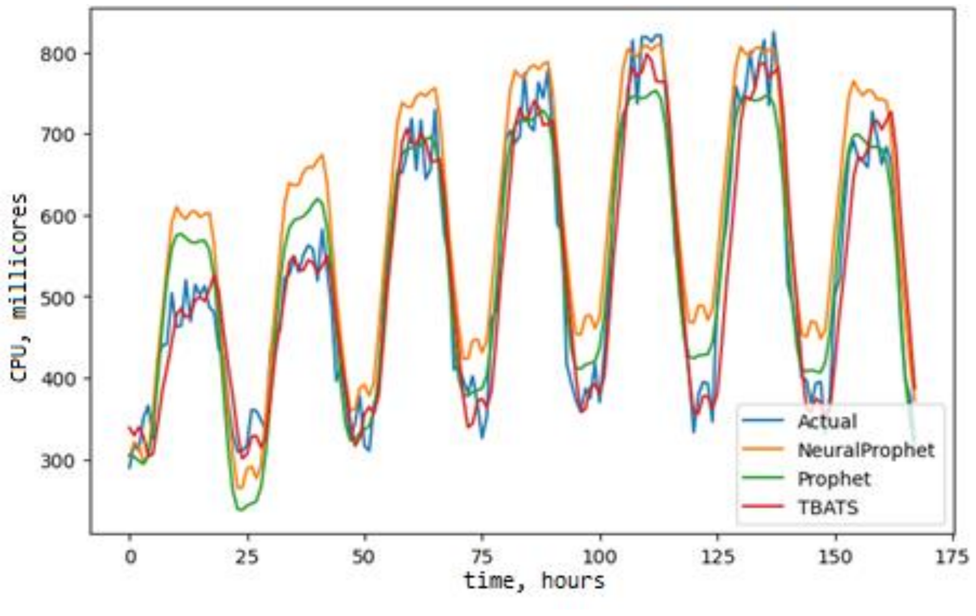

Figure 2. Prediction of daily and weekly fluctuations with two input periods

Table 3. Prediction of daily and weekly fluctuations without data distortion based on a weekly dataset

Model	RMSE	MAPE
TBATS	32.943	0.084
Prophet	51.236	0.089
NeuralProphet	71.378	0.130

Table 4. Predicting daily and weekly fluctuations without data distortion based on a three-week trending dataset

Model	RMSE	MAPE
TBATS	50.17	0.087
Prophet	73.307	0.101
NeuralProphet	70.156	0.093

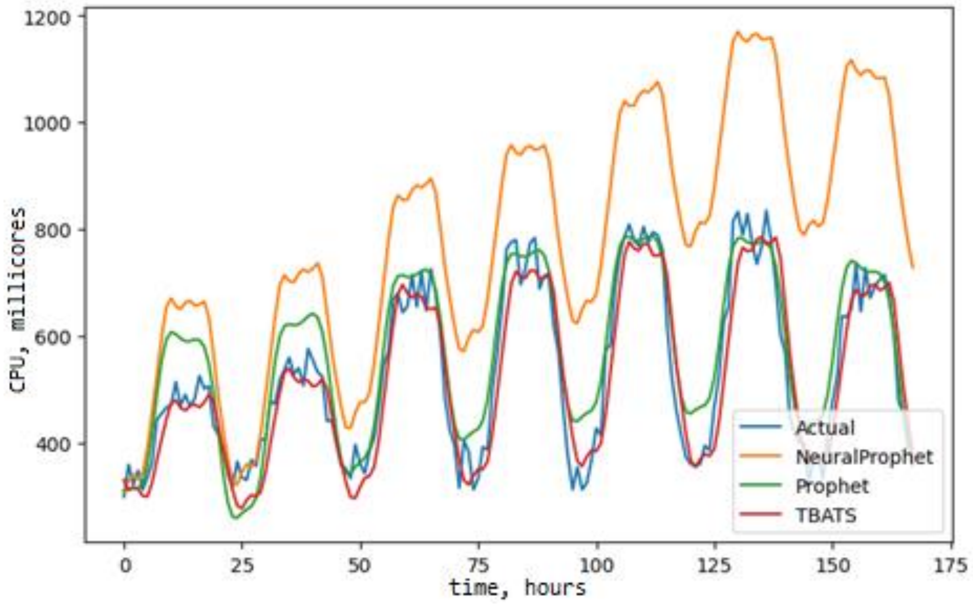

Figure 3. Prediction of daily and weekly fluctuations with one input period

4.3. Experiment: daily and weekly fluctuations with anomalies

In the next experiment, distortions are included in the historical data. Some of the days have atypical increased or decreased values. In real information systems, such distortions can be caused by holidays, network equipment failure, or load-balancing problems.

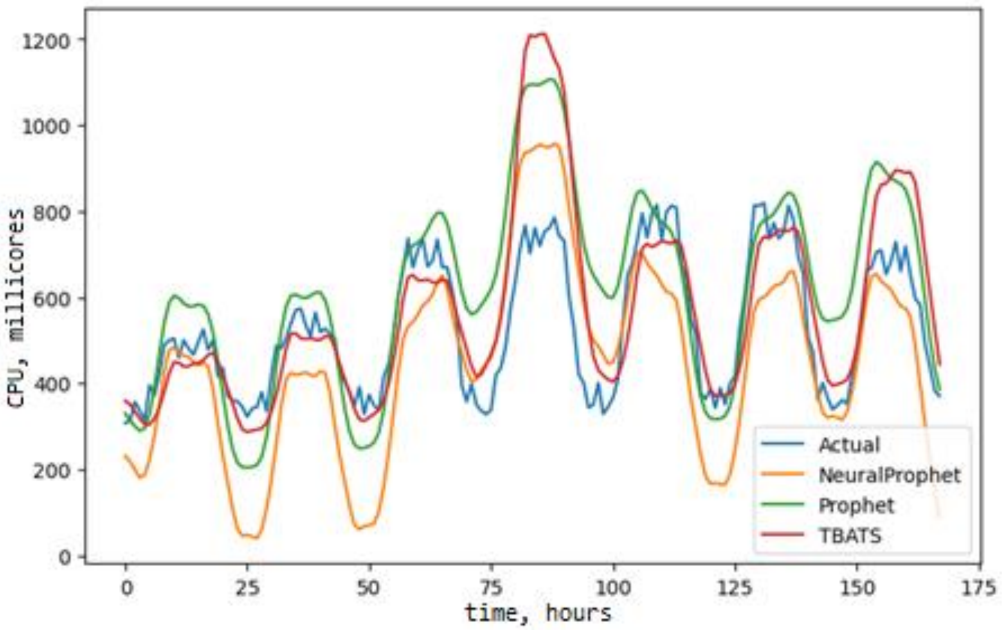

Figure 5. Prediction of daily and weekly fluctuations with three input periods and 10% of days with anomalies

Table 4
Prediction of daily and weekly fluctuations with three input periods and 10% of days with anomalies

Model	RMSE	MAPE
TBATS	135.949	0.174
Prophet	163.716	0.263
NeuralProphet	161.726	0.297

On accuracy compared to the first experiment is halved in this experiment, with Prophet and NeuralProphet's accuracy decreasing by more than 200%.

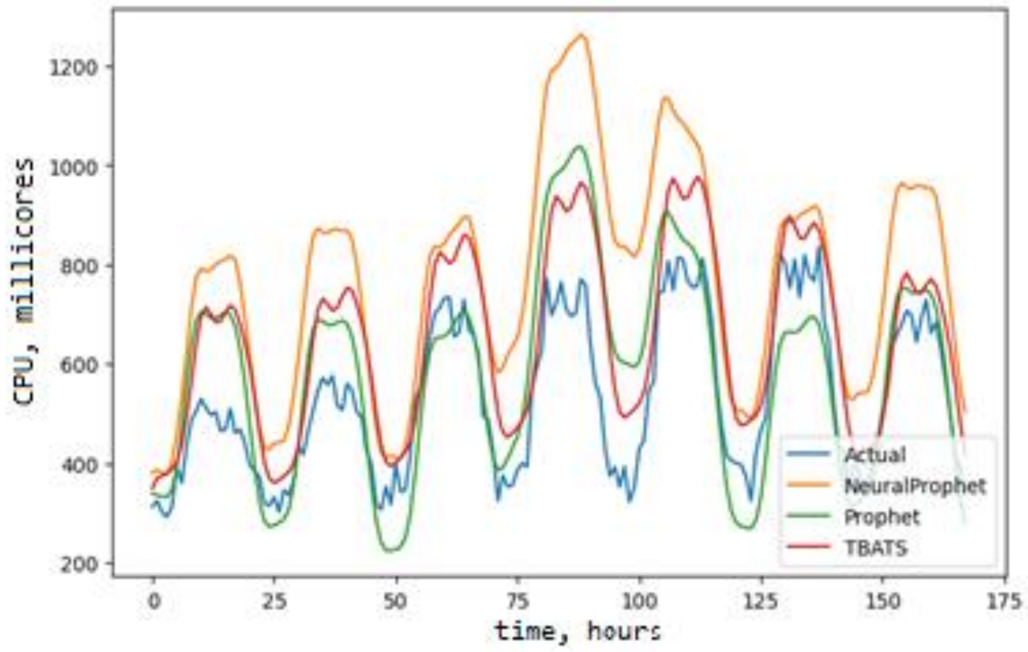

Figure 5. Prediction of daily and weekly fluctuations with three input periods and anomalies in 20% of days

Table 6.
Prediction of daily and weekly fluctuations with three input periods and anomalies in 20% of days

Model	RMSE	MAPE
TBATS	134.313	0.243
Prophet	134.848	0.209
NeuralProphet	274.135	0.476

In such extreme conditions, the accuracy of predictions decreases significantly. In particular, the accuracy of the NeuralProphet model, in this case, is critically low, but the general pattern is nevertheless preserved.

5. Conclusions

The results of the experiments show that, in general, all three selected models are able to predict complex load patterns with several seasonalities without preliminary data processing and with anomalies quite accurately. TBATS is more accurate than Prophet and NeuralProphet in all the experiments, but the difference in accuracy is no more than 16%. TBATS and Prophet predicted the load quite accurately with only one week of data, and the accuracy degradation is only 10%. NeuralProphet requires additional parameter settings or data pre-processing in some of the experiments. Anomalies significantly affect the accuracy, and pre-processing of historical data is necessary.

Given the results and requirements, all three selected models can be used to automate resource scaling in Kubernetes. However, it is necessary to consider the features and prerequisites for their use.

6. References

[1] O. I. Rolik, S. F. Telenyk, and M. V. Yasochka, "Управление корпоративной инфраструктурой," Kyiv: Наукова Думка, 2018, 576 p.

[2] J. Bhimani, Z. Yang, M. Leeser, and N. Mi, "Accelerating Big Data Applications Using Lightweight Virtualization Framework on Enterprise Cloud," in Proceedings of the High Performance Extreme Computing Conference (HPEC), 2017, pp. 1-7.

[3] C. Qu, R. N. Calheiros, and R. Buyya, "Auto-Scaling Web Applications in Clouds," ACM Computing Surveys, vol. 51, no. 4. Association for Computing Machinery (ACM), pp. 1–33, Jul. 13, 2018. doi: 10.1145/3148149.

[4] R. N. Calheiros, E. Masoumi, R. Ranjan, and R. Buyya, "Workload Prediction Using ARIMA Model and Its Impact on Cloud Applications' QoS," IEEE Transactions on Cloud Computing, vol. 3, no. 4. Institute of Electrical and Electronics Engineers (IEEE), pp. 449–458, Oct. 01, 2015. doi: 10.1109/tcc.2014.2350475.

[5] V. R. Messias, J. C. Estrella, R. Ehlers, M. J. Santana, R. C. Santana, and S. Reiff-Marganiec, "Combining time series prediction models using genetic algorithm to autoscaling Web applications hosted in the cloud infrastructure," Neural Computing and Applications, vol. 27, no. 8. Springer Science and Business Media LLC, pp. 2383–2406, Dec. 12, 2015. doi: 10.1007/s00521-015-2133-3.

[6] K. Rzadca et al., "Autopilot," Proceedings of the Fifteenth European Conference on Computer Systems. ACM, Apr. 15, 2020. doi: 10.1145/3342195.3387524.

[7] V. Omelchenko and O. Rolik, "Автоматизація управління ресурсами в інформаційних системах на основі реактивного вертикального масштабування," Адаптивні системи автоматичного управління, vol. 2, no. 41. Kyiv Politechnic Institute, pp. 65–78, Dec. 01, 2022. doi: 10.20535/1560-8956.41.2022.271344.

[8] J. Santos, T. Wauters, B. Volckaert, and F. D. Turck, "gym-hpa: Efficient Auto-Scaling via Reinforcement Learning for Complex Microservice-based Applications in Kubernetes," NOMS 2023-2023 IEEE/IFIP Network Operations and Management Symposium. IEEE, May 08, 2023. doi: 10.1109/noms56928.2023.10154298.

[9] M. Straesser, J. Grohmann, J. von Kistowski, S. Eismann, A. Bauer, and S. Kounev, "Why Is It Not Solved Yet?," Proceedings of the 2022 ACM/SPEC on International Conference on Performance Engineering. ACM, Apr. 09, 2022. doi: 10.1145/3489525.3511680.

[10] Metrics For Kubernetes System Components. Kubernetes Documentation. URL: https://kubernetes.io/docs/concepts/cluster-administration/system-metrics/.

[11] A. M. De Livera, R. J. Hyndman, and R. D. Snyder, "Forecasting Time Series With Complex Seasonal Patterns Using Exponential Smoothing," Journal of the American Statistical Association, vol. 106, no. 496. Informa UK Limited, pp. 1513–1527, Dec. 2011. doi: 10.1198/jasa.2011.tm09771.

[12] G. E. P. Box and D. R. Cox, "An Analysis of Transformations," Journal of the Royal Statistical Society, Series B, vol. 26, no. 2, pp. 211–252, 1964.

[13] G. Skorupa, "Forecasting Time Series with Multiple Seasonalities using TBATS in Python". 2019. URL: https://medium.com/intive-developers/forecasting-time-series-with-multiple-seasonalities-using-tbats-in-python-398a00ac0e8a.

[14] S. J. Taylor and B. Letham, "Forecasting at scale." PeerJ, Sep. 27, 2017. doi: 10.7287/peerj.preprints.3190v2.

[15] J. H. Friedman and W. Stuetzle, "Projection Pursuit Regression," Journal of the American Statistical Association, vol. 76, pp. 817–823, 1981.

[16] O. Triebe, H. Hewamalage, P. Pilyugina, N. Laptev, C. Bergmeir, and R. Rajagopal, "NeuralProphet: Explainable Forecasting at Scale." arXiv, 2021. doi: 10.48550/ARXIV.2111.15397.

[17] O. Triebe, N. Laptev, and R. Rajagopal, "AR-Net: A simple Auto-Regressive Neural Network for time-series." arXiv, 2019. doi: 10.48550/ARXIV.1911.12436.

[18] W. Veenhof, "Workload patterns for cloud computing," URL:

[19] http://watdenkt.veenhof.nu/2010/07/13/workload-patterns-for- cloud-computing/

[20] K. Kim, W. Wang, Y. Qi, and M. Humphrey, "Empirical Evaluation of Workload Forecasting Techniques for Predictive Cloud Resource Scaling," 2016 IEEE 9th International Conference on Cloud Computing (CLOUD). IEEE, Jun. 2016. doi: 10.1109/cloud.2016.0011.

S. Fan and R. J. Hyndman, "Short-Term Load Forecasting Based on a Semi-Parametric Additive Model," IEEE Transactions on Power Systems, vol. 27, no. 1. Institute of Electrical and Electronics Engineers (IEEE), pp. 134–141, Feb. 2012. doi: 10.1109/tpwrs.2011.2162082.