Journal of Computational Science 79 (2024) 102292

Contents lists available at ScienceDirect

Journal of Computational Science

journal homepage: www.elsevier.com/locate/jocs

FinOps-driven optimization of cloud resource usage for high-performance computing using machine learning

Piotr Nawrocki *, Mateusz Smendowski

AGH University of Krakow, Faculty of Computer Science, al. A. Mickiewicza 30, 30-059 Krakow, Poland

ARTICLE INFO

Keywords:
Cloud computing
High-performance computing
Long-term resource prediction
Resource reservation
Machine learning
Time series forecasting
FinOps

ABSTRACT

Cloud computing is gaining popularity in high-performance computing applications. Its utilization enables advanced simulations when local computing resources are limited. However, cloud usage may increase costs and entail resource unavailability risks. This article presents an original approach that employs machine learning to predict long-term cloud resource usage. This enables optimizing resource utilization through appropriate reservation plans, reducing the associated costs. The solution developed utilizes statistical models, XGBoost, neural networks and the Temporal Fusion Transformer. Long-term prediction of cloud resource consumption, especially the *Cloud Resource Usage Optimization System* that is critical for prolonged simulations, involves using prediction results to dynamically create resource reservation plans across various virtual machine types for HPC on the Google Cloud Platform. Experiments using real-life production data demonstrate that the TFT prediction model improved prediction quality (by 31.4%) compared to the best baseline method, particularly in adapting to chaotic changes in resource consumption. However, it should be noted that the best prediction model in terms of error magnitude might not be the most suitable for resource reservation planning. This was validated by the neural network-based method, introducing an FR metric for forecast evaluation. Resource reservation plans were assessed both qualitatively and quantitatively, focusing on various aspects like a service-level agreement compliance and potential downtime. This paper is an extension of work originally presented during the International Conference on Computational Science — ICCS 2023, entitled "Long-Term Prediction of Cloud Resource Usage in High-Performance Computing".

1. Introduction

Cloud computing is increasingly becoming an alternative to supercomputers for some high-performance computing (HPC) applications [1,2]. Potential use areas of cloud computing include various advanced simulations such as, for instance, HPC simulations of disease spread, flow [3] or social simulations [4]. Hybrid environments are also being used, where part of the sensitive computation is performed locally and part in a public cloud [5]. Public clouds can be used where the demand for computing resources is greater and cannot be satisfied by local resources. Of course, the cost of using cloud resources in HPC has to be taken into account, which is why the issue of optimizing their use is so important. In this context, it is very important to be able to predict the consumption of cloud resources in order to reserve them optimally. Reserving too many resources may result in increased costs and reserving too few resources may cause problems with simulation execution. Although cloud resource usage prediction (and subsequent scheduling) is discussed in the literature, it is almost always in the context of autoscaling and short-term prediction. For HPC applications such as advanced simulations, many tasks may require resources in the long term, and therefore the time horizon of mechanisms such as autoscaling and short-term prediction may prove too short for this type of computing. The long-term prediction solution developed by the authors (using machine learning) allows for a longer prediction horizon, which can be useful for advanced simulations using HPC. Despite the flexibility that cloud computing offers in terms of resource utilization or services, attention must be paid to a range of associated risks. Firstly, utilizing resources in an on-demand model may pose a risk of their unavailability, thus the possibility of reserving resources in advance becomes increasingly relevant. In connection with FinOps (*Financial Operations*) aspects [6], it is crucial to implement a suitable culture and approach that aligns with prudent financial resource management, supporting decision-making and operational efficiency. In this scenario, forecasting can successfully serve as a means to understand future consumption and plan costs accordingly. These forecasts can be translated into resource reservation plans, which, however, need to be critically evaluated in terms of quality and quantity. As forecasting time

* Corresponding author.
E-mail addresses: piotr.nawrocki@agh.edu.pl (P. Nawrocki), smendowski@agh.edu.pl (M. Smendowski).

https://doi.org/10.1016/j.jocs.2024.102292
Received 7 December 2023; Received in revised form 1 March 2024; Accepted 9 April 2024
Available online 27 April 2024

series in dynamic environments like cloud computing involves considerable uncertainty, ensuring minimal service-level agreement (SLA) breaches and reducing downtime is crucial. Consequently, a natural tradeoff arises between overprovisioning and the availability of services and resources for HPC. This paper is an extension of work originally presented during the ICCS 2023 conference [7]. In particular, Section 3 expands the system to encompass a multi-virtual machine scenario and introduces a new *Reservation Module*. Furthermore, Section 4 presents additional results of previous prediction methods and introduces new ones (especially based on neural networks and the Temporal Fusion Transformer), subsequently translating them into dynamic resource reservation plans, including their comprehensive assessment.

The major contributions of this paper can be summarized as follows:

- novel approach that employs machine learning to provide long-term resource usage predictions for advanced simulations using HPC;
- a *Reservation Module* which serves as a central point in the FinOps-driven approach;
- the evaluation of diverse time series forecasting methods including neural networks and the Temporal Fusion Transformer, using a real-life dataset from a production system;
- qualitative and quantitative assessment of resource reservation plans based on prediction results for selected reference types of HPC machines.

The rest of this paper is structured as follows: Section 2 contains a description of related work, Section 3 focuses on the description of a *Cloud Resource Usage Optimization System*, Section 4 describes the experiments performed, including the evaluation of prediction results and subsequent cost-effectiveness estimation related to resource reservation plans, and Section 5 contains the conclusion and outlines further work.

2. Related work

There are many publications dealing with the use of cloud computing resources for HPC applications. In [8], the author discusses the use of a virtual cluster for this type of computing (using Elastic Computing Cloud — EC2 as an example). Clusters of computer systems are a common HPC architecture. However, small local clusters, although cost-effective, may not be efficient enough. On the other hand, large clusters are more expensive, and it is not always possible to provide them with a sufficient sustainable load. Therefore, virtual clusters using cloud resources offer a viable alternative. In this case, public commercial cloud environments provide users with storage and CPU (Central Processing Unit) power to build their own dedicated clusters of computers that can be used in scientific computing applications. Based on the experiments performed, the author shows that it is possible to use a virtual cluster to realize various computations including HPC. The author also analyzes the cost of using cloud solutions; however, he does not explore the possibility of optimization of cloud resource consumption.

In [9], the authors discuss scalability, interoperability, and achieving guaranteed Quality of Service (QoS) in a High-Performance Computing Cloud (HPCC) environment. The authors propose a cloud resource management framework to handle a large number of user HPC application requests and manage multiple cloud resources. System tests were performed using a large number of real-world HPC applications. To evaluate the performance of the system proposed, the authors used performance metrics such as response time, the number of requests successfully handled and user satisfaction. What is missing from this work, however, is an analysis of the cost of using cloud resources using the system developed and the possibility of optimizing them.

In [10], the authors present the possibilities of using HPC in the Google Cloud Platform in emergency situations when efficient processing of large amounts of traffic data is required. This allows for effective disaster management using massive data processing and high-performance computing. Another application of HPC in cloud computing is presented in [11] where the authors describe a platform for computer vision applications that enables audio/video processing and can use high performance computing cloud resources for this purpose. However, the aspect related to maximizing efficient resource management for HPC is missing, which, due to its considerable importance, translates into high operational costs.

An extensive analysis of the possibilities of cloud solutions related to High Performance Computing, among other things, is presented in [12]. The authors analyze the capabilities of the four most popular environments — Amazon Elastic Compute Cloud, Microsoft Azure Cloud, Google Cloud and Oracle Cloud. Today, HPC workloads are increasingly being migrated to the cloud. This is due, among other things, to the flexible nature of the cloud where resources can be expanded and reduced on demand, which optimizes the cost of using cloud computing. At the same time, computationally intensive workloads can be performed efficiently on cloud resources that are connected via a high-speed network. The most suitable cloud model for HPC users is Infrastructure as a Service (IaaS), where it is possible to specify all the details of the infrastructure needed to create clusters. There are many areas where the cloud is being used for HPC, including financial risk simulations, molecular dynamics, weather prediction, and scientific or engineering simulations. Despite the fact that the cloud computing model is flexible and appealing, there is no consideration of the risks associated with its application. Within the *CRUOS*, resource forecasting for the purpose of reserving resources was incorporated, giving assurance that the resources reserved will be available at the specified time, what is aligned with FinOps principles.

An important aspect of the use of cloud resources is the optimization of their usage, including accounting for cost aspects. In [13], the authors analyze the problem of cost-effectiveness of the use of cloud resources, which directly affects the competitiveness of companies using cloud solutions. The article presents the SmartCMP platform, which provides cost analysis of cloud usage and enables real-time monitoring, detection and remediation of threats, making it possible to optimize cloud resource usage. The authors also present an interesting FinOps concept [6] related to cloud cost optimization. This concept integrates DevOps, finance and business, and aims to optimize costs and technical solutions in the context of cloud usage. The major cloud service providers in their environments (such as Amazon AWS [14] or Microsoft Azure [15]) are implementing mechanisms to optimize the cost of using cloud resources according to the FinOps concept. It can be seen that this concept is becoming increasingly important for cloud resource providers and users [16]. At the same time, however, there is a lack of research showing the tangible benefits of applying the FinOps concept to real-life cloud environments. This article is one of the first to present research results on the use of the FinOps concept to optimize cloud resource usage for high performance computing in particular. Providing the research context in relation to financial gains is pivotal as it enables more informed decision-making, essentially unlocking the complete potential of HPC in the cloud while managing resources responsibly.

As it can be gathered from the above analysis, the use of cloud resources for HPC is now commonplace. There are also appropriate mechanisms for managing cloud resources so as to adjust them for HPC purposes and, for instance, predict the placement of containers [17]. However, there is a lack of mechanisms to optimally reserve cloud resources for HPC, especially in the long term and accounting for FinOps concepts. One way to conduct such optimization is to use prediction mechanisms to determine what resources will be required in the future for HPC and establish an optimal reservation plan. The most common strategy for using cloud computing resources is to reserve resources at the highest potential demand level, which, however, generates unnecessary costs for unused cloud resources. Predictive resource utilization can prevent this, helping to reserve only the cloud

resources needed. However, the majority of research on cloud resource management and optimization concerns autoscaling [18,19], predictive autoscaling (autoscaling with workload forecasting) [20,21] and short-term prediction [22,23]; there are very few studies of long-term prediction.

The authors' earlier works indicate that cloud resource usage prediction makes it possible to create optimal plans for resource reservation, leading to significant savings. Those works included research on using machine learning and adaptive resource planning for cloud-based services [24], anomaly detection in the context of such planning [25], cloud resource usage prediction mechanisms taking into account QoS parameters [26,27], data-driven adaptive cloud resource usage prediction [28] and resource usage cost optimization in cloud computing [29]. However, none of the previous works analyzed HPC needs in the context of cloud computing resources. In the research described in this article, the authors paid particular attention to the need for long-term prediction (relevant to HPC) and used a full data set from more than one year to develop the model. In previous work (such as in [27,29]), only selected data from three months were considered. In addition to considering the full scope of data, which also encompasses periods of unstable virtual machine operation, the decision was made to evaluate neural networks and the transformer model with a self-attention mechanism. Furthermore, the resource reservation plans presented were contextualized within HPC, with both qualitative and quantitative evaluation. Adopting an approach aligned with FinOps principles not only made it possible to conduct a complementary assessment of reservation plans but also their qualitative and quantitative evaluation within the HPC context. Additionally, not only anomaly detection was considered but also robust feature engineering, which is crucial in the time series forecasting area.

3. Cloud resource usage optimization system

The *Cloud Resource Usage Optimization System*, also referred to as the *CRUOS*, has been designed with the primary objective of predicting resource usage in the long term. The resulting predictions serve as the foundation for resource reservation planning, contributing to the optimization of resource utilization. Such a system aims to ensure that resources are allocated and reserved in accordance with actual needs, thereby preventing resource wastage. The system comprises eight interconnected modules, as depicted in Fig. 1. The architecture is split into two distinct segments. The first one encompasses all modules except the *Reservation Module*. In scenarios involving the forecasting of multiple resources for numerous independent virtual machines, the long-term forecasting task is denoted as multi-series forecasting. Consequently, modules 1 through 7 concentrate on individual time series, thus they are dedicated on a per-resource, per-virtual machine basis. On the other hand, the second segment, which includes the *Reservation Module*, is shared among all virtual machines since its purpose is to handle requests for resource reservation plan preparations.

Given that the system's task is to forecast resource consumption, such as CPU and RAM (Random Access Memory), for multiple virtual machines, and virtual machines are denoted by unique identifiers from *1* to *N*, communication between *CRUOS* segments can be visualized as depicted in Fig. 2. It is important to note that the entire first segment of the system is dedicated to a single resource per machine. Simultaneously, the system concept maintains complete flexibility, allowing its forecasting capabilities to be extended to encompass other resources. While the central focus is on forecasting CPU requirements for computationally intensive tasks, it is worth noting that memory forecasts can also be included in the overall scope. Requests for resource reservations are intrinsic to the communication between system modules and do not necessitate user input, given the system's self-adapting nature. Moreover, it is crucial to emphasize that the message producer can only trigger (send a resource reservation request) the message broker to prepare a resource reservation plan for a specific machine based on predictions once all resource forecasts have been successfully completed. For instance, if the forecasting objective involves two virtual machines, including predictions for both CPU and RAM resources, the system would comprise four independent segments encompassing modules 1–7 and a common segment integrating the *Reservation Module*.

Following the examination of the communication structure within a multi-virtual machine system, the primary focus was on conducting a comprehensive evaluation related to predicting a single resource usage (CPU) for a single virtual machine. Therefore, if only one resource on a single machine is subject to forecasting, the system operates in its basic variant, consisting of two segments that collectively encompass all eight *CRUOS* modules.

In order to conduct long-term forecasting of resource utilization for a selected virtual machine, it is crucial to have access to historical metric values. Typically, these data are collected from monitoring systems integrated into cloud operator services and are considered as raw data, which may include erroneous, inaccurate readings and outliers that have the potential to disrupt the operations of predictive models. As a result, the initial module in *CRUOS* is the *Anomaly Detection Module*, which addresses the identification of outlier samples. This module offers flexibility in terms of the data ranges to which the anomaly detection model is applied. It is essential to note that anomalies are defined as data samples that deviate significantly from the norm, often resulting in sharp spikes or dips in resource consumption. Conceptually, anomaly detection can be applied either globally to the entire dataset or locally to smaller data subsets (data chunks) independently. Such a configuration allows for multiple experiments to determine whether the outlier samples detected are undesirable or provide valuable insights.

Subsequently, the *Feature Enrichment Module* is responsible for enhancing data in various ways to create a robust data representation. When dealing with time series data, there are several techniques that can be implemented at this stage to extract the most valuable information from the dataset, ultimately benefiting the predictive model's results. However, it is important to notice that employing an excessively redundant data representation can lead to increased dimensionality. Therefore, various solutions can be considered, such as dynamic feature selection to reduce the feature set. This module is also tasked with preparing data in the appropriate format required by the prediction model applied.

The training of the prediction model takes place within the *Model Training Module*, where validation is an integral part of the process. Model validation can be performed using either the walk-forward validation technique or a separate dedicated validation set. Moreover, it is essential to note that in each subsequent iteration, this module can be optionally employed for retraining, with the purpose of fine-tuning the model on an expanding dataset to better capture the dynamics of the cloud environment. In practice, the most recent data are especially relevant for understanding changes in the cloud environment and thus should be included in the model's knowledge. However, it is imperative to make an informed decision regarding model retraining, particularly when reactively observing degradation or decline in the model's performance, or proactively identifying underlying shifts in data distribution. The model obtained during training (or updated during retraining) is then utilized in the subsequent *Prediction Module*. It is crucial to note that it is necessary to apply inverse transformations at this stage to obtain the data in the original scale. Multi-step long-term prediction can be achieved through various methods, with direct, iterative, and multi-output methods being popular options. These techniques offer different approaches to forecasting multiple steps into the future, providing the flexibility required to meet various requirements. The forecasts acquired are also forwarded to the *Prediction Module*.

After executing predictions within the *Prediction Module* and transferring the results to the second shared segment containing the *Reservation Module*, predictions are assessed. Within the *Monitoring Module*, periodic checks are conducted to validate predictions at intervals aligned

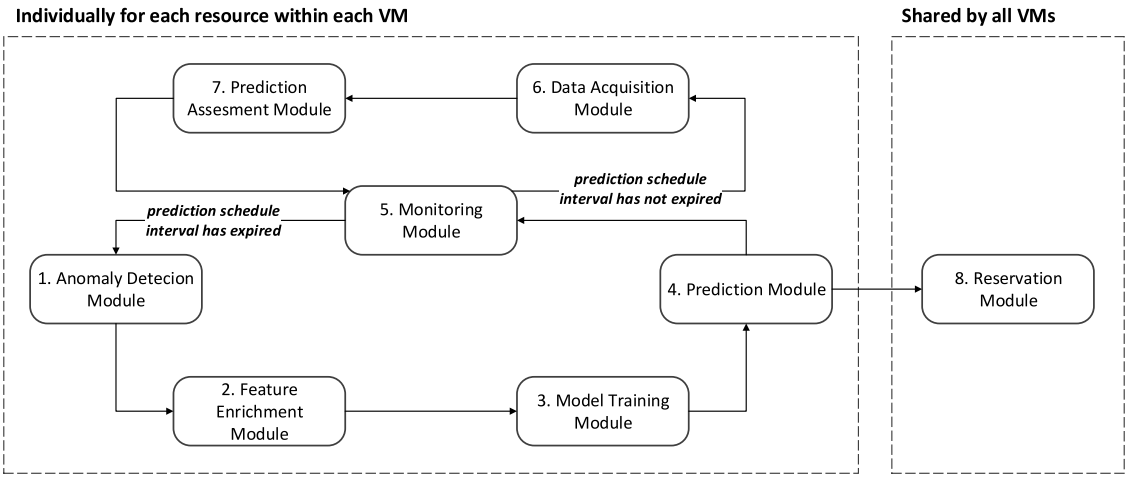


Fig. 1. Cloud resource usage optimization system.

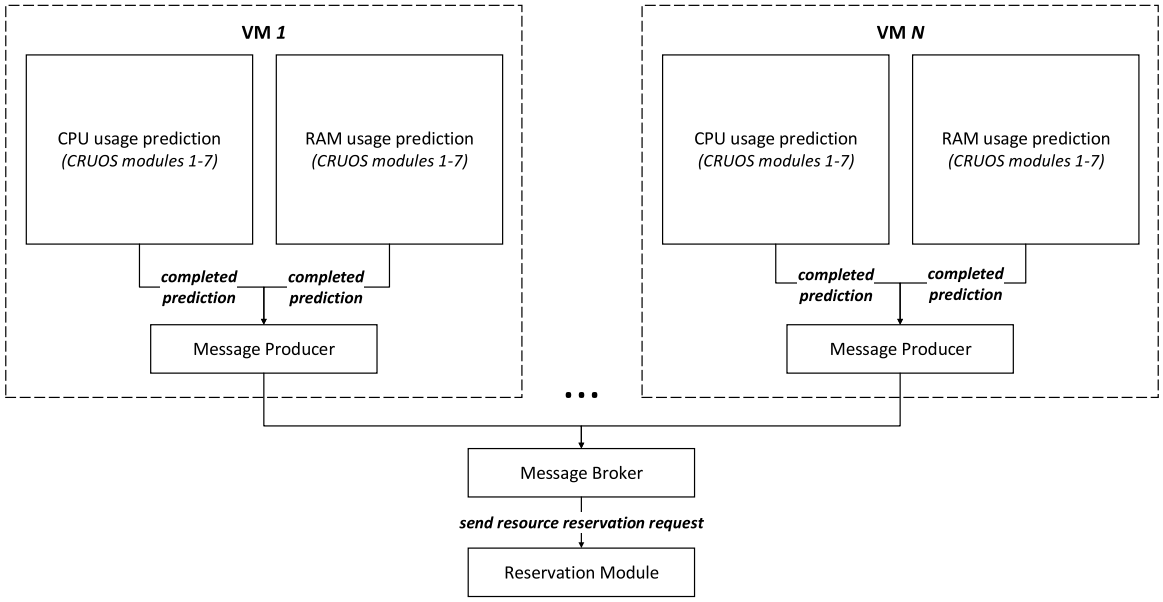


Fig. 2. Generalized communication diagram between CRUOS segments for a scenario with multiple virtual machines.

with the granularity of the metrics recorded . Additionally, if the *prediction schedule interval* configured has not expired, the system records resource usage levels and stores metrics within the *Data Acquisition Module* for the designated time frame. Subsequently, it checks the predictions against ground-truth usage through the *Prediction Assessment Module*. In general, the configurable *prediction schedule interval* can be synchronized with the prediction horizon. Consequently, a new system iteration can commence when the most recent predictions have expired. Upon detection by the *Monitoring Module* that the *prediction schedule interval* configured has elapsed, the system initiates a subsequent iteration of the first *CRUOS* segment. During this phase, the system acquires new historical data, which are examined for anomalies, enriched and optionally used to fine-tune the existing prediction model. This adaptation is particularly critical for dynamic and chaotic time series, where the model must conform to the current data distribution. In this manner, the first VM-specific segment operates within a continuous optimization loop, adjusting to newly recorded data and generating predictions for the configured time horizon. Flexibility and self-adaptation are of great significance in the context of HPC resources, facilitating the evaluation of various prediction schemes to determine the most suitable approach to future resource reservations.

An undoubtedly crucial aspect is the translation of prediction results into resource reservation plans, i.e., putting predictions into practical use for the flexible management of resources in HPC. This becomes a pivotal focus when considering the benefits of utilizing the *CRUOS*. Consequently, the central element in the context of FinOps is the *Reservation Module* (Fig. 3), which aims to map forecasts onto dynamic resource reservation plans. The *Reservation Module* itself consists of several elements, with its primary element being the *Controller*. At a high level, the module's operation commences with metadata retrieval, providing a detailed characterization of the reference virtual machine for which a resource reservation plan request has been originated. Following the high-level retrieval of metadata, it further becomes possible to access associated forecasts, generate resource reservation plans based on these, and estimate the potential financial impact resulting from the implementation of this plan.

When the *Controller* element requests metadata regarding a virtual machine for a reservation plan, it proceeds to the *Metadata* element. At the outset, a check is performed to determine if there are currently cached metadata about all virtual machines in the cloud inventory, including up-to-date information on their types, parameters and associated prices. If such data are not available, the *Inventory* element makes

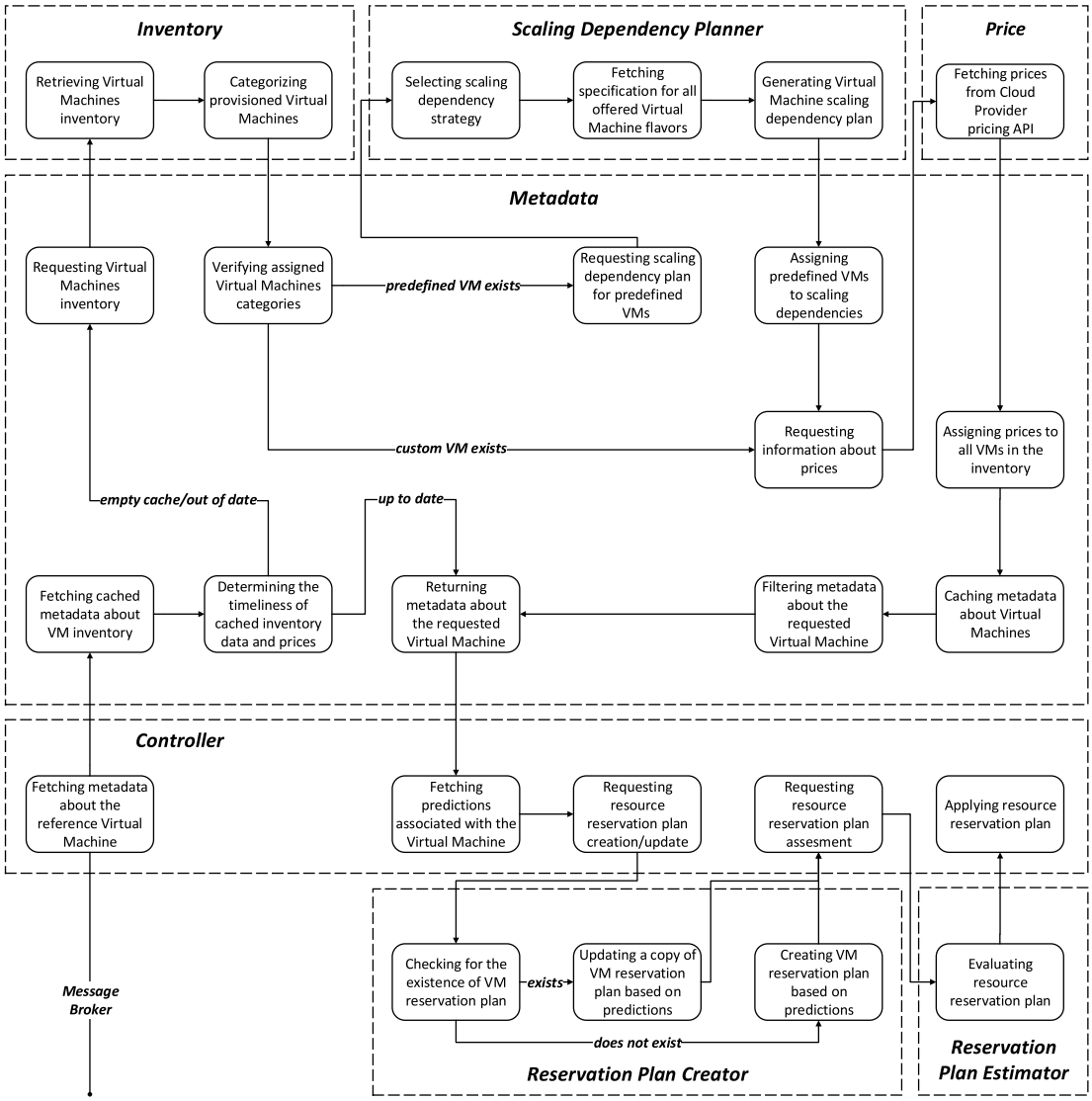


Fig. 3. Reservation module.

it possible to acquire the information needed and then categorizes machines into two groups: custom or predefined, as these are the two generic machine groups distinguished by Cloud Providers. Subsequently, if there is at least one predefined machine in the inventory, the *Metadata* element communicates with the *Scaling Dependency Planner* element, where a scaling dependency plan is established based on the specified strategy. The strategies considered can be default, custom, or hybrid. For example, in the case of the c2d-standard-32 machine, the default strategy provides for scaling following the hierarchy within the c2d-standard flavor. Therefore, it is possible to downsize to any c2d-standard machine type with lower specifications, particularly in terms of vCPUs. The mechanism is similar in the context of upscaling. However, it is feasible to use solely custom dependencies between different machine types in various flavors (custom strategy), or to extend default configurations with certain non-standard dependencies (hybrid strategy). The scaling dependency plan is applicable solely to machines of a predefined type, explicitly provided by cloud service providers. In the case of custom machines, hierarchy is unnecessary for determining resource requirements since vCPU and RAM needs can be directly specified according to the demand predicted. Subsequently, the *Price* element allows for fetching current prices, which are then cached alongside all gathered information. Next, the *Metadata* element returns only the requested machine's metadata (reference virtual machine) to the *Controller* element. In the following step, the *Controller* element retrieves all predictions associated with the virtual machine for which the request has been originated. Based on these predictions, the *Reservation Plan Creator* element constructs resource reservation plans. If a plan is already associated with the machine (typically when the *prediction schedule interval* is lower than the prediction horizon), it is modified and updated; otherwise, a new plan is created. Furthermore, the *Reservation Plan Estimator* element involves estimating the gains resulting from the implementation of the prepared plan. Finally, the *Controller* element approves the plan by submitting it to the cloud resource manager.

4. Evaluation

Regarding the evaluation of the *CRUOS*, two distinct stages were established. Initially, the focus was towards long-term time series forecasting, specifically targeting CPU usage prediction over a one-week period for a single virtual machine. The subsequent phase involved leveraging the prediction results for resource reservation plans, encompassing their comprehensive estimation and assessment, addressing both qualitative and quantitative aspects in line with FinOps principles. Consequently, the *Prediction results* section covers the operation of

the first seven modules of the *CRUOS* system. Furthermore, the *Cost-effectiveness results* section is specifically dedicated to the evaluation of the *Reservation Module*.

4.1. Prediction results

To evaluate the *CRUOS*, historical records of CPU usage collected from a real-life production environment were utilized. A particular emphasis was placed on this metric as it is considered crucial in the realm of high-performance computing alongside RAM. Consequently, the historical records represented consolidated CPU utilization over an 80-week period with hourly aggregation of data samples.

Initially, missing values were identified and addressed through imputation using a Simple Moving Average (SMA) with a window size of 168 samples, aligned with a one-week prediction horizon for long-term CPU forecasting. For data analysis purposes, the time series was split into distinct training, validation, and testing sets. In general, this intentional segmentation aims to implement specific actions on the training set, evaluate them using the validation set, and critically assess generalization capabilities on the test set without previous inadvertent knowledge transfer. Consequently, the training set comprised 10,920 samples, while 2520 samples were assigned to the testing set. Notably, the final 840 samples from the training set were initially allocated to the validation set. However, during the prediction method evaluation phase, walk-forward validation superseded single-set validation, proving notably effective for dynamic time series. Regarding the data analysis phase, the Jarque–Bera statistical test was conducted on the training set to assess its normality and skewness. The initial output value (58.96) represented the Jarque–Bera test statistic, indicating the deviation of sample data from a normal distribution; a higher value signifies a greater deviation from normality. Additionally, the p-value of 1.58×10^{-13} suggested significant evidence against the null hypothesis of the data conforming to a normal distribution. The verification that the data do not follow a normal distribution was crucial in the data analysis stage, influencing the further selection of an appropriate anomaly detection model. Furthermore, a stationary test indicated non-stationarity, implying that the variable under investigation does not exhibit consistent statistical properties over time. It was found that there was no singular principal seasonal or frequency component present, confirming the high dynamics of the data originating from cloud environments. This determination was supported by using the FFT (Fast Fourier Transform) and DWT (Discrete Wavelet Transform).

In the context of long-term forecasting, identifying outlier samples – sudden, localized spikes or dips in resource usage that do not permanently alter the time series trend – is crucial. Understanding whether these deviations negatively impact the prediction model or provide valuable insights is essential. Therefore, the primary module in the *CRUOS* system was the *Anomaly Detection Module*, utilizing an unsupervised Isolation Forest algorithm. Parameters, such as 90 estimators and a contamination fraction of 0.05, were determined through a grid search, while other parameters remained at default values. The strategy selected to handle outliers involved their removal. Although the option of imputing mean values for specific sample windows was considered, this approach was dismissed due to the comprehensive feature engineering that incorporated statistical parameters computed within moving windows.

In general, feature engineering is crucial for creating a robust data representation. Hence, the *Feature Enrichment Module* was utilized to generate lagged features from the samples from the previous week, calendar-based features (day of the week, month, hour, holidays), and statistical parameters calculated within distinct moving windows. Utilizing two window sizes (168 and 72), statistical attributes were computed from the prior week and the preceding three days, respectively. The statistical attributes derived encompassed metrics like median and mean alongside more intricate measures such as interquartile differences, the count of values above the mean, skewness and kurtosis. Window-based features were introduced both in the time domain and in the frequency domain after applying the Fast Fourier Transform to the time series. Additionally, cyclic features based on calendar parameters were enriched using the cosine and sine functions, resulting in a robust set of 250 features. Furthermore, this module introduced dynamic feature selection, enabling a focus on a specific subset of predictors. An excessive number of redundant features typically complicates prediction models without contributing to their performance or generalization capabilities.

Table 1
Evaluation metrics for weekly CPU predictions obtained using baseline methods throughout the entire test set period.

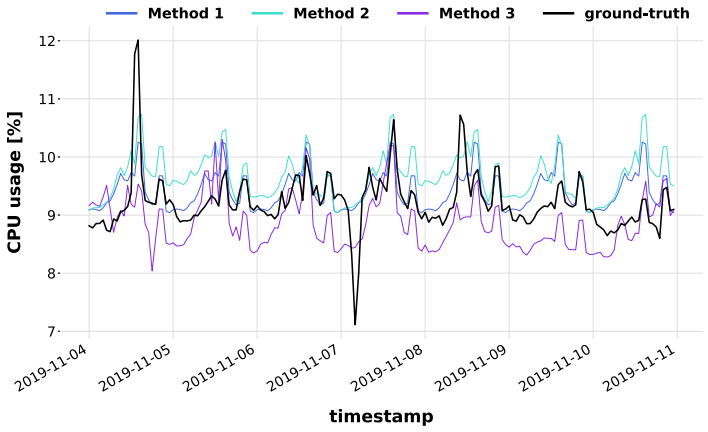
Ref.	RMSE	NRMSE	MAE	MAPE	MdAE	FR
1	2.223	0.141	0.914	0.267	0.327	0.661
2	2.168	0.137	0.936	0.276	0.374	0.623
3	2.339	0.148	0.999	0.261	0.415	0.349

Prior to evaluating the *CRUOS* with target prediction models, certain reference results were established using statistical methods, serving as baselines. Typically, naive prediction (repeating the last observed value throughout the entire prediction horizon) or more sophisticated methods like Simple Exponential Smoothing are considered benchmarks for long-term time series forecasting. However, methods based on Holt-Winters Seasonal Smoothing, incorporating a damping factor, were used as baselines, specifically *Method 1* (with a 1-day seasonality), *Method 2* (with a 3-day seasonality), and *Method 3* (with a 1-week seasonality). The selection of appropriate seasonalities that parameterized the chosen statistical model relied on an analysis of the autocorrelation function (ACF) and partial autocorrelation function (PACF). Additionally, decomposing the time series into trend and seasonality using both additive and multiplicative models served as additional tool to support parameterization. In general, the baseline models applied were fitted to the entire training set. Furthermore, at the end of each prediction horizon (as weekly predictions were iteratively made until the end of the test set was reached), the statistical methods were additionally refitted. In the case of baseline methods, neither anomaly detection nor feature engineering were employed.

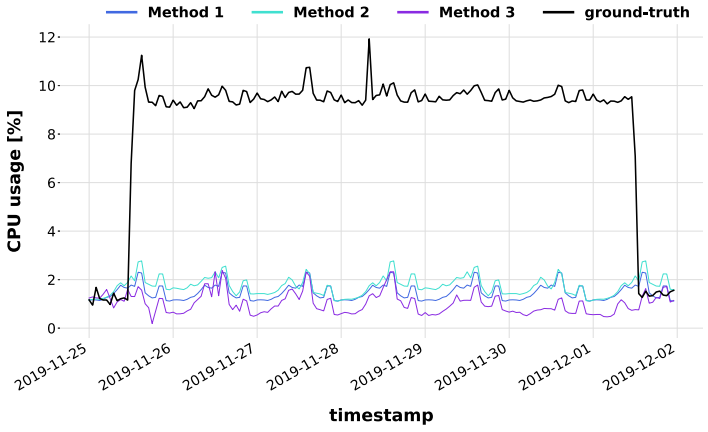
Fig. 4 illustrates the prediction results obtained using three baseline methods across three selected representative periods in the test set, compared to ground-truth CPU usage. Generally, simple methods can effectively capture utilization values in cases where the time series exhibit stability over a specific period, understood as relative constancy of parameters such as mean or standard deviation over time (Figs. 4(a) and 4(c)). However, naive methods – often assuming data linearity and stationarity – may struggle to forecast fluctuations in resource utilization, as observed in Fig. 4(b), where both a sudden increase and subsequent decrease within the given period were not predicted by these methods.

Considering the application of predictions, particularly in resource reservation planning, it is crucial to note that forecasts slightly exceeding the ground truth are notably more advantageous than those below the actual usage. While both overprovisioning and underprovisioning are undesirable, achieving a balance is necessary as there is considerably less risk in minor resource overestimation than in underestimation leading to service unavailability or violating SLAs. Consequently, an additional metric named FR has been introduced, indicating the ratio of forecasts equal to, or exceeding, the actual usage to the total number of forecasts made. This metric is crucial when analyzed alongside error metrics, since the aim is to achieve a high FR value coupled with low error metrics – essentially, aiming for forecasts close to the ground truth while favoring those above it. Consequently, the formula (1) represents the method for computing the FR metric, assuming n predictions ($\hat{y}_i$ – predicted value, y_i – ground truth value).

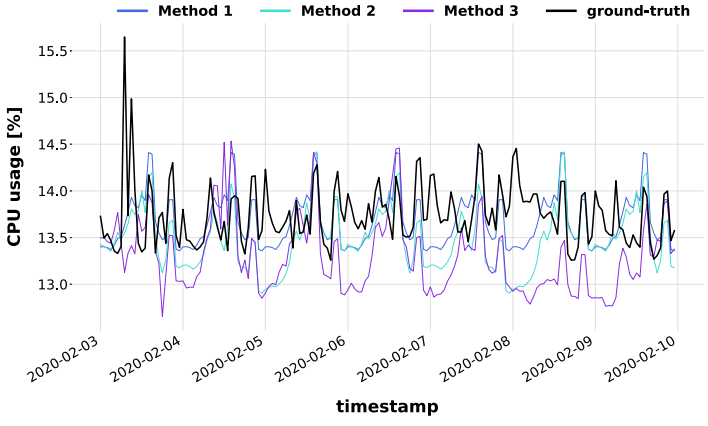
$$FR = \frac{\sum_{i=1}^{n} \mathbf{1}(\hat{y}_i \geq y_i)}{n} \tag{1}$$



(a) The first representative prediction period



(b) The second representative prediction period



(c) The third representative prediction period

Fig. 4. Visualization of weekly CPU prediction results obtained using the baseline methods for representative periods in the test set.

Consequently, Table 1 contains the evaluation metrics related to baseline methods and calculated for the entire test set. The first column refers to specific methods by their numerical identifiers. This convention of referring to individual methods remains consistent throughout the remaining tables. *Method 2* demonstrates the lowest RMSE (Root Mean Squared Error) and NRMSE (Normalized RMSE). Additionally, *Method 2* exhibits a slightly higher MAE (Mean Absolute Error) and MAPE (Mean Absolute Percentage Error) compared to *Method 1*. On the other hand, *Method 3* shows relatively poorer performance across most metrics, particularly evident in the MdAE (Median Absolute Error), where significant amplitude and oscillations in forecasts consistently place them below the ground truth, resulting in an unfavorably low FR. Consequently, both *Method 1* and *Method 2* offer similarly promising results in terms of evaluation metrics, with the second one chosen as the most favorable. Nevertheless, despite their relatively satisfactory evaluation metrics, these methods failed to predict sudden changes in resource consumption. This indicated the necessity to explore more advanced prediction methods, especially in HPC systems, where overprovisioning incurs high costs and underprovisioning leads to undesirable drops in Quality of Service (QoS).

Having established a robust reference point with the baseline methods, the subsequent model incorporated into both the *Training Module* and *Prediction Module* was XGBoost. Parameter selection was conducted using grid search, ultimately obtaining 240 estimators, a maximum tree depth of 7 and a learning rate of 0.28 by leveraging walk-forward validation. Moreover, a single-step-ahead prediction approach was adopted. Consequently, to enable long-term forecasting, an iterative approach was employed, utilizing each prediction autoregressively as a training sample for the subsequent step until the prediction horizon was reached. Additionally, in each iteration, the samples predicted were individually enriched to prevent any information leakage. In XGBoost-based methods (a total of eleven prediction methods), all features available after the feature enrichment stage were utilized. The objectives associated with the methods based on XGBoost were basically twofold: first, to assess the impact of anomalous samples; second, to verify the necessity of periodic retraining, potentially preventing degradation in performance over time. Similar to the baseline methods, the *prediction schedule interval* was set to 7 days, triggering the creation of new forecasts upon the expiration of the preceding one.

Firstly, *Method 4* reflected a basic *CRUOS* configuration, operating without the *Anomaly Detection Module* and lacking periodic model retraining. Subsequently, *Method 5*, *Method 6*, and *Method 7* expanded upon this by integrating anomaly detection within the training set and handling the outliers detected through a removal strategy. The distinction among these methods lay in the data scope for the Isolation Forest: it was applied to the entire training set, 4-week chunks and weekly chunks, respectively, allowing for the examination of local and global impacts of outlier samples. Additionally, *Method 8* involved refitting the model after each configured *prediction schedule interval* expired, just before making new predictions. This process incorporated additional data available after each *prediction schedule interval*, updating the model with the use of an expanding historical dataset. Its purpose was solely to verify retraining values without the detection of anomaly samples. Following this established schema, *Method 9*, *Method 10*, and *Method 11* adopted a periodic re-training approach, integrating anomaly detection for the entire training set, 4-week chunks, and 1-week chunks, respectively. Ultimately, the remaining three methods based on XGBoost – *Method 12*, *Method 13*, and *Method 14* – essentially extended *Methods 9*, *10*, and *11*. An additional factor involved anomaly detection being applied to newly collected data before each refit, ensuring the scanning of all data used for model training for potential outliers.

Fig. 5 presents prediction results using XGBoost-based methods for representative periods from the test set. Similar to the baseline methods, when resource usage values remained stable, it is evident that the prediction characteristics of the methods applied closely matched ground-truth CPU usage values (Fig. 5(a)). Notably, Fig. 5(b) showcases results that are particularly interesting and crucial to the problem at hand. Here, the methods reacted properly to an increase in consumption, closely approaching the actual values recorded – a scenario completely missed by the baseline methods. These predictions dynamically followed the consumption trend, with various methods reacting differently, each responding to the potential change in resource usage trends. An important observation concerns the visualized predictions in Fig. 5(c), particularly regarding *Method 6* and *Method 7*, which exhibited a susceptibility to accumulating errors. This was primarily due to the iterative forecasting approach and the absence of periodic refitting, coupled with the removal of outlier samples ultimately deemed crucial based on established scenarios. Given that CPU usage metrics were recorded hourly and then aggregated into averages, these values did not represent randomness or measurement errors.

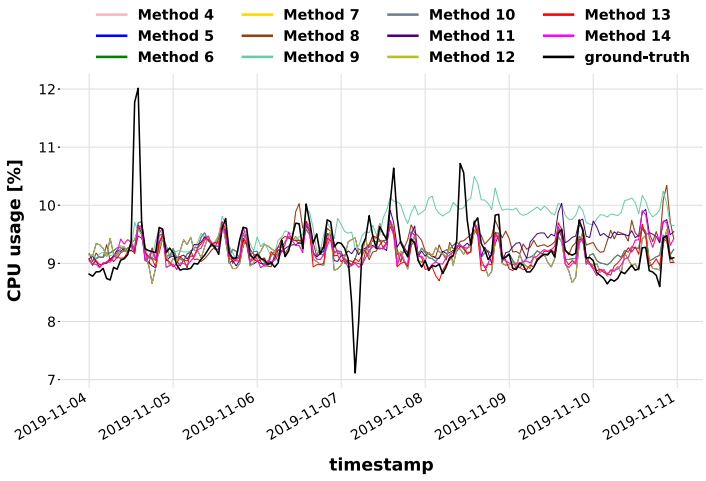
Reflecting on the evaluation metrics presented in Table 2, it was confirmed that anomaly detection within this specific CPU dataset did not contribute to improved accuracy. Consequently, *Method 4* outperformed *Methods 5*, *Method 6*, and *Method 7* due to the loss of valuable information and a significant reduction in accuracy when anomalies were removed. Among XGBoost-based methods, *Method 8*, which involved adapting to new data without anomaly detection, demonstrated superior performance in terms of the RMSE, NRMSE, and MAE metrics. Other methods that included refitting but also removed anomalies performed worse. However, it is apparent that refitting helped reduce errors, as exemplified by the comparison between, for instance, *Method 5* and *Method 13*, which differed in this regard. Additionally, it is important to note that assessing methods solely based on the FR metric is impractical, as finding a method with both low errors and a relatively high FR is crucial. This is evident in the best-performing XGBoost-based method – *Method 8*.

Table 2
Evaluation metrics for weekly CPU predictions obtained using XGBoost-based methods throughout the entire test set period.

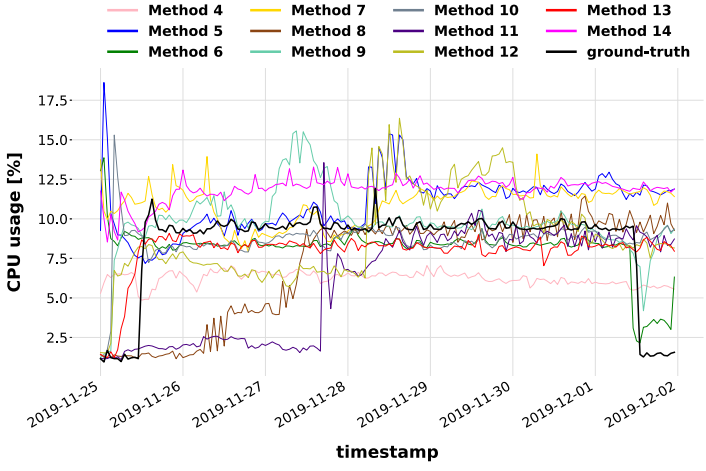
Ref.	RMSE	NRMSE	MAE	MAPE	MdAE	FR
4	2.477	0.157	1.355	0.502	0.407	0.481
5	6.757	0.428	5.756	3.316	7.051	0.822
6	14.762	0.936	9.352	5.121	1.882	0.73
7	19.45	1.233	13.565	7.819	5.353	0.743
8	1.814	0.115	0.909	0.335	0.335	0.549
9	2.701	0.171	1.359	0.526	0.332	0.569
10	2.573	0.163	1.185	0.58	0.321	0.519
11	2.332	0.148	1.082	0.317	0.325	0.498
12	2.627	0.167	1.195	0.546	0.321	0.524
13	3.209	0.203	1.851	0.75	0.419	0.534
14	3.209	0.192	1.399	0.646	0.334	0.572

Considering the inconveniences associated with single-step ahead methods, particularly their susceptibility to error accumulation, the decision was made to expand the prediction methods by incorporating neural networks. In line with this decision, anomaly detection and the removal of outlier samples were omitted. Instead, a direct, multi-output prediction approach was employed, allowing for the simultaneous prediction of values across the entire forecast horizon in a single step. Long-term time series forecasting can be approached as a sequence prediction task where state-of-the-art methods encompass recurrent networks, particularly architectures employing LSTM cells and their simpler counterparts, GRUs. Consequently, these architectures were considered, along with bidirectional layers, leading to the exploration of BiLSTM and BiGRU models. An essential aspect was the incorporation of the ADAM optimizer, Dropout layers, and Early Stopping mechanism that allows for training to be interrupted on the basis of the error metrics monitored on the validation set. For neural network-based methods, a walk-forward validation approach was employed, focusing notably on favoring simpler architectures. As periodic retraining enables adaptation to the dynamics of the cloud environment, simpler models also translate to lower costs of the *CRUOS* system. Moreover, a dynamic feature selection mechanism was shown to play a vital role in limiting the model to only the most relevant subset of features, contributing to simpler models.

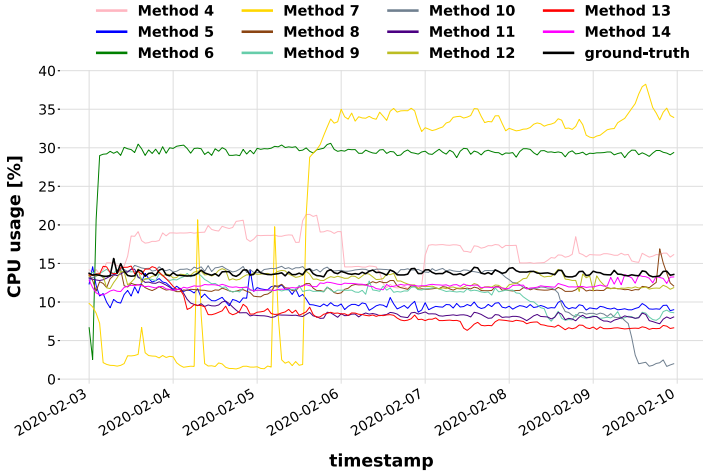
In regard to neural network-based methods, their parameters were established through grid search and walk-forward validation, outlined in Table 3. Various architectures utilizing non-linear activation functions were explored, considering cost implications and *CRUOS* adaptability. Therefore, the *Layers* column indicates the layers utilized and their order. For all layers, the number in parentheses identifies the count of LSTM or GRU cells. The only exception is the Dropout layer, where the fraction of neurons randomly dropped during the learning process is specified in parentheses. In general, recurrent neural networks require a three-dimensional input sequence shape specified as (batch size, time steps, number of features). To examine how the length of time steps affects the model's ability to capture temporal dependencies, sequences of 24, 72, and 168 samples were specifically evaluated, representing daily, three-day, and weekly time frames, respectively. As for the loss functions, besides MSE, the Huber Loss function was favored in the grid search due to its reduced sensitivity to outliers. Additionally, the table detailing the parameterization of neural



(a) The first representative prediction period



(b) The second representative prediction period



(c) The third representative prediction period

Fig. 5. Visualization of weekly CPU prediction results obtained using the XGBoost-based methods for representative periods in the test set.

network-based methods includes the number of training epochs used in each step of walk-forward validation and retraining.

The weekly CPU usage predictions obtained using neural network-based methods are presented in Fig. 6. The first noticeable trend is a significantly reduced occurrence of error accumulation effects, which were characteristic of iterative approaches, supporting the prevalence of the multi-output approach. It is also worth noting that *Method 15* and *Method 16* did not employ dynamic feature selection, resulting in larger models, as shown in Table 3. Since both *Method 15* and *Method 16* did not yield significantly better error metrics, this confirmed that introducing an excessive number of redundant features was not beneficial, as it led to larger models. Consequently, the introduction of

Table 3
Parameterization of neural network-based methods.

Ref.	Layers	Loss function	Activation function	Learning rate	No. of epochs	Input sequence shape
15	LSTM (64) → LSTM (32) → Dense (168)	MSE	ReLU	0.0001	10	(64, 168, 90)
16	GRU (16) → GRU (8) → Dense (168)	MSE	Tanh	0.0001	10	(64, 168, 90)
17	LSTM (16) → Dropout (0.2) → Dense (168)	MSE	ReLU	0.001	20	(64, 24, 37)
18	LSTM (16) → Dropout (0.2) → LSTM (8) → Dropout (0.2) → Dense (168)	MSE	ReLU	0.001	20	(64, 24, 37)
19	BiLSTM (8) → Dropout (0.2) → BiLSTM (4) → Dropout (0.2) → Dense (168)	MSE	ReLU	0.0001	20	(64, 72, 37)
20	BiGRU (16) → Dropout (0.2) → BiGRU (8) → Dropout (0.2) → Dense (168)	Huber	Tanh	0.0001	20	(32, 72, 37)
21	BiGRU (8) → Dropout (0.3) → BiGRU (4) → Dropout (0.3) → Dense (168)	Huber	Tanh	0.0001	20	(64, 168, 37)
22	BiLSTM(8) → Dropout (0.3) → BiLSTM (4) → Dropout (0.3) → Dense (168)	Huber	Tanh	0.0001	20	(64, 168, 37)
23	GRU (16) → Dropout (0.3) → GRU (8) → Dropout (0.3) → Dense (168)	Huber	ELU	0.001	20	(64, 168, 37)
24	GRU (16) → Dropout (0.3) → GRU (8) → Dropout (0.3) → Dense (168)	Huber	ELU	0.001	20	(64, 168, 90)

Table 4
Evaluation metrics for weekly CPU predictions obtained using neural network-based methods throughout the entire test set period.

Ref.	RMSE	NRMSE	MAE	MAPE	MdAE	FR
15	4.262	0.290	2.883	1.613	1.285	0.614
16	4.905	0.334	2.346	1.166	0.889	0.593
17	3.385	0.230	2.289	1.190	1.208	0.697
18	5.957	0.405	3.644	1.835	1.223	0.863
19	6.133	0.417	4.014	1.935	1.689	0.783
20	5.390	0.367	3.411	1.926	1.425	0.818
21	1.79	0.122	1.404	0.663	1.207	0.792
22	2.682	0.182	1.694	0.796	0.651	0.794
23	3.234	0.220	2.686	1.405	2.627	0.736
24	2.081	0.142	1.552	0.708	1.078	0.877

Table 5
Evaluation metrics for weekly CPU predictions obtained using *Method 25 — Quantile 0.50* (TFT-based method) throughout the entire test set period.

Ref.	RMSE	NRMSE	MAE	MAPE	MdAE	FR
Q 0.02	1.403	0.095	0.505	0.234	0.268	0.604
Q 0.10	1.445	0.098	0.543	0.246	0.292	0.674
Q 0.25	1.464	0.100	0.559	0.251	0.301	0.689
Q 0.50	1.487	0.101	0.582	0.256	0.313	0.712
Q 0.75	1.513	0.103	0.607	0.263	0.321	0.731
Q 0.90	1.552	0.106	0.643	0.272	0.339	0.755
Q 0.98	1.714	0.117	0.773	0.305	0.412	0.815

dynamic feature selection in methods 17 through 24 did not result in deterioration; instead, it helped employ simpler models. Additionally, introducing excessively short time sequences into the model caused an underestimation problem (methods from *Method* 17 to *Method* 20). Moreover, *Method 24* was an experimental representative exploring the influence or necessity of lagged features and subsequent feature selection. However, it did not enhance performance, as recurrent networks modeled the dependencies well with a proper sequence length of seven days. Analyzing the evaluation metrics presented in Table 4, it is evident that *Method 21*, which leverages the BiGRU architecture, exhibits significantly better metrics compared to others.

With the BiGRU architecture exhibits significantly better metrics compared to others. Bidirectional layers, enabling modeling of samples concerning both past and future contexts, the utilization of feature selection, and an appropriate sequence length enabled the network to produce forecasts close to the ground truth, demonstrating a high FR with relatively low error metric values. This shows that promising results with respect to resource reservation plans can be produced using of this method.

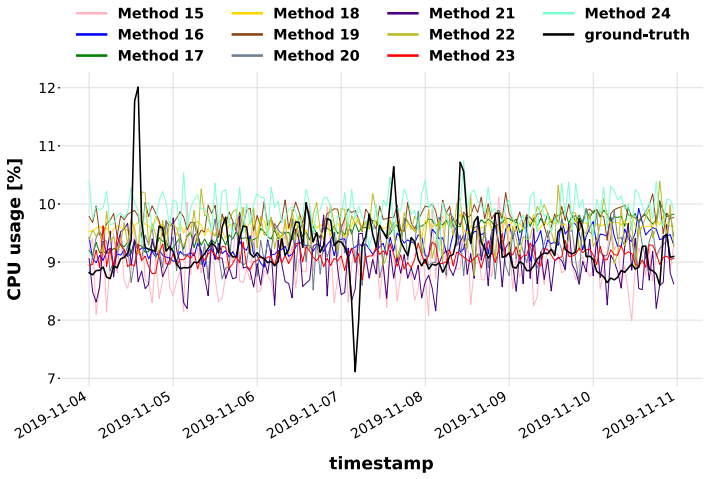
The last method employed in evaluating prediction results was *Method 25*, leveraging a Temporal Fusion Transformer tailored for time series forecasting. The motivation behind its application was to verify the self-attention mechanism that revolutionized sequence prediction in the natural language processing domain. Consequently, four attention heads were defined, and the number of neurons in the hidden layer was set to 160 for both continuous and discrete variable processing. To ensure regularization, Dropout layers were utilized that randomly deactivated 10% of neurons. Fig. 7 illustrates the prediction outcomes attained by *Method 25*, utilizing the Quantile Loss function for representative periods within the test set, contrasted against actual CPU utilization. The performance proved highly accurate, and the extreme quantiles presented enabled the depiction of the confidence interval alongside the 0.5 quantile. Upon analyzing forecasts and error metrics (Table 5), it is evident that *Method 25* achieved the best results. The usage of the transformer with dynamic feature selection, though resulting in a significantly higher model complexity, translated into satisfactory evaluation metrics and a relatively high FR.

Establishing baseline methods was a crucial step before employing more complex predictive models. Even where evaluation metric values are promising, it is essential to analyze them alongside forecast results to identify scenarios where baselines are not applicable. Using XGBoost-based methods as an example, iterative forecasting may lead to error accumulation, which is notably less frequent in direct, multi-output approaches. In the analyzed dataset, the outliers identified turned out to carry significant information. This, coupled with periodic model retraining, resulted in the best-performing XGBoost-based method – measured in terms of RMSE – being better by 16.3% compared to *Method 2*. Considering the application of models in a production context, it is crucial to focus on the profitability and cost of the *CRUOS*, which results in favoring simpler and less complex prediction methods. In this case, the concept of dynamic feature selection – relying only on a subset of the most impactful features – becomes a significant focal point. Furthermore, *Method 21* and *Method 25 – Q 0.50* exhibited improvements of 17.4% and 31.4%, respectively, in comparison to the RMSE of *Method 2*. While the transformer-based approach generally yielded superior results, it was also the most complex and probably the costliest if applied in *CRUOS*, and the methods required to interpret its results were more advanced as well. It is also noteworthy that FR analysis favored neural network-based methods despite their marginally worse error metrics compared to TFT.

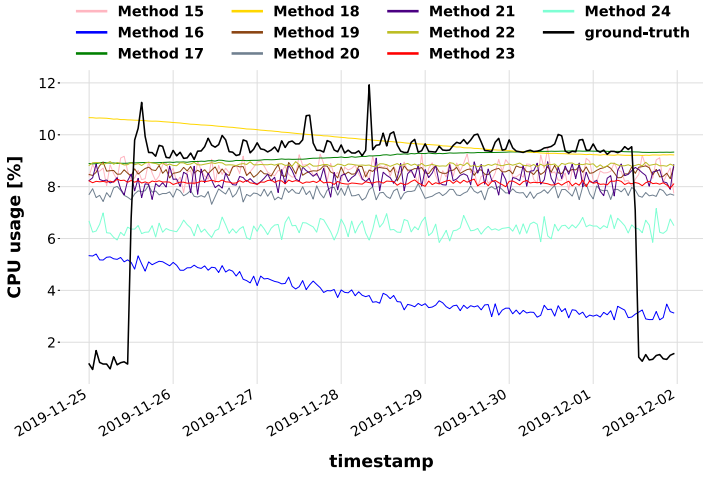
In Table 6, a summary of selected characteristics of the applied prediction methods is presented, taking into account the presence or absence of anomaly detection, retraining and dynamic feature selection. The analysis of the results revealed that dynamic feature selection and periodic retraining had a positive impact on the performance of prediction methods.

4.2. Cost-effectiveness results

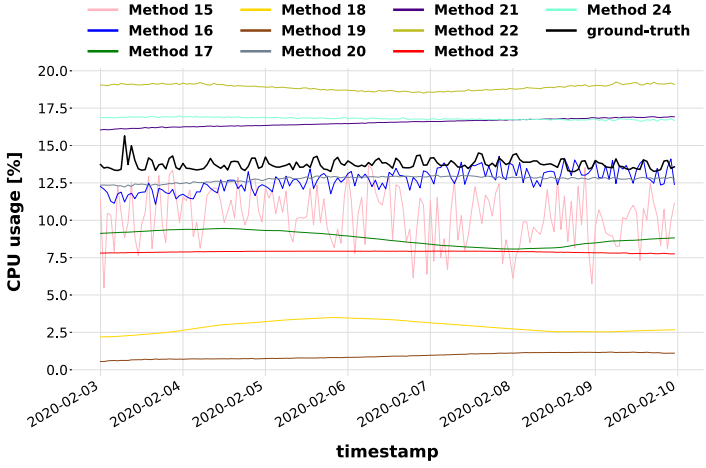
Although accurate long-term time series forecasting poses a significant challenge, it holds a considerably greater potential for resource reservation compared to the short-term approach. The longer the forecasting horizon, the more informed decisions can be made, although this often comes with reduced accuracy. Therefore, when evaluating prediction results for resource reservation, it is crucial to assess whether methods consistently producing the best error metrics are always the



(a) The first representative prediction period



(b) The second representative prediction period

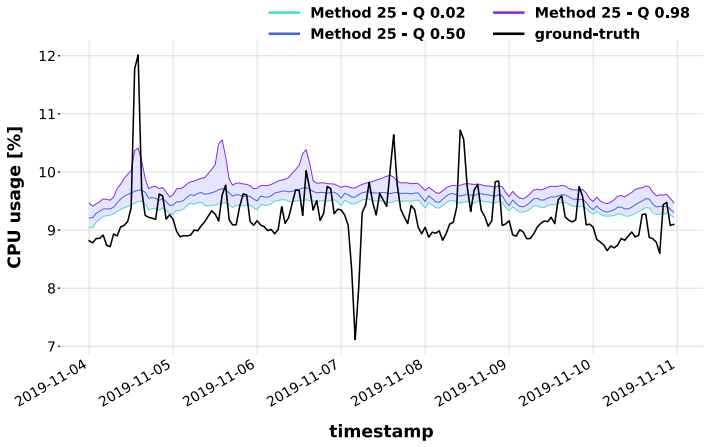


(c) The third representative prediction period

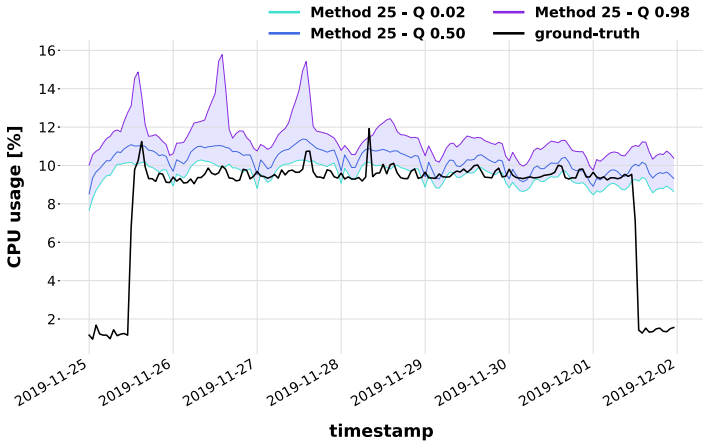
Fig. 6. Visualization of weekly CPU prediction results obtained using the neural network-based methods for representative periods in the test set.

optimal choices for generating reservation plans. Consequently, to assess cost-effectiveness, focus was directed solely towards representative methods from each group – *Method 2*, *Method 8*, *Method 21*, and *Method 25*.

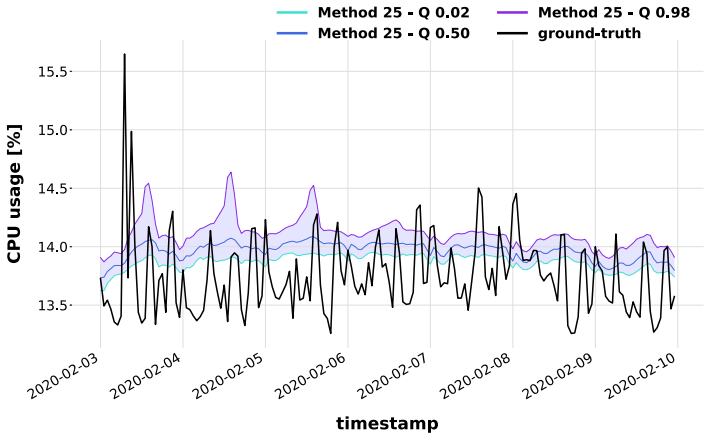
The *CRUOS* system was configured with a 7-day *prediction schedule interval* (equivalent to the prediction horizon). However, a more effective approach for resource reservation plans might involve more frequent forecasting, such as a daily *prediction schedule interval*. Nonethe-



(a) The first representative prediction period



(b) The second representative prediction period



(c) The third representative prediction period

Fig. 7. Visualization of weekly CPU prediction results obtained using the TFT-based method for representative periods in the test set.

less, to illustrate the core concept of aligning forecasts with reservation plans, the decision was made to generate a new forecast upon the expiration of the previous one. The selection of the *prediction schedule interval* constitutes an individual system parameter and depends on the conditions under which it operates.

In the cost-effectiveness evaluation, the emphasis was on Google Cloud Platform's virtual machines tailored for HPC, which are of the c2d-standard flavor. Nonetheless, the analysis methodology employed remains universally applicable across various machine types and different cloud service providers.

The long-term time series forecasting integrated into *CRUOS* focused on predicting the percentage of CPU usage rather than specific vCPU values. This had a significant impact on the methodology used for estimating cost-effectiveness. Relying on percentage-based forecasts made it possible to introduce a fundamental assumption solely for the purpose of the conducted research and experiments. Consequently, even though the historical data used in *CRUOS* are derived from

Table 6
Summary of selected characteristics of applied prediction methods.

No.	Anomaly detection	Retraining	Dynamic feature selection
1	No	Yes	No
2	No	Yes	No
3	No	Yes	No
4	No	No	No
5	Yes	No	No
6	Yes	No	No
7	Yes	No	No
8	No	Yes	No
9	Yes	Yes	No
10	Yes	Yes	No
11	Yes	Yes	No
12	Yes	Yes	No
13	Yes	Yes	No
14	Yes	Yes	No
15	No	Yes	No
16	No	Yes	No
17	No	Yes	Yes
18	No	Yes	Yes
19	No	Yes	Yes
20	No	Yes	Yes
21	No	Yes	Yes
22	No	Yes	Yes
23	No	Yes	Yes
24	No	Yes	Yes
25	No	Yes	Yes

a specific type of virtual machine, working with percentage usage permits these predictions to be universally applied and linked with any machine type. Furthermore, the concept of a reference virtual machine type was introduced. For instance, when the reference type is set as c2d-standard-4, it is assumed that the forecasted percentage usage corresponds to the number of vCPUs associated with that specific machine type. Naturally, the same levels of percentage CPU usage translate into different vCPU demand for various virtual machine types. With this assumption, it was possible to conduct experiments flexibly by setting different types of reference virtual machines. This facilitated the illustration of diverse resource reservation plans based on the reference machine selected. Therefore, with a focus on HPC, simulations were conducted for each predefined c2d-standard machine type available in the europe-west-2b region.

As a result, each virtual machine type within the e2d-standard flavor was designated as a reference type in its own distinct simulation. Following this, prediction results from *Method 2*, *Method 8*, *Method 21*, and *Method 25 — Quantile 0.50* were separately employed to dynamically determine resource reservation plans for each reference type. Thus, resource reservation plans resulted from dynamically mapping forecasts to the respective virtual machine flavors according to the scaling dependency strategy defined in the *Scaling Dependency Planner*. Since we exclusively considered predefined machines, the employed strategy was the default one. Consequently, c2d-standard-4 could be primarily downsized to c2d-standard-2 or upsized to c2d-standard-8. Subsequently, following the same approach, c2d-standard-8 could be downsized to c2d-standard-4 or upsized to c2d-standard-16 to meet the predicted demands. For instance, c2d-standard-112 can be downscaled to any type below (withing c2-standard), but it cannot be upscaled, as it is the most robust type within the c2d-standard category. For this reason, the Reservation Module remains flexible concerning possible scaling strategies for predefined machines.

Table 7 presents qualitative and quantitative assessment outcomes of the resource reservation plans generated based on prediction results from *Method 2*. These assessments were carried out across diverse simulations involving various reference types of HPC-oriented virtual machines. The initial column indicates the reference virtual machine type to which the predictions were related. Subsequently, the *Total price (without CRUOS) in USD* column represents the expenses accrued by the referenced machine type throughout the entire test dataset duration, assuming no *CRUOS* usage. The prices are in USD and were calculated based on rates retrieved from the Google Cloud Platform pricing API as at November 5th, 2023. On the other hand, the *Total price with CRUOS in USD* reflects the cost incurred when dynamically switching from the reference type to different types based on forecasted demand. Moreover, the two subsequent columns present the average prices per hour assuming both *CRUOS* usage and no *CRUOS* usage. The *Relative gain* column represents the percentage by which costs would be reduced when using *CRUOS*. Following FinOps principles, implementing *CRUOS* results in gains for both small instances with a low vCPU count and those with larger resources where wastage is significantly more costly. It is worth noting that these results are estimated values, yet they undeniably underscore the essence of resource reservation in accordance with predicted demand. Despite promising results, several potential negative effects should be considered that may arise as a result of reservations based on forecasts. Firstly, typically, changing machine types requires machine shutdown, potentially resulting in several minutes of downtime. Consequently, the *No. of re-scalings* column indicates how many times such changes would occur during the duration of the test dataset. While the concept of reservation based on forecasts involves rescaling, it is essential to minimize operational overhead. This might inspire the implementation of strategies to avoid frequent switching between types when such changes do not yield significant financial benefits. Secondly, predictions in dynamic cloud computing environments are particularly susceptible to inaccuracies. Hence, close monitoring and the above suggestion that the *prediction schedule interval* be configured daily rather than being aligned with the prediction horizon are imperative. More frequent forecasts allow for more frequent corrections and minimize the probability of violating SLAs. Therefore, the *No. of SLA violations* column represents the number of times during the test dataset duration where the machine type would not meet requirements, i.e., the ground truth would surpass the reserved resources.

Accordingly, Fig. 8 shows the outcomes of mapping weekly predictions of CPU usage onto *Method 2*, relating them to c2d-standard-112 as the reference machine. Instead of using all 112 vCPUs for the entire duration, various machine types were selected in the dynamic resource reservation plan based on the forecasted demand. Unlike in prediction visualization, the Y-axis represents the number of vCPUs rather than the percentage of CPU usage. Among the various available reference machine types, the decision was made to visualize the reservation plan for c2dstandard-112, which remained consistent across the subsequent visualizations. This choice was due to the fact that it provided the clearest illustration of transitions between different machine types, demonstrating that in the case of Fig. 8, adhering entirely to the forecasts would not require machines larger than c2dstandard-32.

The evaluation of the *Reservation Module* was also conducted across the remaining representative prediction methods in relation to different reference virtual machine types. Consequently, Table 8 presents the results of reservation plan evaluation for *Method 8*. The risk reflected by the *No of SLA violations* column is remarkably similar to that observed in results for *Method 2*. This resemblance arises from the iterative nature of transitioning from a one-step ahead forecast to the intended weekly prediction horizon, which leads to accumulated errors and fluctuations in forecast outcomes. Fig. 9 illustrates how the forecasts were translated into machine types through the operation of the *Reservation Module*. The visualization specifically focuses on c2d-standard-112 due to the heightened visibility of transitions in robust machines with higher computational capabilities. Additionally, the visualization highlights a substantial count of adverse rescaling events that reflect these regular shifts between machine types. This is contingent upon the specific environment and requires fine-tuning of the *Reservation Module* based on the needs and the time series considered.

Similarly to the convention used for *Method 2* and *Method 8*, Table 9 presents qualitative and quantitative assessment results of the dynamic

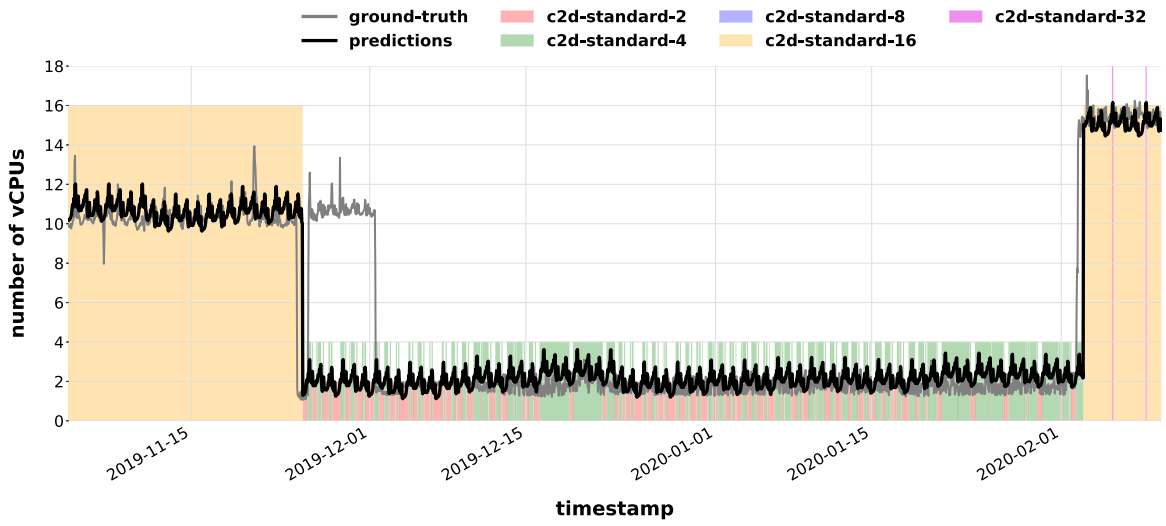


Fig. 8. Weekly CPU usage predictions obtained using *Method 2* (baseline method) and their mapping to virtual machine types for the reference machine set as *c2d-standard-112*.

Table 7
Reservation plan evaluation results: quantitative and qualitative analysis of diverse reference HPC-purpose machine types in relation to weekly CPU usage forecasts obtained using *Method 2* (baseline method).

Reference VM type	Total price (without *CRUOS*) in USD	Total price (with *CRUOS*) in USD	Avg. price per hour (without *CRUOS*) in USD	Avg. price per hour (with *CRUOS*) in USD	Relative gain	No. of re-scalings	No. of *SLA* violations
c2d-standard-2	275.128	275.128	0.117	0.1	0.0%	0	0
c2d-standard-4	550.255	275.128	0.234	0.117	50.0%	0	0
c2d-standard-8	1100.510	275.128	0.468	0.117	75.0%	0	0
c2d-standard-16	2201.020	294.780	0.936	0.125	86.607%	1	12
c2d-standard-32	4402.041	393.039	1.872	0.167	91.071%	2	159
c2d-standard-56	7703.571	512.823	3.275	0.218	93.343%	6	169
c2d-standard-112	15 407.143	940.370	6.551	0.400	93.897%	267	286

Table 8
Reservation plan evaluation results: quantitative and qualitative analysis of different reference HPC-purpose machine types in relation to weekly CPU usage forecasts obtained using *Method 8* (XGBoost-based method).

Reference VM type	Total price (without *CRUOS*) in USD	Total price (with *CRUOS*) in USD	Avg. price per hour (without *CRUOS*) in USD	Avg. price per hour (with *CRUOS*) in USD	Relative gain	No. of re-scalings	No. of *SLA* violations
c2d-standard-2	275.128	275.128	0.117	0.117	0.0%	0	0
c2d-standard-4	550.255	275.128	0.234	0.117	50.0%	0	0
c2d-standard-8	1100.510	275.128	0.468	0.117	75.0%	0	0
c2d-standard-16	2201.020	277.116	0.936	0.118	87.41%	12	163
c2d-standard-32	4402.041	371.282	1.872	0.158	91.566%	17	211
c2d-standard-56	7703.571	571.194	3.275	0.243	92.585%	17	70
c2d-standard-112	15 407.143	1013.597	6.551	0.431	93.421%	215	318

resource reservation plan based on the representative neural network-based method – *Method 21*. Clearly, financial benefits derived from *CRUOS* usage with the appropriate reference type depend on CPU forecasts, offering insight into the advantages of proactive forecasting over traditional machine reservation methods. As concerns the forecasts from *Method 21* employed in the *Reservation Module*, there is a significant reduction in rescaling operations and SLA violations compared to previous methods. This result is promising as no additional strategies were implemented that were aimed at safeguarding against SLA violations. Fig. 10 illustrates the translation of predictions into individual virtual machine types against ground-truth usage. Transitions between machine types are clearly significantly smoother, and the number of such re-scaling events is considerably lower compared to the representative methods previously examined. This outcome is advantageous as it suggests that reservation plans reduce the risk associated with frequent rescaling, minimizing additional overhead and downtime related to the necessity of shutting down the machine to alter its parameters.

Table 10 presents assessment results of the resource reservation plan derived from the latest method, *Method 25 — Quantile 0.50*, for various reference VM types. Despite the superior error metric performance of the TFT model in terms of forecasting accuracy, the resulting reservation plans based on it appeared to be less efficient. This observation is critical, indicating that the most accurate method in forecasting may not necessarily produce the best plans in terms of practical applicability. Creating resource reservation plans directly based on forecasts resembling ground-truth usage poses a risk of potentially increasing SLA violations. Hence, the relationship between a high FR and low error metrics appeared to offer a natural safeguard against these concerns in the representative neural network-based method. This result also provides insights into potential enhancements in the *Reservation Module* by introducing a certain additional margin (intentionally providing slightly more resources than forecasted), minimizing the risk of SLA violations but also preventing machines from reaching high CPU usage, which could lead to performance drops. It is a tradeoff between

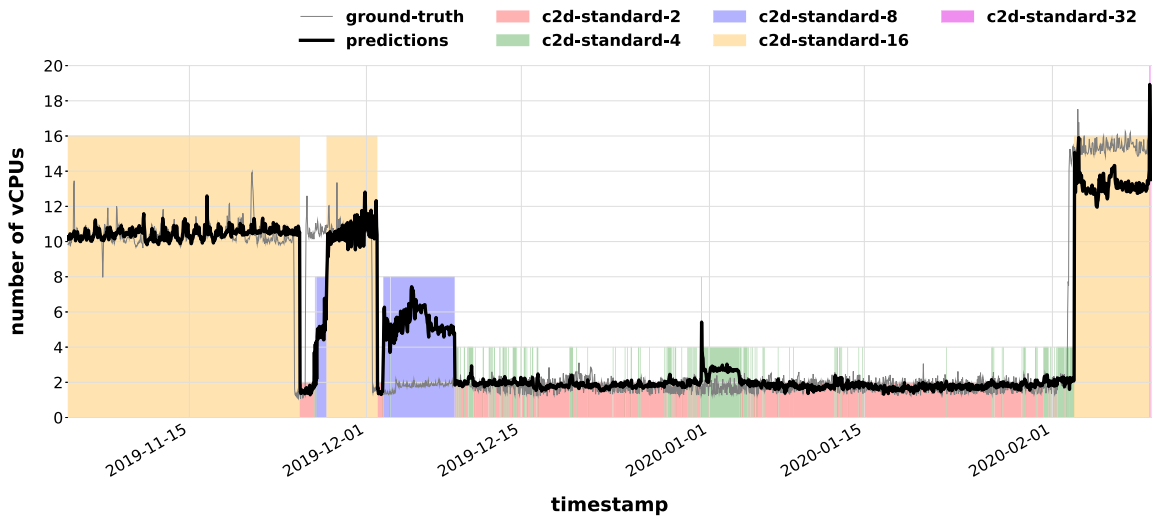


Fig. 9. Weekly CPU usage predictions obtained using *Method 8* (XGBoost-based method) and their mapping to virtual machine types for the reference machine set as *c2d-standard-112*.

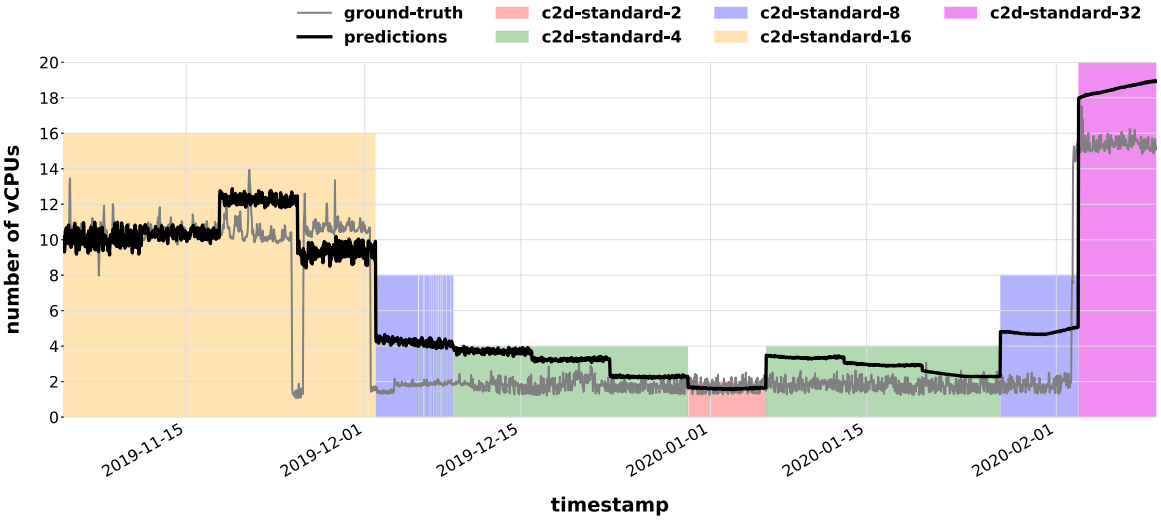


Fig. 10. Weekly CPU usage predictions obtained using *Method 21* (neural network-based method) and their mapping to virtual machine types for the reference machine set as *c2d-standard-112*.

Table 9
Reservation plan evaluation results: quantitative and qualitative analysis of different reference HPC-purpose machine types in relation to weekly CPU usage forecasts obtained using *Method 21* (neural network-based method).

Reference VM type	Total price (without *CRUOS*) in USD	Total price (with *CRUOS*) in USD	Avg. price per hour (without *CRUOS*) in USD	Avg. price per hour (with *CRUOS*) in USD	Relative gain	No. of re-scalings	No. of *SLA* violations
c2d-standard-2	275.128	275.128	0.117	0.117	0.0%	0	0
c2d-standard-4	550.255	275.128	0.234	0.117	50.0%	0	0
c2d-standard-8	1100.510	275.128	0.468	0.117	75.0%	0	0
c2d-standard-16	2201.020	294.780	0.936	0.125	86.607%	1	12
c2d-standard-32	4402.041	412.691	1.872	0.175	90.625%	2	14
c2d-standard-56	7703.571	686.181	3.275	0.292	91.093%	28	12
c2d-standard-112	15 407.143	1352.710	6.551	0.575	91.22%	30	43

resource wastage and the quality of service. Finally, Fig. 11 shows weekly CPU usage predictions obtained using *Method 25 — Quantile 0.50* (TFT-based method) and their mapping to virtual machine types for the reference machine set as c2d-standard-112. The significantly increased number of transitions between machine types was a result of the forecasts closely matching the ground truth.

The unambiguous results of the cost-effectiveness evaluation align with FinOps principles. Attempting to forecast resource utilization enables more informed decision-making, leading to improved cost efficiency and greater resource control. This is particularly vital in high-performance computing and advanced simulations, where predefined machines with high vCPUs entail significant costs. The evaluation of the *Reservation Module* presented relied solely on CPU forecasts concerning predefined machines. However, using custom machines in cloud environments, instead of predefined ones, could potentially offer a more flexible translation of forecasts into resource reservation plans. In the default scaling dependency plan, c2d-standard32 scales

Table 10
Reservation plan evaluation results: quantitative and qualitative analysis of different reference HPC-purpose machine types in relation to weekly CPU usage forecasts obtained using *Method 25 — Quantile 0.50* (TFT-based method).

Reference VM type	Total price (without *CRUOS*) in USD	Total price (with *CRUOS*) in USD	Avg. price per hour (without *CRUOS*) in USD	Avg. price per hour (with *CRUOS*) in USD	Relative gain	No. of re-scalings	No. of *SLA* violations
c2d-standard-2	275.128	275.128	0.117	0.117	0.0%	0	0
c2d-standard-4	550.255	275.128	0.234	0.117	50.0%	0	0
c2d-standard-8	1100.510	275.128	0.468	0.117	75.0%	0	0
c2d-standard-16	2201.020	294.780	0.936	0.125	86.607%	1	12
c2d-standard-32	4402.041	412.691	1.872	0.175	90.625%	2	14
c2d-standard-56	7703.571	569.907	3.275	0.242	92.602%	2	24
c2d-standard-112	15 407.143	1023.657	6.551	0.435	93.356%	82	199

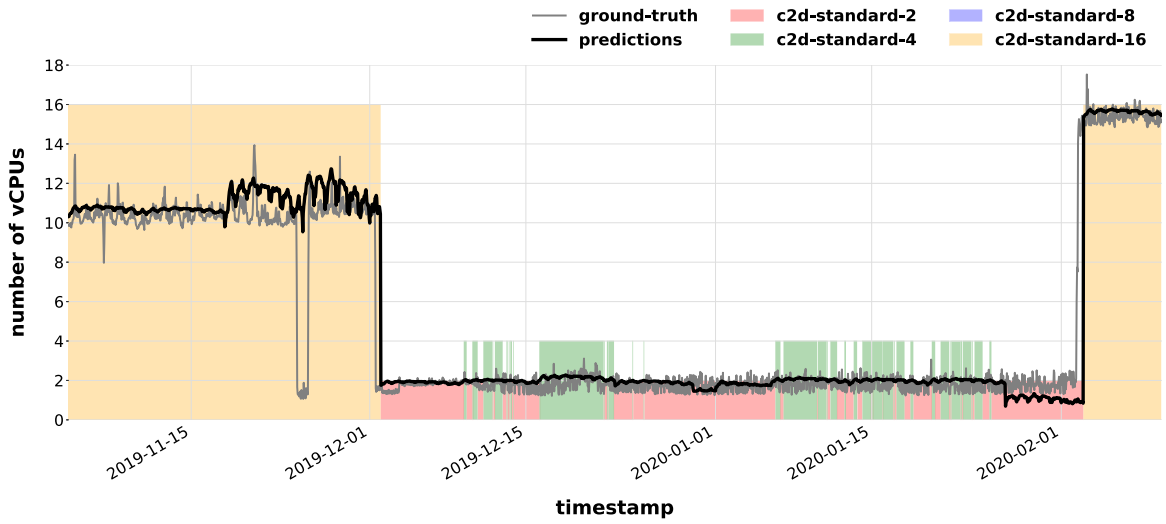


Fig. 11. Weekly CPU usage predictions obtained using *Method 25 — Quantile 0.50* (TFT-based method) and their mapping to virtual machine types for the reference machine set as *c2d-standard-112*.

up to c2d-standard-56, which might be overly robust. Consequently, custom machines can be more cost-effective and flexible when resource reservation plans are formulated on their basis as opposed to predefined plans. The financial gains presented are estimations; however, utilizing forecasts for resource reservation allows for substantial savings, even with small virtual machines. As concerns dynamic resource reservation plans, it is essential to anticipate potential inconveniences, such as SLA breaches and overly frequent rescalings. Therefore, focusing on a strategy where scaling occurs only when necessary is crucial. For instance, if the reservation plan suggests downscaling for just an hour, this might not be operationally feasible. This could necessitate an expansion of the *Reservation Module* to create both an initial reservation plan and an adjusted reservation plan. The second plan would involve the introduction of necessary adjustments to meet the defined objectives. One potential strategy could involve intentionally adding margins — if the prediction indicates a demand for 7 vCPUs for a specific period, allocating 8 vCPUs might lead to minimal wastage but reduce the risk of SLA violations. Moreover, the CPU usage percentage of such a machine would generally be lower, resulting in improved overall performance. In such cases, the utilization of custom machines appears to be a significantly better choice compared to predefined ones. As concerns the potential variance between long-term forecasts and ground truth, it becomes imperative to monitor the outcomes of the *Reservation Module*. Consequently, the *prediction schedule interval* configured in *CRUOS*, specifically if set to a daily interval rather than synchronized with the prediction horizon, facilitates prediction updates, thereby introducing an additional point for more frequent refinement of reservation plans. In the context of FinOps and operational strategies, it is crucial to consider the cost implications of *CRUOS*. Therefore, employing simpler prediction models can be especially significant, particularly when dealing with a larger number of machines for which resources are forecasted. Additionally, reserved resources can be included in declarations concerning future resource usage (in Google Cloud Platform nomenclature, this is known as resource-based committed use discounts), resulting in further discounts from providers.

5. Conclusions

Currently, cloud computing cannot fully replace supercomputers, especially for applications like long-running and advanced simulations using high-performance computing. However, it plays a complementary role, supporting computations in scenarios where local resources are constrained. While cloud computing offers an exceedingly flexible interface to seemingly limitless resources, acknowledging the risks linked to their utilization remains crucial. Consequently, FinOps merges financial expertise with operational proficiency in cloud management, enabling the implementation of resource-efficient strategies that address the risks associated with cloud resource utilization.

When it comes to HPC virtual machines, resource wastage correlates with notably higher costs. Despite this, given concerns about potential resource shortages during simulations, resources are often provisioned in a pessimistic manner, significantly exceeding actual demand. Therefore, in this work, we introduced the *CRUOS*, enabling the utilization of long-term predictions to dynamically create resource reservation plans. Long-term forecasting allows for more informed financial decisions and enables the utilization of resources based on actual needs. The crucial aspect involves utilizing forecasts for reservation plans, ensuring resources are available in advance when needed, which is not always guaranteed with on-demand provisioning.

The solution developed enables forecasting various resources across independent virtual machines. The research conducted focuses primarily on evaluating the system's CPU utilization results within a single virtual machine over a one-week prediction horizon. The *CRUOS* system involves eight modules, where each of the first seven is individually linked to a resource on a per-virtual machine basis. The final module is the *Reservation Module*, which serves as a central component in the FinOps-driven approach. Long-term forecasting enables more informed decisions to be made compared to short-term forecasting, especially in dynamic environments like cloud computing. As a result, the operation of the *CRUOS* system involves the outlier detection, feature engineering, model training and prediction stages, monitoring forecasts against actual ground-truth consumption. Thus, the evaluation presented in this study focused on both assessing prediction results and cost-effectiveness results.

During *CRUOS* evaluation, various prediction methods were employed, including statistical models, XGBoost, neural networks and the Temporal Fusion Transformer. These experiments were conducted using real-life data from a production system. Subsequently, these predictions were utilized to formulate resource reservation plans across different types of HPC machines. The outcomes of these evaluations illustrate the superiority of proactive resource usage prediction over reactive methods. Traditional approaches entail reserving resources at their maximum capacity, which often leads to consistent reservation at peak levels, even during periods of low demand. The implementation of dynamic reservations, utilizing predictive models, allows for adaptable resource scaling in response to fluctuating demand.

In the system developed, we observed the superiority of the prediction method based on the Temporal Fusion Transformer. The best-performing prediction method showed an approximately 31.4% improvement in RMSE compared to the top-performing baseline model. Besides quantitative assessment, the model demonstrated significantly better responses to erratic changes in data characteristics, indicating its higher reliability in predicting system behavior. The predictions generated by this system contribute to creating resource reservation plans, facilitating resource scaling. However, a crucial insight revealed that the best-performing prediction method in terms of error metrics might not always yield the best results for reservation plans. Hence, we introduced the FR metric in prediction evaluation. Consequently, despite the TFT-based method showing superior error metrics, both qualitative and quantitative assessments favor the neural network-based method for generating more efficient reservation plans.

Standard supervised learning assumes independent samples from an identical distribution. However, dynamics and data with characteristics of chaotic time series, especially from HPC systems, present challenges due to changing distributions, nonlinearities, and non-stationarity. Long-term predictions, a subject less explored than short-term ones, result in increased uncertainty with extended horizons, yet provide value by enabling better decision-making, cost management, and energy optimization according to the FinOps principles. While setting a universal rule is difficult, evaluating multiple methods incrementally helps draw robust conclusions. Consequently, when using the *CRUOS* system, a more effective approach might involve configuring the *prediction schedule interval* that is shorter than the prediction horizon, enabling additional adjustments to prediction outcomes. However, for research purposes, this interval was aligned with the prediction horizon. In terms of cost-effectiveness, plan evaluation on the basis of the number of rescaling events and SLA breaches was crucial. As indicated by the results obtained, estimated gains from using *CRUOS* depend primarily on the accuracy of forecasts and secondarily on the reference type of the virtual machine. Nevertheless, gains can be achieved, and irrespective of computing power or the resources available, it is worthwhile to adhere to FinOps principles from the start and employ certain cost management strategies in advance.

In further research, we aim to advance the FinOps-driven approach by focusing on forecasting multiple resources for multiple virtual machines, taking into account not just independent virtual machines but also potential relationships between them. Such studies would involve utilizing a dataset that covers the entire computing cluster. The *CRUOS* has been tailored for HPC machines, whereas in many cases, due to the costs involved, there are fewer HPC machines compared to non-HPC ones. However, future work will also cover various types of machines irrespective of their type or purpose. This necessitates careful consideration of system costs both during estimation and when formulating potential cost minimization strategies. Additionally, apart from evaluating diverse time series forecasting models affecting the *CRUOS* system cost, further development of the *Reservation Module* is envisaged. Given the outlined risks associated with translating predictions into reservations, additional strategies aiming at achieving a tradeoff between minimizing overprovisioning and SLA violations, while minimizing the number of rescaling events, can be developed. Incorporating other dependencies in multivariate forecasting, such as network traffic, disk usage, and other resource metrics may improve the results and contribute to further advancements in prediction model selection. Additionally, the application of custom virtual machine types appears particularly crucial, as they tend to be far more suitable than predefined ones when it comes to translating forecasts into reservation plans. Furthermore, as an important aspect of future research, we aim to thoroughly address the phenomena of both data drift and concept drift [30], which can significantly impact the performance of systems relying on multiple machine learning models. This can involve introducing deliberate and informed decision-making processes regarding the circumstances under which model retraining should be undertaken.

CRediT authorship contribution statement

Piotr Nawrocki: Writing – review & editing, Supervision, Methodology, Conceptualization. **Mateusz Smendowski:** Writing – original draft, Software, Methodology, Investigation.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

The data that support the findings of this study are available from ASEC S.A., 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 ASEC S.A.

Acknowledgment

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

References

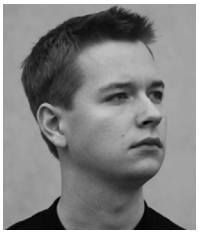
[1] A. Gupta, P. Faraboschi, F. Gioachin, L.V. Kale, R. Kaufmann, B.-S. Lee, V. March, D. Milojicic, C.H. Suen, Evaluating and improving the performance and scheduling of HPC applications in cloud, IEEE Trans. Cloud Comput. 4 (3) (2014) 307–321.

[2] C. Kotas, T. Naughton, N. Imam, A comparison of amazon web services and microsoft azure cloud platforms for high performance computing, in: 2018 IEEE International Conference on Consumer Electronics, ICCE, 2018, pp. 1–4, http://dx.doi.org/10.1109/ICCE.2018.8326349.

[3] P. Jarmatz, F. Maurer, H. Wittenberg, P. Neumann, MaMiCo: Non-local means and POD filtering with flexible data-flow for two-way coupled molecular-continuum HPC flow simulation, J. Comput. Sci. 61 (2022) 101617, http://dx.doi.org/10.1016/j.jocs.2022.101617, URL: https://www.sciencedirect.com/science/article/pii/S1877750322000424.

[4] P. Wittek, X. Rubio-Campillo, Scalable agent-based modelling with cloud HPC resources for social simulations, in: 4th IEEE International Conference on Cloud Computing Technology and Science Proceedings, IEEE, 2012, pp. 355–362.
[5] M.A. Netto, R.N. Calheiros, E.R. Rodrigues, R.L. Cunha, R. Buyya, HPC cloud for scientific and business applications: taxonomy, vision, and research challenges, ACM Comput. Surv. 51 (1) (2018) 1–29.
[6] J. Storment, M. Fuller, Cloud FinOps, O'Reilly Media, Inc., 2023.
[7] P. Nawrocki, M. Smendowski, Long-term prediction of cloud resource usage in high-performance computing, in: J. Mikyška, C. de Mulatier, M. Paszynski, V.V. Krzhizhanovskaya, J.J. Dongarra, P.M. Sloot (Eds.), Computational Science – ICCS 2023, Springer Nature Switzerland, Cham, 2023, pp. 532–546.
[8] S. Hazelhurst, Scientific computing using virtual high-performance computing: A case study using the amazon elastic computing cloud, SAICSIT '08, Association for Computing Machinery, New York, NY, USA, 2008, pp. 94–103, http://dx.doi.org/10.1145/1456659.1456671.
[9] K. Govindarajan, V.S. Kumar, T.S. Somasundaram, A distributed cloud resource management framework for High-Performance Computing (HPC) applications, in: 2016 Eighth International Conference on Advanced Computing (ICoAC), 2017, pp. 1–6, http://dx.doi.org/10.1109/ICoAC.2017.7951735.
[10] B. Posey, A. Deer, W. Gorman, V. July, N. Kanhere, D. Speck, B. Wilson, A. Apon, On-demand urgent high performance computing utilizing the google cloud platform, in: 2019 IEEE/ACM HPC for Urgent Decision Making (UrgentHPC), IEEE Computer Society, Los Alamitos, CA, USA, 2019, pp. 13–23, http://dx.doi.org/10.1109/UrgentHPC49580.2019.00008, URL: https://doi.ieeecomputersociety.org/10.1109/UrgentHPC49580.2019.00008.
[11] S.A. Mahmoudi, M.A. Belarbi, S. Mahmoudi, G. Belalem, P. Manneback, Multimedia processing using deep learning technologies, high-performance computing cloud resources, and Big Data volumes, Concurr. Comput.: Pract. Exper. 32 (17) (2020) e5699, http://dx.doi.org/10.1002/cpe.5699, arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1002/cpe.5699 URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/cpe.5699.
[12] R. Aljamal, A. El-Mousa, F. Jubair, A user perspective overview of the top infrastructure as a service and high performance computing cloud service providers, in: 2019 IEEE Jordan International Joint Conference on Electrical Engineering and Information Technology, JEEIT, 2019, pp. 244–249, http://dx.doi.org/10.1109/JEEIT.2019.8717453.
[13] F. Li, G. Wu, J. Lu, M. Jin, H. An, J. Lin, SmartCMP: A cloud cost optimization governance practice of smart cloud management platform, in: 2022 IEEE 7th International Conference on Smart Cloud (SmartCloud), 2022, pp. 171–176, http://dx.doi.org/10.1109/SmartCloud55982.2022.00034.
[14] P. Chung, AWS FinOps Simplified: Eliminate Cloud Waste Through Practical FinOps, Packt Publishing, 2022.
[15] M. Soni, FinOps Handbook for Microsoft Azure: Empowering Teams to Optimize Their Azure Cloud Spend with FinOps best practices, Packt Publishing, 2023.
[16] J. Mulder, Multi-Cloud Strategy for Cloud Architects: Learn How to Adopt and Manage Public Clouds by Leveraging BaseOps, FinOps, and DevSecOps, Packt Publishing, 2023.
[17] M.K. Abhishek, D. Rajeswara Rao, A scalable framework for high-performance computing with cloud, in: M. Tuba, S. Akashe, A. Joshi (Eds.), ICT Systems and Sustainability, Springer Nature Singapore, Singapore, 2022, pp. 225–236.
[18] P. Singh, P. Gupta, K. Jyoti, A. Nayyar, Research on auto-scaling of web applications in cloud: survey, trends and future directions, Scalable Comput.: Pract. Exp. 20 (2) (2019) 399–432.
[19] Y. Garí, D.A. Monge, E. Pacini, C. Mateos, C. García Garino, Reinforcement learning-based application autoscaling in the cloud: A survey, Eng. Appl. Artif. Intell. 102 (2021) 104288, http://dx.doi.org/10.1016/j.engappai.2021.104288, URL: https://www.sciencedirect.com/science/article/pii/S0952197621001354.
[20] E. Radhika, G. Sudha Sadasivam, A review on prediction based autoscaling techniques for heterogeneous applications in cloud environment, Mater. Today: Proc. 45 (2021) 2793–2800, http://dx.doi.org/10.1016/j.matpr.2020.11.789, URL: https://www.sciencedirect.com/science/article/pii/S2214785320394657, International Conference on Advances in Materials Research - 2019.
[21] S. Xue, C. Qu, X. Shi, C. Liao, S. Zhu, X. Tan, L. Ma, S. Wang, S. Wang, Y. Hu, L. Lei, Y. Zheng, J. Li, J. Zhang, A meta reinforcement learning approach for predictive autoscaling in the cloud, in: Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, KDD '22, Association for Computing Machinery, New York, NY, USA, 2022, pp. 4290–4299, http://dx.doi.org/10.1145/3534678.3539063.
[22] A.A. Rahmanian, M. Ghobaei-Arani, S. Tofighy, A learning automata-based ensemble resource usage prediction algorithm for cloud computing environment, Future Gener. Comput. Syst. 79 (2018) 54–71, http://dx.doi.org/10.1016/j.future.2017.09.049, URL: https://www.sciencedirect.com/science/article/pii/S0167739X17309378.
[23] X. Chen, H. Wang, Y. Ma, X. Zheng, L. Guo, Self-adaptive resource allocation for cloud-based software services based on iterative QoS prediction model, Future Gener. Comput. Syst. 105 (2020) 287–296, http://dx.doi.org/10.1016/j.future.2019.12.005, URL: https://www.sciencedirect.com/science/article/pii/S0167739X19302894.
[24] P. Nawrocki, M. Grzywacz, B. Sniezynski, Adaptive resource planning for cloud-based services using machine learning, J. Parallel Distrib. Comput. 152 (2021) 88–97, http://dx.doi.org/10.1016/j.jpdc.2021.02.018, URL: https://www.sciencedirect.com/science/article/pii/S0743731521000393.
[25] P. Nawrocki, W. Sus, Anomaly detection in the context of long-term cloud resource usage planning, Knowl. Inf. Syst. 64 (10) (2022) 2689–2711.
[26] P. Nawrocki, P. Osypanka, Cloud resource demand prediction using machine learning in the context of QoS parameters, J. Grid Comput. 19 (2) (2021) 20, http://dx.doi.org/10.1007/s10723-021-09561-3.
[27] P. Osypanka, P. Nawrocki, QoS-aware cloud resource prediction for computing services, IEEE Trans. Serv. Comput. 16 (2) (2023) 1346–1357, http://dx.doi.org/10.1109/TSC.2022.3164256.
[28] P. Nawrocki, P. Osypanka, B. Posluszny, Data-driven adaptive prediction of cloud resource usage, J. Grid Comput. 21 (1) (2023) 6.
[29] P. Osypanka, P. Nawrocki, Resource usage cost optimization in cloud computing using machine learning, IEEE Trans. Cloud Comput. 10 (3) (2022) 2079–2089, http://dx.doi.org/10.1109/TCC.2020.3015769.
[30] J. Lu, A. Liu, F. Dong, F. Gu, J. Gama, G. Zhang, Learning under concept drift: A review, IEEE Trans. Knowl. Data Eng. 31 (12) (2019) 2346–2363, http://dx.doi.org/10.1109/TKDE.2018.2876857.

Piotr Nawrocki is Professor in the Faculty of Computer Science at the AGH University of Krakow, Poland. His research interests include distributed systems, computer networks, mobile systems, machine learning, cloud computing and service-oriented architectures. He has participated in several EU research projects including MECCANO, 6WINIT, UniversAAL and national projects including IT-SOA and ISMOP. He is a member of the Polish Information Processing Society (PTI).



Mateusz Smendowski MSc is a PhD student at the AGH University of Krakow. His interests encompass machine learning, software engineering, and cloud computing, with a specific focus on cloud resource usage optimization and time series forecasting.