J Grid Computing (2023) 21:6
https://doi.org/10.1007/s10723-022-09641-y

Data-Driven Adaptive Prediction of Cloud Resource Usage

Piotr Nawrocki · Patryk Osypanka · Beata Posluszny

Received: 23 August 2022 / Accepted: 14 December 2022 / Published online: 3 January 2023

Abstract Predicting computing resource usage in any system allows optimized management of resources. As cloud computing is gaining popularity, the urgency of accurate prediction is reduced as resources can be scaled on demand. However, this may result in excessive costs, and therefore there is a considerable body of work devoted to cloud resource optimization which can significantly reduce the costs of cloud computing. The most promising methods employ load prediction and resource scaling based on forecast values. However, prediction quality depends on prediction method selection, as different load characteristics require different forecasting mechanisms. This paper presents a novel approach that incorporates data-driven adaptation of prediction algorithms to generate short- and long-term cloud resource usage predictions and enables the proposed solution to readjust to different load characteristics as well as both temporary and permanent usage changes. First, preliminary tests were performed that yielded promising results – up to 36% better prediction quality. Subsequently, a fully autonomous, multi-stage optimization solution was proposed. The proposed approach was evaluated using real-life historical data from various production servers. Experiment results demonstrate 9.28% to 80.68% better prediction quality when compared to static algorithm selection.

Keywords Cloud computing · Resource usage prediction · Machine learning · Adaptation

P. Nawrocki (✉) · P. Osypanka · B. Posluszny
Institute of Computer Science, AGH University of Science and Technology, al. A. Mickiewicza 30, 30-059 Krakow, Poland
e-mail: piotr.nawrocki@agh.edu.pl

P. Osypanka
e-mail: patryk.osypanka@agh.edu.pl

B. Posluszny
e-mail: bposluszny@student.agh.edu.pl

P. Osypanka
ASEC S.A., ul. Wadowicka 6, 30-415 Krakow, Poland

1 Introduction

Corporate computer systems have frequently shifted to cloud computing environments, with the COVID-19 pandemic providing a strong boost to this trend. According to *Flexera 2021 State of the Cloud Report*, three-fourths of companies using cloud environments spend more than a million dollars annually on their operation. Public clouds offer storage, network and computing resources with virtually unlimited scaling capabilities. Large scaling safety margins are often adopted to avoid system unavailability during sudden load changes, but this may result in unnecessary cloud resource allocation and therefore, as stated in the aforementioned report, generate additional costs and electricity usage. As estimated in [4], in the near future data centers will use a considerable percentage

of global electricity, and thus reducing cloud resource usage helps protect the environment.

The authors propose a system which will precisely predict optimized usage and use this for accurate cloud resource scaling. To ensure the accuracy of both short- and long-term forecasts, the system proposed may require dynamic prediction algorithm selection. Therefore, the authors performed preliminary tests to investigate whether prediction algorithm adaptation improves forecast quality. Based on the results obtained, a Self-Adapting Data-Driven Prediction System (SA-DDPS) was defined to optimize cloud resource usage with prediction algorithm adaptation based on data characteristics. The solution proposed is suitable for optimizing cloud resources for various kinds of systems with variable load characteristics, as it automatically determines the optimal prediction algorithm based on load analysis, predicts usage levels, creates resource provisioning plans, and performs resource scaling. The major contributions of this paper can be summarized as follows:

- performing preliminary tests confirming the validity of the solution proposed;
- designing a solution that provides resource usage predictions without any prior configuration or knowledge of load characteristics;
- developing a self-adapting data-driven system that can operate in production-grade environments with short- and long-term load changes;
- conducting evaluation using data collected from a real-life production system and presenting the results with tests performed for different load types, and demonstrating the importance of data-driven adaptation;
- comparing the results obtained using other prediction techniques, illustrating the improved prediction quality achieved.

The rest of this paper is structured as follows: Section 2 contains a description of various cloud resource forecasting methods, Section 3 presents preliminary research, Section 4 is concerned with defining a data-driven adaptive prediction approach, Section 5 describes the tests conducted on the proposed solution, and Section 6 contains the summary and further work.

2 Related Work

The literature describes various cloud resource forecasting methods, as such forecasting is vital for proactive cloud resource usage optimization. In this section, the most common techniques are presented and compared with the proposed solution.

One popular prediction method is incoming request modelling. A system based on online incremental learning is presented in [12], Ranjbari et al. [26] describe learning automata, and time-series with queuing theory are used by the authors of [8]. One of the shortcomings of incoming request modelling becomes manifest when it is used with anomalous data which include random changes. On the other hand, the authors of [31] presented a method based on time-series analysis and tested it using real-life Quality of Service (QoS) data from [15]; nevertheless, their load data were artificially generated. This approach is prone to temporary deviations. On the other hand, the SA-DDPS proposition, which uses forecasting techniques supported by Machine Learning, is capable of operating with industry-grade loads.

Other resource prediction methods are based on the fact that incoming traffic can be treated as time series. In [20], a comparison of Recurrent Neural Networks (RNNs) and their successor Long Short-Term Memory (LSTM) is presented. The dataset was provided by a private company and included information about the activity of company employees entering the intranet. Although LSTM outperforms RNNs, forecasts only 30 minutes ahead were achieved, with the dataset including values from just two months. In contrast, the authors use data from an entire year to explore how models cope with such long periods and if they can adapt to changes successfully. Likewise, Sriram N. Rao et al. [27] presented a comparison of the Holt-Winters, Auto Regressive Integrated Moving Average (ARIMA), and LSTM models, but only one-step forecasting was provided, which is not sufficient in the context of long-term prediction.

A distinct approach to cloud resource allocation is described in [14], where the authors use Random Forest to select the best resource offered by a provider in order to satisfy the users' requirements. Furthermore, the authors of [2] and [13] proposed a novel scheduling of scientific workflows, while in [19] QoS and

cost optimization of cloud resource allocation is discussed. Sung et al. [28] describe Optimized Memory Bandwidth Management Machine Learning to manage resources for latency-critical workloads and the authors of [1] describe an algorithm that evaluates resource utilization requirements for incoming tasks. Although this method does not require load prediction, the solution must know the incoming requests' resource demands, which may create a big disadvantage where load data are not well defined. On the other hand, SA-DDPS handles different types of incoming traffic properly without the need for structural assessment.

Another group of widely used cloud resource prediction methods is based on artificial intelligence techniques. The autoscaling of network resources is proposed in [25], recurrent neural networks are discussed in [10], a general framework for a VM reservation plan supported by Machine Learning and a set of different methods is presented in [33] and the authors of [16] used the Random Forest and ARIMA models. Likewise, Nawrocki et al. in [21] carried out a comprehensive experiment with the use of Multilayer Perceptron, the authors of [32] use a deep learning model to predict resource utilization, and Shuai Wang et al. in [30] propose a stacking strategy that integrates models such as LSTM, Random Forest, linear regression and Gaussian process regression. The solutions proposed convert timestamps into a vector of one-hot encoded information on features such as days of the week, seasons or holidays. This may be beneficial with specific traffic patterns, but requires data analysis and is prone to changes in those characteristics. The proposed solution not only is capable (as it does not rely on any additional features) of handling diverse data characteristics properly, but it also adapts to changing environments by monitoring incoming traffic attributes and selecting suitable prediction algorithms.

As sub-optimal resource prediction may lead to a deterioration in optimized system responsiveness and the quality of the service provided, a significant body of work has been devoted to QoS-driven resource allocation. One of the approaches to this problem is to allocate incoming requests to the resources available in the most optimal way. This method allows a timely response just before the defined timeout. The solutions proposed in [7] and [17] combine incoming task analysis with appropriate cloud resources. Likewise, Chen et al. [6] present an iterative QoS prediction model, and time series prediction is discussed in [29]. Despite its many advantages, this method requires the optimizing solution to assess the incoming requests' resource demands or to know these demands beforehand. The SA-DDPS works autonomously without any prior requirements; additionally, improved prediction quality prevents QoS violations.

Authors' earlier works [22–24] indicate that resource usage optimization may result in significant savings. A multi-stage optimization process with sophisticated data-cleaning, monitoring and scaling mechanisms yielded remarkable resource optimization. Nevertheless, the former solutions used statically selected prediction algorithms, which might result in sub-optimal forecasts in systems with highly dynamic load characteristics. The solution presented in this paper not only performs load prediction and uses it for automatic cloud resource scaling, but also continuously determines historical data characteristics for optimal prediction algorithm selection.

An analysis of existing solutions (Table 1) shows that currently none of them can tackle the problem of short- and long-term prediction for cloud resources allocated to systems with highly varied load data. The need to formalise the load, delayed response to rapid changes and fixed prediction techniques are factors that make them ineffective in industry-grade, noisy and highly diverse load level predictions, especially when QoS constraints are taken into consideration. The contribution of this study is to provide a novel, data-driven adaptive short- and long-term prediction solution that overcomes the aforementioned challenges. The SA-DDPS works with different load types without prior knowledge of incoming data parameters, is resilient to both temporary and permanent usage changes and self-adapts the prediction algorithms utilized. In addition, the SA-DDPS was evaluated with industry-grade test data, demonstrating its capability to function successfully in these conditions.

3 Preliminary Research

As stated above, dynamic prediction algorithm selection may improve forecasting quality. Therefore, the authors decided to perform preliminary research to evaluate this approach. The idea is to pay attention to the specific characteristics of data patterns. Based on

Table 1 Comparison of various parameters in the literature review

Approach	Reference	Methods	Test data		Remarks
			Artificial	Real-life	
Incoming request modelling	[12]	Online incremental learning	X		
	[26]	Learning automata	X		
	[8]	Time-series with queuing theory	X		
	[31]	Time-series analysis	X	X	Real-life QoS data with artificially generated load data
Incoming traffic as time series	[20]	RNNs and LSTM		X	Dataset from Afry, 30 minutes long forecast
	[27]	Holt-Winters, ARIMA and LSTM		X	One-step forecasting
Scheduling incoming task to the available resources	[14]	Random Forest	X		
	[2, 13]	Scheduling of scientific workflows	X		
	[19]	QoS and cost optimization	X		
	[28]	Optimized Memory Bandwidth Management Machine Learning	X		
	[1]	Evaluation of resource utilization	X		
Artificial intelligence techniques	[25]	Autoscaling of network resources	X		
	[10]	RNNs		X	Dataset partially from PlanetLab
	[33]	General framework for a VM reservation plan		X	Dataset from Wikipedia
	[16]	Random Forest and ARIMA		X	Dataset from EMPRES-I
	[21]	Multilayer Perceptron		X	Dataset from IPTV
	[32]	Deep learning model		X	Dataset from PlanetLab
	[30]	LSTM, Random Forest, linear regression and Gaussian process regression	X		
QoS-driven resource allocation	[7]	Incoming task analysis		X	Dataset from WSDream
	[17]	Incoming task analysis	X		
	[6]	Iterative QoS prediction model	X		
	[29]	Time series prediction		X	Dataset from Amazon and Google
	[22–24]	Multi-stage optimization process with sophisticated data-cleaning, monitoring and scaling mechanisms		X	Dataset from ASEC, one week long prediction
	SA-DDPS	Multi-stage optimization process with data-driven adaptive prediction		X	Dataset from ASEC, one week long prediction

it, a suitable Machine Learning prediction model is matched and trained. Section 3.1 presents how dataset values passed to the model are converted. Section 3.2 demonstrates how the model selected based on data analysis was determined. Section 3.3 presents preliminary test results which are then summarized in Section 3.4.

3.1 Data Preparation

While working on time-series associated questions, the time component is the factor that makes problems truly difficult to solve. In particular, time-series data should be preprocessed adequately as the order of values is crucial and the data used for evaluation must not

be lost. One solution to that problem is adding additional features which hold information about the day of the week, month, or even hour depending on the needs – on a case-by-case basis. Such a solution is complex, since loads may exhibit multi-day or weekly trends. Apart from that, there are holidays, special events and promotions that affect periodicity. A lot of human analysis is required, and such solutions may prove insufficient for slightly different dependencies such as those in Fig. 1 which presents the datasets used in the evaluation. Moreover, a dataset may become vast since, for some models, one-hot encoding of these values might be needed. As a result, the time required for model preparation increases. To avoid that inconvenience, the authors apply a *sliding-window* technique, which transforms a sequential supervised learning problem into a classical one [9]. With this approach, any seasonalities deviating from standard, commonly known patterns are covered.

In the presented case, the input window has a seven-day length. The output window has a seven-day length, too. Figure 2 illustrates what the input and output windows look like. Such window lengths appear optimal, since for larger inputs, more training data would be required. As concerns larger outputs, errors could grow in the case of a sudden change in the pattern and the model would not detect it immediately, causing *delayed results*.

Apart from that, the authors use multi-output models, which enable multi-step prediction that returns N values for the entire future period at once. This is less computationally expensive than retraining separate models N times as in the *Direct Strategy*. Moreover, the authors wanted to avoid error propagation which occurs in the *Recursive Strategy* when the model predicts a single value N times while being fed with its own predictions from previous steps [3].

3.2 Model Selection

As the subject of this preliminary research is forecasting, prediction is performed on a weekly basis, covering values for the next seven days. The size of the data window is a trade-off between the ability to quickly adjust to persistent changes and the ability to ignore short anomalous load changes. Therefore, the authors train the proposed model on five weeks of data as this ensures a sufficient amount of training data while allowing proper adaptation to rapid data

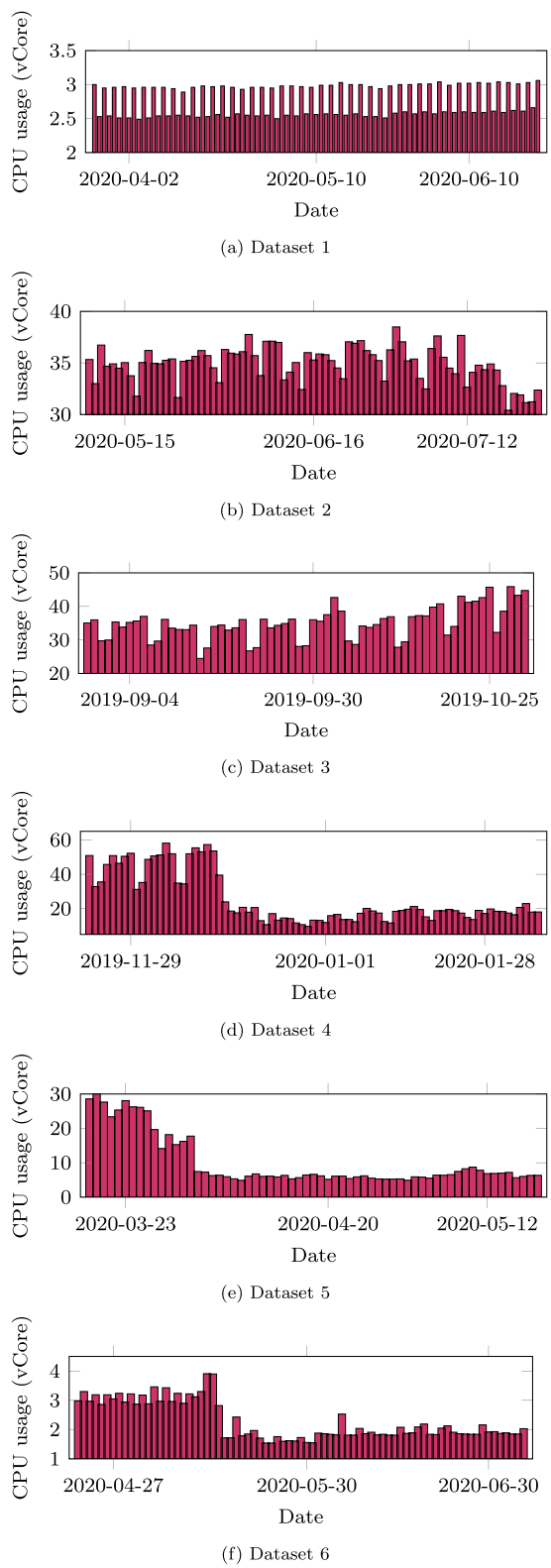


Fig. 1 History data used for evaluation

```
while True do
    mean_filter = mean_filter(data);                /* Calculate mean filter from data */
    f_amplitude = amplitude(mean_filter); /* Calculate amplitude of mean filter values */
    v_amplitude = amplitude(data);     /* Calculate amplitude of values from training data */
    model_index = M(f_amplitude, v_amplitude); /* Determine index of model in vector P based on M metrics */
    model = P[model_index];                  /* Get model based on calculated index */
    window_data = series_to_supervise(data);   /* Transform data into sliding window */
    model.predict(window_data);             /* Predict future values based on reshaped training set */
end
```

Algorithm 1 Data-driven prediction algorithm.

changes. Each row consists of input values from one week and labels with values for the next week. Since usage values are not only features but also labels, the entire training set spans six weeks.

For each week, before actual prediction, a different machine learning model can be utilized. In order to determine the model which is to be used, a data transformation pipeline is prepared. Firstly, vector **P** is calculated:

$$X(d_1, ..., d_n) = \begin{bmatrix} p_1 \\ ... \\ p_m \end{bmatrix} = \mathbf{P} \tag{1}$$

where d_i is the historical resource level. Transformation X provides statistical information about past data. Next, the authors introduce metric M and vector **Y**. Metric M imitates the expert system and is figured out based on p_j:

$$M(\mathbf{P}) = w \tag{2}$$

where $w \in (1, ..., k)$ is simply the index in vector **Y**:

$$\mathbf{Y} = \begin{bmatrix} y_1 \\ ... \\ y_k \end{bmatrix} \tag{3}$$

M defines the index of the model from vector **Y**, whereas vector **Y** holds the mapping between index w and machine learning model – y_w. As a result of this preprocessing, solution can adapt to changes in historical data and pick a model automatically without relying on a single one constantly. Different models can be selected depending on data patterns and these patterns should be detected automatically.

3.3 Preliminary Tests of the Proposed Solution

The tests were conducted in the Python3 language, using common open-source libraries such as NumPy and pandas. The datasets were provided by Polcom.[1] These included daily real-life CPU usage values for Virtual Machines, measured as the number of vCores. Periods are individual months of the year. As a result, it can be compared how models behave under different conditions and loads throughout the year (see Fig. 1).

Basically, the authors start with a five-week training test and perform predictions for the next few weeks. Initially, the authors decided to compare the Linear model, Random Forest, and Boosted Decision Trees to forecast future values as they work well for data with random patterns [5]. As a reference model, the authors prepared the naive mean approach, which simply takes the mean of values from the last week as a prediction for the next week. In other words, the forecast value for each day is the same average value from the last week. The aim was to obtain results that would be at least better than the naive-mean approach.

To compare the models' effectiveness, the authors use the Mean Absolute Percentage Error (*MAPE*) and Root Mean Squared Error (*RMSE*). The former is expressed as a percentage value, which makes comparisons easier for data with different ranges of values.

After performing a number of tests, it turned out that the Linear model and tree-based models alternately yielded better results. After observing these behaviors, the main focus moved to defining the conditions under which each individual models perform

[1] Polcom – https://polcom.com.pl/

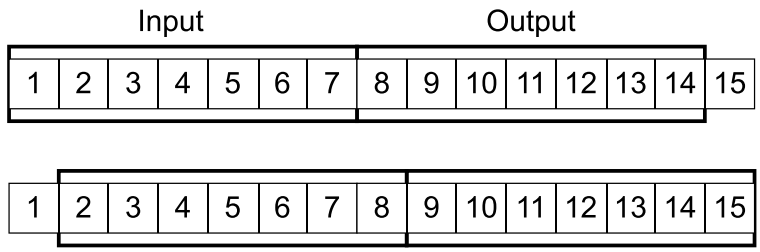


Fig. 2 Sliding window example

the best. Trying to understand these relationships, the authors came to the conclusion that the Linear model quickly adjusted to sudden drops in data. These circumstances could be analyzed automatically, and then the appropriate model could be selected to predict more accurate values.

Finally, the implementation was extended to include automated data analysis and model selection. In particular, the transformation X returns vector $[p_1, p_2]$: p_1 is the amplitude of mean filter values (Moving Average method [11]). It smoothes the plot so that it is resistant to outliers and averages oscillating values. It provides a visualization similar to an analysis conducted by a human to check the *norm* level at which historical values remain. In turn, p_2 is the amplitude of values from the recent training window.

Metric M is introduced as follows:

$$M(p_1, p_2) = \begin{cases} 1, & \text{if } p_1 > p_2 * \alpha \\ 2, & \text{otherwise} \end{cases} \tag{4}$$

If the amplitude of the moving average calculated from training data is greater than the amplitude of training data, the Linear Model is selected; otherwise, Random Forest is selected. Multiplication by an empirically adjusted coefficient is also taken into account. The value of the coefficient is equal to 0.3.

$$\mathbf{Y} = \begin{bmatrix} Linear \\ Random\,Forest \end{bmatrix} \tag{5}$$

For their purposes, the authors selected the Linear model and Random Forests as those algorithms, during initial tests, produced predictions with the least RMSE variance, promising stable results. The overall algorithm is shown as Algorithm 1. Tests were repeated and a comparison of results for each separate model and newly proposed approach of adaptive machine-learning model selection is presented in Table 2. Each dataset number shown in the table corresponds to the numbering in Fig. 1. It can be observed that the new approach yields better results. Considerable gain is seen principally for data with sudden changes in values. Figure 3 shows that Random Forest underestimates values while the proposed solution quickly adjusts its predictions for dataset 6. Although tree-based models perform well in general, they cannot be utilized in every case. Where the model is selected imprudently, the results are much worse. Although the proposed approach performs equally well as just a single model in the case of the first dataset and a bit worse in the case of the second and third datasets, it produces much better results in other cases. The proposed solution self-adapts to unexpected instabilities and picks the model best suited to recent conditions. It is a very good trade-off, as it always works well and proves highly effective (up to 36% better prediction quality), especially in complicated and demanding testing scenarios.

3.4 Conclusions

Preliminary research yields promising results and demonstrates that data characteristic-based machine learning model selection might improve overall prediction quality. Therefore, based on the aforementioned results, the authors propose the Self-Adapting Data-Driven Prediction System (SA-DDPS). It will not only use adaptive ML algorithm selection from the preliminary research mechanism but is also based on the former work [24], which was tested in industry-grade environments and can significantly scale down cloud resource utilization along with reducing cloud usage costs.

4 Data-driven adaptive prediction

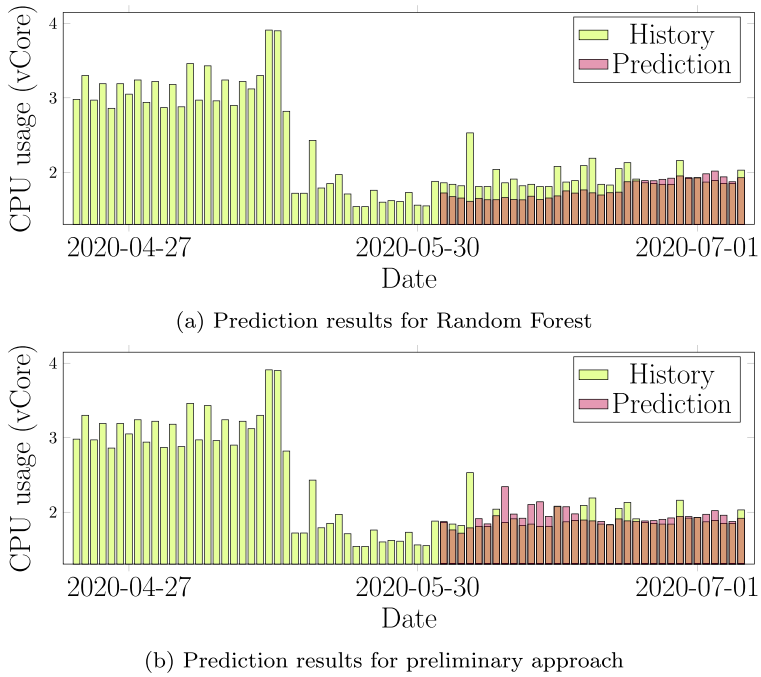
Predicting production system load on the basis of real-world data that contain anomalies and change

Table 2 Preliminary result comparison

Dataset	Model	RMSE	MAPE %
1	Naive	0.22	7.75
	Linear	0.03	1.15
	Forest	0.03	1.01
	XGBoost	0.03	1.10
	Proposed solution	0.03	1.01
2	Naive	1.99	4.89
	Linear	2.11	4.89
	Forest	2.09	4.82
	XGBoost	1.95	4.46
	Proposed solution	2.10	4.83
3	Naive	5.12	11.91
	Linear	5.98	13.62
	Forest	5.87	11.47
	XGBoost	5.50	11.25
	Proposed solution	5.86	11.44
4	Naive	3.18	15.00
	Linear	2.63	13.56
	Forest	3.67	15.91
	XGBoost	3.86	16.06
	Proposed solution	2.59	13.02
5	Naive	1.35	15.54
	Linear	1.92	20.49
	Forest	1.28	14.64
	XGBoost	1.41	17.20
	Proposed solution	1.22	12.97
6	Naive	0.21	7.23
	Linear	0.20	6.83
	Forest	0.25	9.19
	XGBoost	0.26	9.96
	Proposed solution	0.20	6.70

rapidly may pose significant difficulties. Therefore, it is crucial to adjust prediction techniques to potentially changing operating conditions. Furthermore, it is not feasible to manually configure prediction parameters, and thus a fully automated system is required. Based on these requirements, the authors developed the SA-DDPS – a data-driven self-adaptive prediction system which automatically creates the ML models' performance knowledge base, uses this knowledge base to select the optimal ML model and predicts system usage levels using this model. Subsequently, it uses the data predicted to generate a resource allocation plan and scales cloud resources according to this plan. This enables the allocation of required cloud resources only, which results in substantial resource usage optimization.

The system operation concept is presented in Fig. 4. Firstly, *Knowledge base creation* is performed – just once before any other operations. Next, *ML model selection*, *Prediction*, *Planning* and *Scaling* are performed sequentially in a loop, allowing for continuous cloud resource optimization. Loop length has to be related to the prediction horizon as the forecast is performed only once per loop. To calculate SA-DDPS efficiency, the authors defined the Δ metric that measures differences in prediction quality between the



(a) Prediction results for Random Forest

(b) Prediction results for preliminary approach

Fig. 3 Comparison of Random Forest and preliminary prediction approach

SA-DDPS system (N_{var}) and a statically selected prediction algorithm (N_{const}). Δ is defined as:

$$\Delta = 1 - \frac{N_{var}}{N_{const}} \tag{6}$$

and is expressed as a percentage.

The rest of the section is structured as follows: Section 4.1 describes knowledge base creation, the detailed ML model selection process is described in Sections 4.2, and 4.3 focuses on prediction, planning and scaling. The main focus of this work is on knowledge base creation and optimal ML model selection, and therefore details of prediction, planning

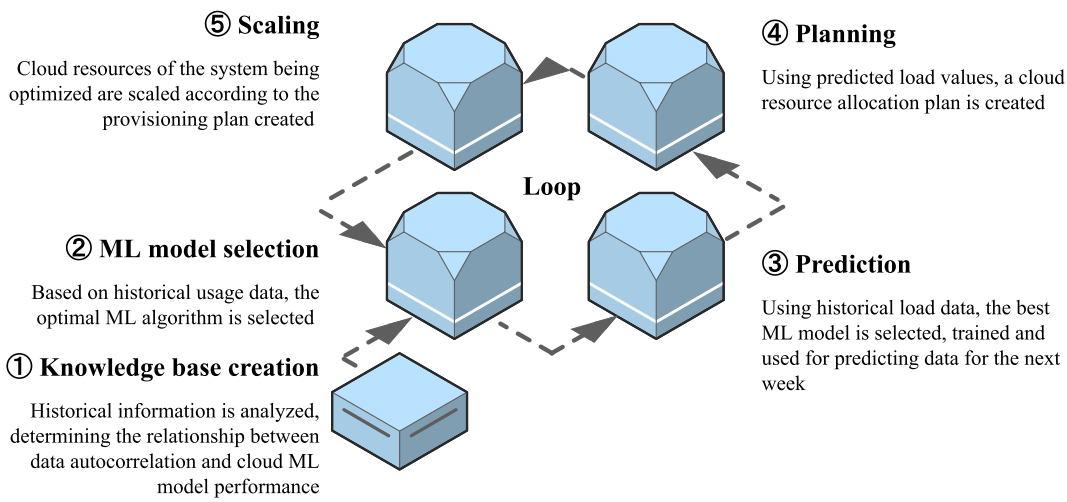


Fig. 4 SA-DDPS operation concept

and scaling described in [24] are omitted for clarity purposes.

4.1 Knowledge base creation

The ML optimal model selection discussed above requires ground-truth data. Production environments tend to differ in terms of load characteristics and usage patterns. Therefore, the SA-DDPS uses all available historical usage data and performs prediction using a set of different ML algorithms $O = (o_1, \ldots, o_x)$, and registers prediction parameters for each ML model used, i.e., the normalized root-mean-square error (NRMSE) defined as:

$$NRMSE = \frac{1}{\bar{y}} \cdot \sqrt{\frac{\sum_{t=1}^{T} \left(\hat{y}_t - y_t\right)^2}{T}} \tag{7}$$

the normalized mean absolute error (NMAE) defined as:

$$NMAE = \frac{1}{\bar{y}} \cdot \sqrt{\frac{\sum_{t=1}^{T} \left|\hat{y}_t - y_t\right|}{T}} \tag{8}$$

and the relative squared error (RSE) defined as:

$$RSE = \frac{\sum_{t=1}^{T} \left(\hat{y}_t - y_t\right)^2}{\sum_{t=1}^{T} \left(\bar{y} - y_t\right)^2} \tag{9}$$

where y_t is the actual resource usage level at time t, $\hat{y}_t$ is the predicted usage at time t and $\bar{y}$ is the arithmetic mean: $\bar{y} = \frac{1}{n} \cdot \sum_{t=1}^{T} y_t$ where n is the number of all samples in the set $S = (s_1, \ldots, s_n)$.

During preliminary work, the authors observed that a very significant usage data characteristic is periodicity, usually observed in hourly and daily periods. As a metric for this data property, the authors used the autocorrelation function (ACF) defined as:

$$ACF_l(S) = \frac{K_l(S)}{K_0(S)} \tag{10}$$

where l is defined as the lag between samples and $K_l(S)$ is defined as:

$$K_l(S) = \frac{1}{n} \cdot \sum_{t=1}^{T-l} (y_t - \bar{y}) \cdot (y_{t+l} - \bar{y}) \tag{11}$$

where n and $\bar{y}$ are defined above. It is worth noting that $K_l(S)$ is the autocovariance at lag l, which denotes the covariance of the data with itself at pairs of points separated by l-size gaps.

To calculate both hourly and daily periodicity, the authors defined the H and D metrics accordingly, which are calculated as the autocorrelation of autocorrelation maxima and are defined as follows:

$$\begin{aligned} H &= h \Leftrightarrow \forall i \in (2, \ldots, \alpha)\ ACF_h(\Gamma(ACF_h(S_H))) \\ &\geq ACF_i(\Gamma(ACF_i(S_H))) \end{aligned} \tag{12}$$

$$\begin{aligned} D &= d \Leftrightarrow \forall i \in (2, \ldots, \beta)\ ACF_d(\Gamma(ACF_d(S_D))) \\ &\geq ACF_i(\Gamma(ACF_i(S_D))) \end{aligned} \tag{13}$$

where α and β determine how many lag (l) values are considered, S_H and S_D are sets calculated from S with hourly and daily data resolution, respectively, and Γ is defined as:

$$\Gamma(x) = \begin{cases} 1, & x \text{ is a local maximum} \\ 0, & \text{otherwise} \end{cases} \tag{14}$$

Equations 12 and 13 make it possible to determine data periodicity by calculating the autocorrelation of data (inner ACF), extracting the local maxima of the autocorrelation calculated (Γ function), and calculating the autocorrelation of Γ function results (outer ACF). The process is repeated for defined values of lag $(2, \ldots, \alpha, 2, \ldots, \beta)$, and subsequently the lag with the highest ACF value is chosen as the predominant data periodicity.

The authors also defined the $ACF_{\max}$ metric – the maximum autocorrelation observed:

$$ACF_{\max}(S_H) = ACF_H(S_H) \tag{15}$$

$$ACF_{\max}(S_D) = ACF_D(S_D) \tag{16}$$

The knowledge base generated defines the dependencies between the set of ML models O, prediction quality (NRMSE, NMAE, RSE) and autocorrelation parameters ($ACF(S_H)$, $ACF(S_D)$). It is used as a base for the expert system utilized during the model selection stage.

4.2 ML model selection

The knowledge base defines dependencies between the data parameters, ML algorithms, and prediction quality. However, these dependencies are defined only for those values which were present in the historical data, and different values may be observed during the system optimization process. As a result, an expert system with fuzzy logic [18] was created to handle all possible data parameters. As the universe of discourse the authors used the maximum autocorrelation observed (ACF_{max}) and prediction quality metrics $N \in$ (NRMSE, NMAE, RSE). ML models O_i for $i \in (1, \ldots, q)$ were chosen as fuzzy sets

with membership functions $\mu_{O_i}(N, ACF_{\max})$. Membership function shapes were determined by the N metrics observed in historical data. NRMSE, NMAE and RSE values were scaled to $\langle 0, 1\rangle$ and inverted; as a result, 0 represents the largest values of N observed and 1 the lowest values, which are desirable, as lower N means better prediction quality. The continuity of $\mu_{O_i}(N, ACF_{max})$ was achieved by a linear interpolation of known values. Sample data were shown in Section 5 (Fig. 11).

The optimal ML algorithm $\hat{O}$ is calculated as follows:

$$\hat{O} = \Omega(ACF_{max}) \tag{17}$$

where Ω is the defuzzification process using first of maxima (FoM). The predicted optimal ML algorithm is subsequently used in the Prediction stage.

4.3 Prediction, Planning and Scaling

As mentioned above, some details of prediction, planning and scaling (described in [24]) have been omitted for clarity purposes, since knowledge base creation and the optimal ML model selection are the main scope of this paper.

In the next step of the optimization process, the ML algorithm selected ($\hat{O}$) is used to predict system load. The SA-DDPS is suitable for both short- and long-term predictions (from hours to months) as the timespan of historical data used to train the prediction model determines the model's ability to learn usage patterns. However, extensive training data timespans and long prediction horizons tend to degrade prediction quality if rapid load trend changes emerge. Therefore, both parameters have to be adjusted to the system being optimized.

The load predicted is used to determine the resources required, which allows the provisioning plan to be created. Cloud providers usually offer a set of machine types, which consists of machines with enhanced memory size, enhanced computing power or the golden mean between the two. As a result, the required resource level can be achieved using different configurations and at different costs. Therefore, for every predicted resource level, the SA-DDPS considers multiple possibilities and selects a cost-optimal choice.

The provisioning plans created are used to scale the optimized system resources. The SA-DDPS reads the current resource configuration from the cloud environment and compares it to the values from the provisioning plan. As many resources already provisioned as possible are reused, since provisioning and decommissioning take time. Additionally, the SA-DDPS performs warm-up for newly created resources to ensure their full availability.

5 SA-DDPS Evaluation

Presented in Section 4, the concept of the SA-DDPS was used to develop a real-life working application. The authors chose Azure as the environment to run the implemented system in; Azure is also used by the real-life application that was optimized during the evaluation: an Internet of Things hub (TMS) for devices scattered across the world. The TMS utilizes different cloud resources: virtual machines, app services and databases, thus creating a valuable testing environment as usage patterns for these resources differ significantly. The SA-DDPS was implemented using the C# language and Microsoft Azure Machine Learning Studio (MAMLS). The rest of the section is structured as follows: Section 5.1 describes the evaluation of the knowledge base creation process, testing the dynamic ML model selection process is described in Sections 5.2, and 5.3 focuses on comparing the results obtained by the SA-DDPS to those obtained by the static selection of ML algorithms.

5.1 Knowledge base creation

As mentioned before, the TMS historical load consists of data with different characteristics. Therefore, to build the knowledge base, the network incoming data level and the IOPS level (virtual machines), incoming requests (app service), and database transaction units – DTU[2] (database) were selected. For the sake of clarity, detailed calculation results are presented only for app service incoming requests. Historical levels with hourly resolution are presented in Fig. 5, and autocorrelation values calculated according to (10) along with local maxima are presented in Fig. 6.

The autocorrelation of autocorrelation maxima ($ACF(\Gamma(ACF(S_H)))$) is presented in Fig. 7; the

[2]Database transaction units – https://docs.microsoft.com/en-us/azure/azure-sql/database/service-tiers-dtu?view=azuresql

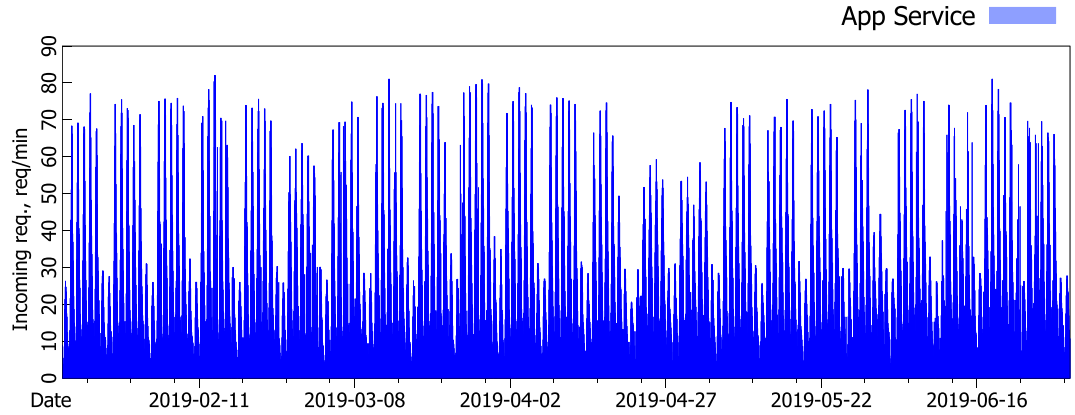


Fig. 5 Historical data from app service – hourly resolution

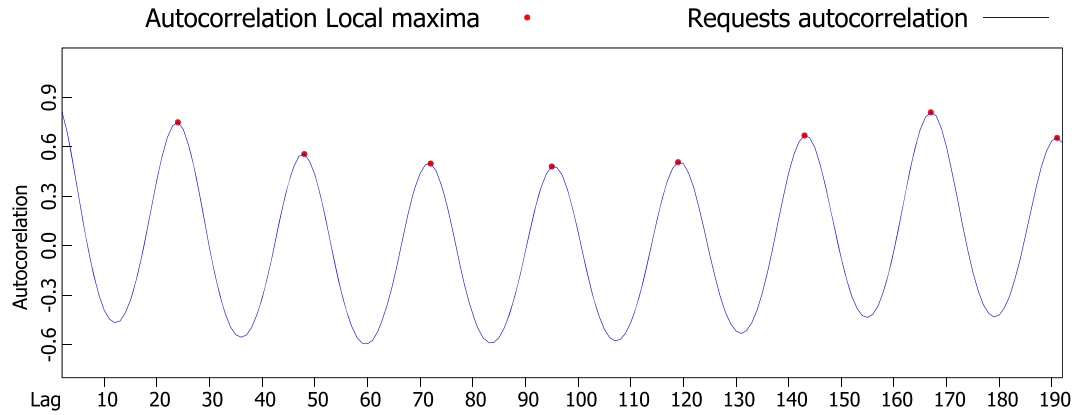


Fig. 6 App service autocorrelation with local maxima – hourly resolution

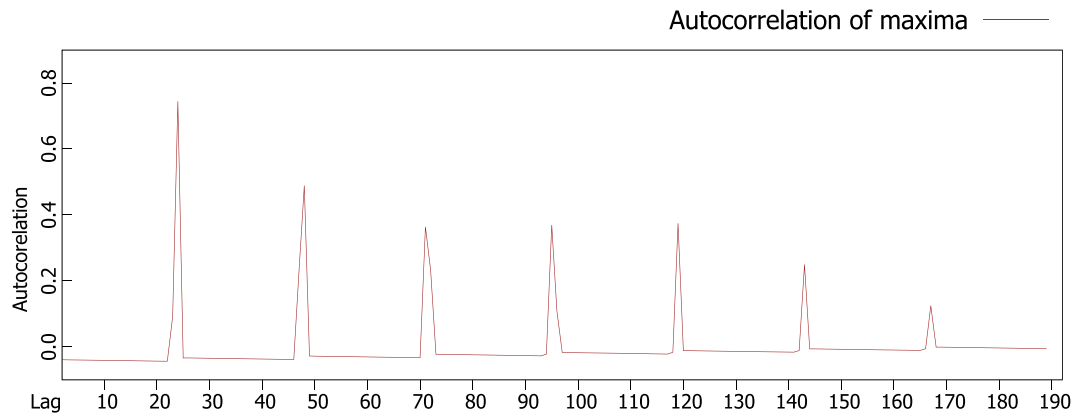


Fig. 7 App service autocorrelation of autocorrelation maxima – hourly resolution

maximum value for lag $l = 24$ indicates a strong daily (24-hour) usage pattern. Similarly, Figs. 8, 9 and 10 present calculation steps for historical levels with daily resolution. The maximum value for lag $l = 7$ implies a weekly (7 days) usage pattern.

The characteristics of all historical data are presented in Table 3. Hourly and daily periods calculated exhibit daily and weekly dependencies in many cases, especially for high ACF_{max} values which indicate high data repeatability. Database DTU data with daily resolution did not exhibit any periodicity, and the periods calculated for data with ACF_{max} lower than 0.3 indicate very weak recurrence.

5.2 ML model selection

As mentioned in Section 4.2, in order to select the optimal ML algorithm, the authors used an expert system with fuzzy logic. As the universe of discourse, the authors selected prediction quality metrics $N \in$ (NRMSE, NMAE, RSE), and, based on the results from Table 3, $ACF_{max}(S_H) \in \langle 0.15, 0.74 \rangle$. For simplification purposes, $ACF_{max}(S_D)$ was not included. As the ML model set O, the authors picked seven commonly used algorithms: Decision Forest Regression (DF), Boosted Decision Tree Regression (BDT), Fast Forest Quantile Regression (FFQ), Bayesian Linear Regression (BLR), Linear Regression (LR), Neural Network Regression (NN) and Poisson Regression (PR). All of them were used in two modes to create fourteen models in total. In the first mode, the algorithms were trained using historical data enhanced with additional features – hours and days encoded as one-hot according to the hourly and daily periods calculated (Table 3). In the second mode, models were trained using historical univariate data with a ten-day long sliding window (Fig. 2). To distinguish between these two modes, the univariate models were named DF*, BDT*, FFQ*, BLR*, LR*, NN* and PR*, respectively. All ML models were trained using self-tuning Tune Model Hyper-Parameters with initial values set to MAMLS defaults.

Values of membership functions (μ_{O_i}), calculated in accordance with the description in Section 4.2, are presented in Fig. 11a for NRMSE, Fig. 11b for NMAE and Fig. 11c for RSE.

To evaluate SA-DDPS efficiency, the authors selected four different data types: App Service (incoming requests), Virtual machine (IOPS), Virtual machine (incoming data) and Database (DTU) with dynamically changing data characteristics. For the periods selected, the authors performed two cycles of ML model selection, Prediction, Planning and Scaling (Fig. 4). In every cycle, for all four data types, autocorrelation and period values were calculated, and optimal ML algorithms were selected using (17). Sample μ_{O_i} calculation results for Virtual machine (IOPS) in the 2nd cycle are presented in Fig. 12.

The complete results for both cycles and all four data types are presented in Table 4. In every case, the ACF_{max} values calculated for the 1st and 2nd steps were sufficiently different to trigger prediction algorithm adjustment. It is worth noting that calculated periodicity was used only for App Service (incoming requests) in the 1st step, as in the other cases the SA-DDPS selected univariate modes of ML algorithms.

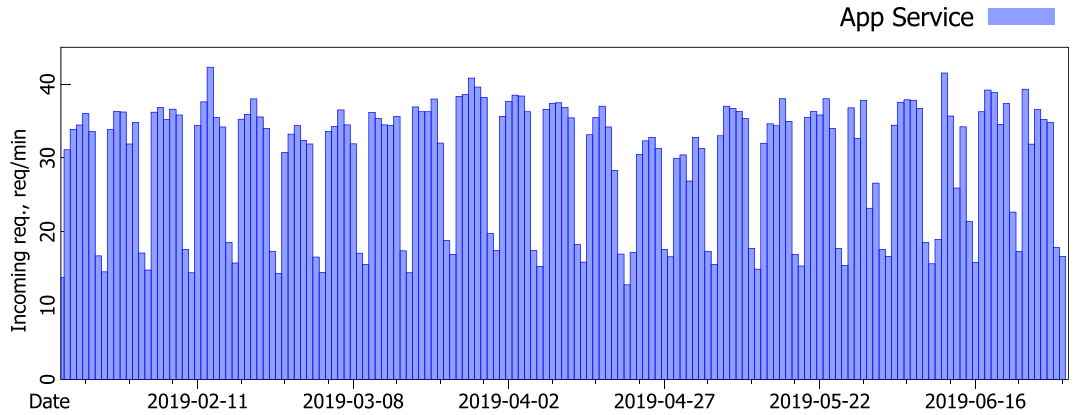


Fig. 8 Historical data from app service – daily resolution

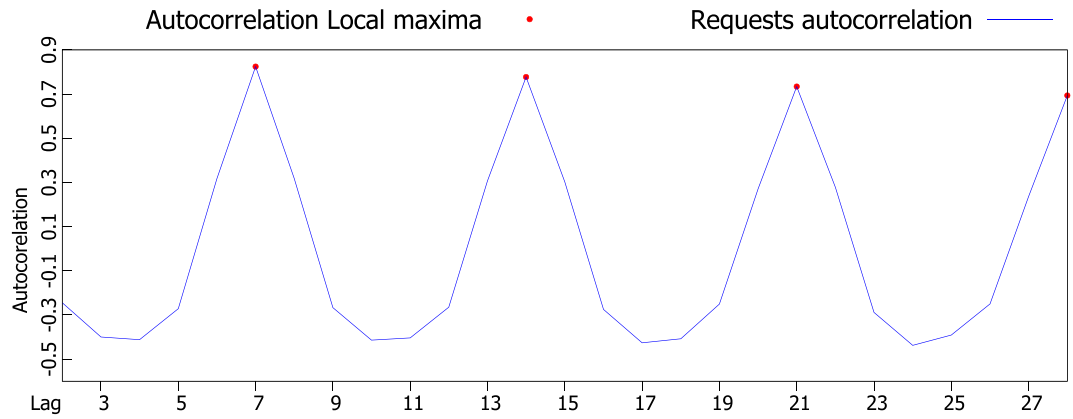


Fig. 9 App service autocorrelation with local maxima – daily resolution

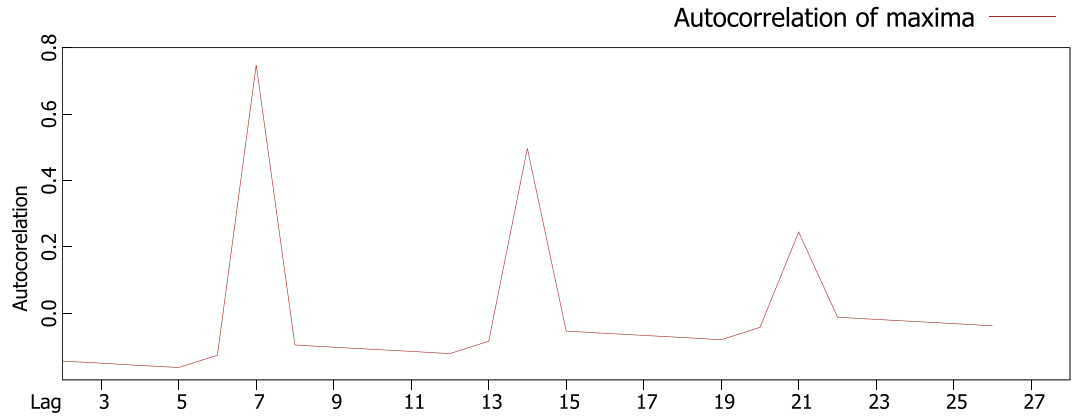


Fig. 10 App service autocorrelation of autocorrelation maxima – daily resolution

Table 3 Characteristics of all data used in the knowledge base creation

Service type	Load type	ACF_{max}		Period	
		Hourly	Daily	Hourly	Daily
App Service	Incoming requests	0.74	0.75	24	7
		0.48	0.45	24	7
Virtual machine	Incoming data	0.42	0.42	24	4
		0.15	0.27	131	15
	IOPS	0.71	0.64	24	7
		0.37	0.02	72	15
Database	DTU	0.27	–	24	–

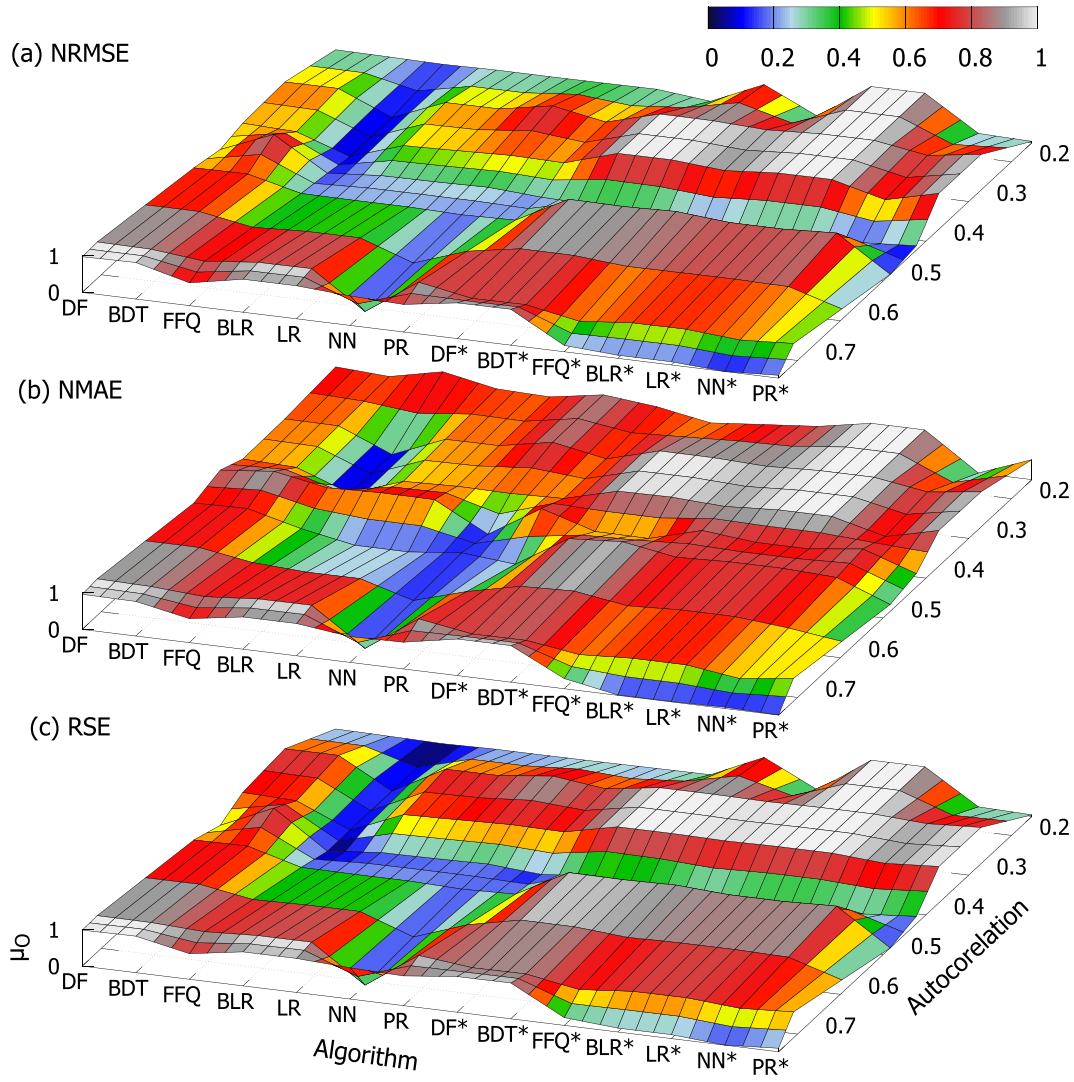


Fig. 11 μ_{O_i} values calculated for NRMSE, NMAE and RSE

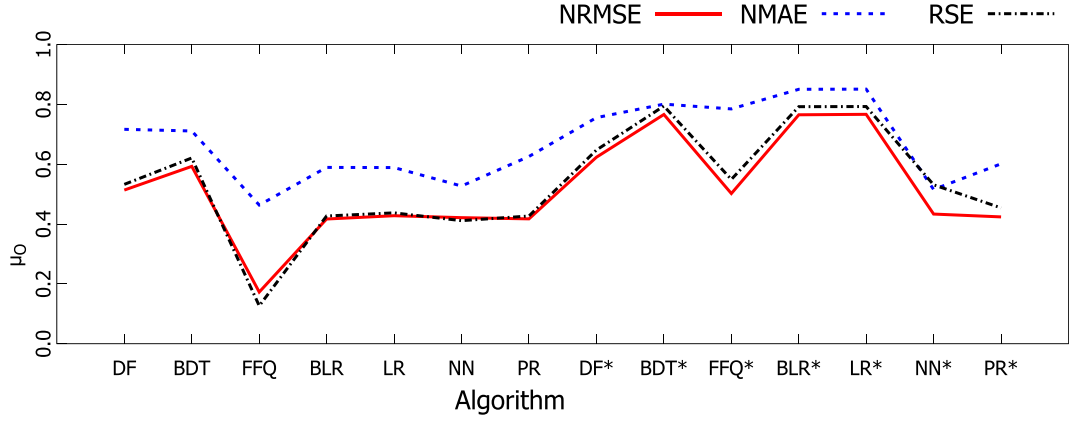


Fig. 12 μ_{O_i} calculation results for Virtual machine (IOPS) in the 2nd evaluation cycle

Table 4 ACF_{max}, periodicity and model selection results for both cycles

	ACF_{max}		Period		Selected model
	Hourly	Daily	Hourly	Daily	
App Service (incoming requests)					
1st step	0.75	0.38	24	7	BDT
2nd step	0.43	0.30	72	7	BDT*
Virtual machine (IOPS)					
1st step	0.49	–	24	–	BDT*
2nd step	0.36	–	49	–	LR*
Virtual machine (incoming data)					
1st step	0.28	0.22	78	18	LR*
2nd step	0.41	0.28	24	4	BDT*
Database (DTU)					
1st step	0.38	0.27	23	15	BDT*
2nd step	0.15	0.27	11	15	LR*

5.3 Comparison of prediction quality

The authors calculated Δ metrics (6) using different algorithms as statically selected ML models (Table 5). The proposed solution performed better than any other algorithm evaluated: a prediction quality improvement ranging from 9.28% to 80.68% was observed. ML models in the univariate mode are more resilient to data characteristic changes, and therefore the improvement in quality was smaller in these cases. Nevertheless, the proposed solution significantly improved prediction quality, which is crucial for accurate resource usage prediction.

Table 5 Δ calculated against different ML algorithms

Algorithm	Δ metric		
	NMAE	NRMSE	RSE
DF	44.08%	46.08%	61.20%
BDT	47.50%	43.74%	57.98%
FFQ	61.84%	56.29%	76.33%
BLR	68.36%	59.05%	80.68%
LR	65.98%	57.38%	77.95%
NN	62.35%	54.50%	71.01%
PR	62.75%	54.49%	75.11%
DF*	27.16%	22.99%	35.49%
BDT*	16.96%	16.18%	21.71%
FFQ*	23.69%	23.62%	32.90%
BLR*	13.78%	9.28%	25.03%
LR*	18.11%	16.47%	29.12%
NN*	52.69%	43.54%	60.91%
PR*	25.61%	15.26%	29.28%

Additionally, the authors compared the NMAE, NRMSE and RSE values (separately for every evaluation data type) between the SA-DDPS and the best suited (for the given data type) ML algorithms statically selected at the beginning of the first evaluation cycle. In Fig. 13, evaluation results are presented in the form of a histogram. Due to the magnitude of the metric values observed, the authors decided to use a logarithmic scale; it can be observed that the SA-DDPS outperforms statically selected algorithms. Prediction quality improvements observed for every test case indicate not only that the forecast algorithm ought to be fitted to data characteristics, but also that the adjustment process ought to be performed continuously to account for changes in data parameters.

6 Conclusions

In this paper, the authors firstly present preliminary research evaluating a novel approach, and next, a data-driven solution for prediction algorithm adaptation. Dynamic algorithm selection combined with data pre-processing allows the SA-DDPS to formulate precise short- and long-term predictions. The experiments

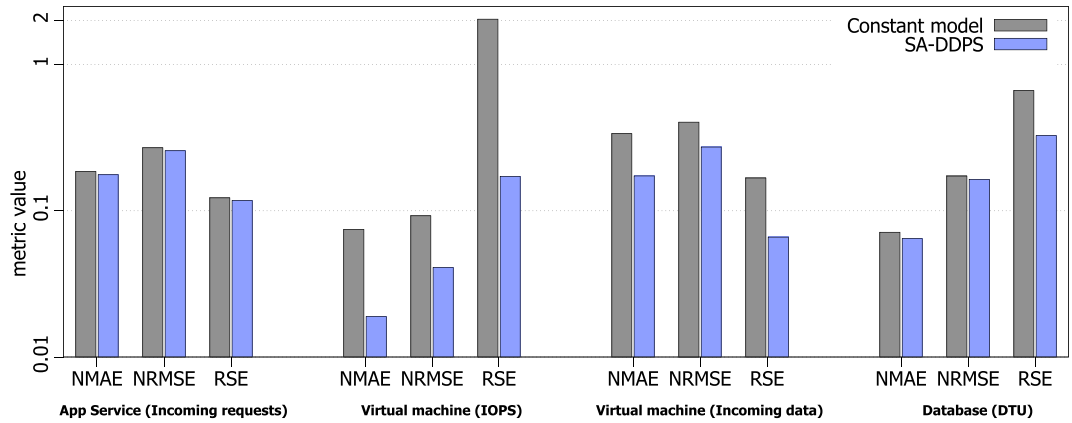


Fig. 13 Δ calculated against best-suited ML models selected at the beginning of the first evaluation cycle

conducted demonstrate that both machine learning model selection and data types significantly affect the results obtained and therefore they should be properly matched. The evaluation performed against industry-grade test data confirmed that the proposed solution performs better than any single prediction algorithm based on machine learning. This is especially evident in test cases that contain rapid pattern changes; in these cases, thanks to dynamic algorithm selection. A 9.28% to 80.68% improvement in predictions was observed, and there was no single model with better predictions for every test scenario. Therefore, it would not be possible to obtain better results in a real-life situation unless the prediction algorithm is manually selected whenever data characteristics change. Thus, the proposed solution not only yields better predictions for dynamic data with variable patterns but, in general, also performs better for data with less fluctuations.

As a subject of future studies, the authors would like to consider transferring the knowledge base created between different systems, which will make the above step unnecessary, or investigate federated learning (collaborative learning) to incrementally create such a knowledge base. Alternatively, the authors would like to investigate merging different knowledge bases to improve precision. Another way to improve the SA-DDPS may involve additional algorithms for investigating data characteristics, as more information about data should result in better model selection.

Acknowledgements The research presented in this paper was supported by funds from the Polish Ministry of Education and Science allocated to the AGH University of Science and Technology. The authors would like to thank Polcom for providing the data used in the tests.

Author Contributions Piotr Nawrocki – developed the concept of the article, designed the structure of the article, analyzed the available literature (Section 2), developed the introduction and summary, checked the entire article. Patryk Osypanka – developed Sections 4, 5, and summary. Beata Posluszny – developed Section 3 and partly Section 2.

Funding The research presented in this paper was supported by funds from the Polish Ministry of Education and Science allocated to the AGH University of Science and Technology.

Data Availability The data that support the findings of this study are available from Polcom, but restrictions apply to the availability of these data, which were used under license for the current study, and thus are not publicly available. The data are, however, available from the authors upon reasonable request and with permission of Polcom.

Compliance with Ethical Standards

Conflict of Interests The authors declare that they have no conflict of interest.

References

1. Abrol, P., Gupta, S., Singh, S.: Qos aware social spider cloud web algorithm: Analysis of resource placement approach. In: Proceedings of International Conference on Advancements in Computing & Management (ICACM) (2019)
2. Adhikari, M., Amgoth, T.: Multi-objective accelerated particle swarm optimization technique for scientific workflows in IaaS cloud. In: 2018 International Conference on Advances in Computing, Communications and Informatics (ICACCI), pp. 1448–1454. IEEE (2018)
3. An, N.H., Anh, D.T.: Comparison of strategies for multi-step-ahead prediction of time series using neural network. In: 2015 International Conference on Advanced Computing and Applications (ACOMP), pp. 142–149 (2015). https://doi.org/10.1109/ACOMP.2015.24
4. Andrae, A.S., Edler, T.: On global electricity usage of communication technology: Trends to 2030. Challenges **6**(1), 117–157 (2015)
5. Biswal, T.: Random forest for time series forecasting. Data Sci. Blogathon - 8 (2021)
6. Chen, X., Wang, H., Ma, Y., Zheng, X., Guo, L.: Self-adaptive resource allocation for cloud-based software services based on iterative QoS prediction model. Futur. Gener. Comput. Syst. **105**, 287–296 (2020)
7. Chen, Y., Huang, J., Lin, C., Shen, X.: Multi-objective service composition with QoS dependencies. IEEE Trans. Cloud Comput. **7**(2), 537–552 (2016)
8. Crecana, C.C., Pop, F.: Monitoring-based auto-scalability across hybrid clouds. In: Proceedings of the 33rd Annual ACM Symposium on Applied Computing, pp. 1087–1094 (2018)
9. Dietterich, T.G.: Machine learning for sequential data: A review. In: Caelli, T., Amin, A., Duin, R.P.W., de Ridder, D., Kamel, M. (eds.) Structural, Syntactic, and Statistical Pattern Recognition, pp. 15–30. Springer Berlin Heidelberg, Berlin (2002)
10. Duggan, M., Mason, K., Duggan, J., Howley, E., Barrett, E.: Predicting host CPU utilization in cloud computing using recurrent neural networks. In: 2017 12th International Conference for Internet Technology and Secured Transactions (ICITST), pp. 67–72. IEEE (2017)
11. Hansun, S.: A new approach of moving average method in time series analysis. In: 2013 Conference on New Media Studies (CoNMedia), pp. 1–4 (2013). https://doi.org/10.1109/CoNMedia.2013.6708545
12. Hilman, M.H., Rodriguez, M.A., Buyya, R.: Task runtime prediction in scientific workflows using an online incremental learning approach. In: 2018 IEEE/ACM 11th International Conference on Utility and Cloud Computing (UCC), pp. 93–102. IEEE (2018)
13. Hosseinzadeh, M., Ghafour, M.Y., Hama, H.K., Vo, B., Khoshnevis, A.: Multi-objective task and workflow scheduling approaches in cloud computing: a comprehensive review. J. Grid Comput, 1–30 (2020)
14. Jain, D., Goutam, A.: Optimization of resource and task scheduling in cloud using random forest. In: 2017 International Conference on Advances in Computing, Communication and Control (ICAC3), pp. 1–5 (2017). https://doi.org/10.1109/ICAC3.2017.8318757
15. Jiang, W., Lee, D., Hu, S.: Large-scale longitudinal analysis of soap-based and restful web services. In: 2012 IEEE 19th International Conference on Web Services, pp. 218–225 (2012)
16. Kane, M., Price, N., Scotch, M., Rabinowitz, P.: Comparison of ARIMA and random forest time series models for prediction of avian influenza H5N1 outbreaks. BMC Bioinform. **15**, 276 (2014). https://doi.org/10.1186/1471-2105-15-276
17. Mani, S.K., Meenakshisundaram, I.: Improving quality-of-service in fog computing through efficient resource allocation. Computational Intelligence (2020)
18. Medsker, L.R.: Fuzzy Logic and Expert Systems, pp. 95–105. Springer US, Boston (1995). https://doi.org/10.1007/978-1-4615-2353-6_6
19. Mireslami, S., Rakai, L., Far, B.H., Wang, M.: Simultaneous cost and QoS optimization for cloud resource allocation. IEEE Trans. Netw. Serv. Manag. **14**(3), 676–689 (2017)
20. Nääs Starberg, F., Rooth, A.: Predicting a business application's cloud server CPU utilization using the machine learning model LSTM (2021)
21. Nawrocki, P., Grzywacz, M., Sniezynski, B.: Adaptive resource planning for cloud-based services using machine learning. J. Parallel Distrib. Comput. **152**, 88–97 (2021). https://doi.org/10.1016/j.jpdc.2021.02.018
22. Nawrocki, P., Osypanka, P.: Cloud resource demand prediction using machine learning in the context of QoS parameters. J. Grid Comput. **19**(2), 20 (2021). https://doi.org/10.1007/s10723-021-09561-3
23. Osypanka, P., Nawrocki, P.: Resource usage cost optimization in cloud computing using machine learning. IEEE Trans. Cloud Comput., 1–1 (2020)
24. Osypanka, P., Nawrocki, P.: QoS-aware cloud resource prediction for computing services. IEEE Transactions on Services Computing (2022)
25. Rahman, S., Ahmed, T., Huynh, M., Tornatore, M., Mukherjee, B.: Auto-scaling VNFs using machine learning to improve Qos and reduce cost. In: 2018 IEEE International Conference on Communications (ICC), pp. 1–6 (2018)
26. Ranjbari, M., Torkestani, J.A.: A learning automata-based algorithm for energy and SLA efficient consolidation of virtual machines in cloud data centers. J. Parallel Distrib. Comput. **113**, 55–62 (2018)
27. Rao, S.N., Shobha, G., Prabhu, S., Deepamala, N.: Time series forecasting methods suitable for prediction of CPU usage. In: 2019 4th International Conference on Com-

putational Systems and Information Technology for Sustainable Solution (CSITSS), vol. 4, pp. 1–5 (2019). https://doi.org/10.1109/CSITSS47250.2019.9031015
28. Sung, H., Min, J., Koo, D., Eom, H.: OMBM-ML: efficient memory bandwidth management for ensuring QoS and improving server utilization. Clust. Comput. **24**(1), 181–193 (2021)
29. Syu, Y., Kuo, J.Y., Fanjiang, Y.Y.: Time series forecasting for dynamic quality of web services: an empirical study. J. Syst. Softw. **134**, 279–303 (2017)
30. Wang, S., Yao, Y., Xiao, Y., Chen, H.: Dynamic resource prediction in cloud computing for complex system simulatiuon: A probabilistic approach using stacking ensemble learning. In: 2020 International Conference on Intelligent Computing and Human-Computer Interaction (ICHCI), pp. 198–201 (2020). https://doi.org/10.1109/ICHCI51889.2020.00050
31. Ye, Z., Mistry, S., Bouguettaya, A., Dong, H.: Long-term QoS-aware cloud service composition using multivariate time series analysis. IEEE Trans. Serv. Comput. **9**(3), 382–393 (2014)
32. Zhang, Q., Yang, L.T., Yan, Z., Chen, Z., Li, P.: An efficient deep learning model to predict cloud workload for industry informatics. IEEE Trans. Ind. Inform. **14**(7), 3170–3178 (2018)
33. Śnieżyński, B., Nawrocki, P., Wilk, M., Jarzab, M., Zielinski, K.: VM reservation plan adaptation using machine learning in cloud computing. J. Grid Comput. **17**. https://doi.org/10.1007/s10723-019-09487-x (2019)