Knowledge and Information Systems (2022) 64:2689–2711
https://doi.org/10.1007/s10115-022-01721-5

REGULAR PAPER

Anomaly detection in the context of long-term cloud resource usage planning

Piotr Nawrocki[1] · **Wiktor Sus**[1]

Received: 12 May 2021 / Revised: 27 June 2022 / Accepted: 2 July 2022 /
Published online: 26 July 2022

Abstract
This paper describes a new approach to automatic long-term cloud resource usage planning with a novel hybrid anomaly detection mechanism. It analyzes existing anomaly detection solutions, possible improvements and the impact on the accuracy of resource usage planning. The proposed anomaly detection solution is an important part of the research, since it allows greater accuracy to be achieved in the long term. The proposed approach dynamically adjusts reservation plans in order to reduce the unnecessary load on resources and prevent the cloud from running out of them. The predictions are based on cloud analysis conducted using machine learning algorithms, which made it possible to reduce costs by about 50%. The solution was evaluated on real-life data from over 1700 virtual machines.

Keywords Anomaly detection · Cloud computing · Cloud resource prediction · Resource planning · Machine learning

1 Introduction

The increasing demand for cloud computing services is forcing providers to expand and optimize the resources available.

Cloud computing services are commonly used in multiple areas by many customers. Therefore, the cloud service providers hosting them are trying to optimize the costs of providing their services and ensure the efficient use of their resources. As a result, optimization mechanisms and automatic reservation systems are growing in importance [26]. This is especially important for public clouds where the gains are vastly more noticeable. Traditionally, the predictions were made using classic statistics. That required greater effort from the operator to properly set up, as well as insight into the analyzed system. Nowadays, that method has been replaced by machine learning (ML) algorithms which can work with minimal effort

✉ Piotr Nawrocki
piotr.nawrocki@agh.edu.pl

Wiktor Sus
wiktor.sus@gmail.com

[1] Institute of Computer Science, AGH University of Science and Technology, al. A. Mickiewicza 30, 30-059 Krakow, Poland

and find patterns in the data without any prior knowledge of the data. However, to correctly predict (and subsequently plan) resource usage, they require accurate data on usage patterns, and to obtain these data, proper anomaly detection must be employed. If this is not done, the data used will inevitably introduce noise in the form of random usage spikes, which leads to inaccurate predictions. Therefore, it is crucial to detect and remove this noise so that the data obtained can be later used for learning. This is especially important for long-term prediction where there may be more anomalies than for short-term prediction.

The presented solution can be used to predict changes in dynamic environments such as computing clouds, reducing resource usage through minimizing over-provision (in the process of planning resource reservation), which in turn allows more efficient usage of available resources and reduces operating costs. As needs are predicted ahead of time, it is possible to scale resources (which take time to prepare) so that they are available when needed.

This paper focuses on accurate long-term resource usage prediction (much less researched than short-term prediction), which is integral for any automatic planning tool. We define long term as days, weeks or months ahead as opposed to short-term prediction, which is made seconds, minutes or hours ahead. The problem of planning resource usage can be formulated as follows:

Theorem 1 *Let us assume a set of discrete resources (such as 2, 4, 6, 8 gigabytes of RAM) $X = \{x_0, x_1, x_2, \ldots, x_n\}$ that can be assigned to the virtual machine. The goal of the planning algorithm is to select such $x_i \in X$ that for given resource usage y the following condition will be fulfilled: $x_{i-1} < y < x_i$. The planning algorithm receives data from the anomaly detection algorithm in the form of time-series from which the anomalies have been removed.*

This makes the planning tool assign as little resources as is minimally required for the virtual machine to operate without problems, thus minimizing over-provision and, by extension, the cost of services.

For this purpose to be achieved, it is important to realize that in order to correctly predict resource requirements, many factors, both direct and indirect, need to be taken into account. These can be divided into direct and indirect factors. Direct factors include cloud topology, virtual machine hardware and software configuration and the physical hardware used. Indirect factors include anomalies related to time of the year and holidays. The approach adopted assumes that the analysis of virtual machine behavior makes it possible to accurately predict the requirements. The solution proposed allows cloud service providers to better utilize their system resources and results in better experience for the users, since adequate actions can be taken before anomalies even occur.

The analysis of current state of the art in the area of cloud resource usage planning revealed that the most prevalent methods are based on pre-defined rules and machine learning algorithms. In the first case, the rules are rigid and do not take into consideration dynamic changes in the environment as well as ignoring the aspect of prediction altogether. In the second case, ML begins to be often used to predict resource consumption; however, in the vast majority of cases, this is only short-term prediction that covers a period from a few to several dozen minutes. Also, none of the methods analyzed in Sect. 2 (with the exception of the authors' earlier works) in use leverage anomaly detection to improve overall results. At the same time, earlier works by the authors only use the basic anomaly detection mechanisms, without examining this topic thoroughly. Therefore, in this paper we propose a novel hybrid anomaly detection method called the Weighted Hybrid Algorithm (WHA), which employs a combination of anomaly detection mechanisms and significantly improves the results of long-term prediction. Due to this approach, it is possible to use multiple configurations of

individual anomaly detection algorithms suited for different scenarios, which improves the overall accuracy.

Summarizing the above, the main contributions of this paper can be defined as follows:

- The development of a machine learning-based long-term prediction method that can adapt to changes in the analyzed system and optimize long-term cloud resource usage planning;
- The development of a novel hybrid anomaly detection algorithm;
- The analysis of the impact of anomaly detection mechanisms on the accuracy of long-term usage prediction;
- The analysis of the methods developed and their usefulness in real-life scenarios.

The rest of this paper is structured as follows: Sect. 2 contains a description of related work, Sect. 3 contains a description of selected anomaly detection algorithms useful to our use case and our approach, Sect. 4 is concerned with long-term cloud resource usage planning using anomaly detection and machine learning, Sect. 5 describes the evaluation of the prediction solution using anomaly detection, and Sect. 6 contains the conclusion and further work.

2 Related work

This section has been structured as follows: the first subsection presents the current state of the art in the area of anomaly detection, the second subsection presents the state of the art of the cloud resource usage planning methods, the third subsection presents methods that utilize anomaly detection methods in prediction, and the fourth subsection summarizes the presented related work.

2.1 Anomaly detection

The issue of anomaly detection has been frequently analyzed by researchers. In [1], the authors demonstrate a rule-based method for detecting anomalous states in cloud virtual machines. The method detects and, if possible, attempts to mitigate abnormal behavior. The authors tested their solution with the Java Virtual Machine and observed a decrease in Out of Memory Errors as well as memory swaps while increasing the overall density of machines that could run on a host. While it is easy to configure, this method suffers from an inability to respond to dynamic changes in live systems, which are prone to fluctuate over time, especially over the course of weeks and years. Our solution can adapt to dynamic changes in the system and properly respond to them as opposed to merely conducting a static analysis.

In [13], the authors approach the problem from another perspective—by monitoring response time and the users' perception of the service. The tests demonstrated that the algorithm indeed recognized anomalous states; however, it was unable to pinpoint the cause and therefore will not be further analyzed. A similar method based on outlier detection and statistical analysis was employed in [28] and in [14], but it did not employ any mechanism allowing it to adapt to changes in usage over time. Our solution works directly on the data from the system, and thus, it is able to discover which resource is experiencing anomalous usage, which is not possible with the solution presented.

In [10], the authors describe a method similar to rule-based approaches. It is extended using virtual machine analysis mechanisms to allow greater flexibility and adapt to systems with different usage characteristics. The implementation was tested against a standard rule-based solution. Authors observed an improvement in accuracy of over 50% as well as a

reduced false positive count. The method could be further improved by modeling recurrent anomalies based on machine usage characteristics over time. A similar approach is described in [23]. Our solution includes mechanisms to detect recurrent anomalies and use them in predictions, which is not possible with the solution presented.

A similar method was proposed in [3]. The algorithm proposed also analyzes the host to properly model its usage patterns; however, the calculation method is different. The method creates a fingerprint for the user, which is calculated from the tensors generated on the basis of various metrics deployed. The method allows greater flexibility by analyzing the user's behavior. Test results showed that the algorithm accurately modeled and responded to users whose signature was stable over time. It was unable to respond to purely random behavior, however. The authors also noted difficulties in collecting sufficient amounts of user data in real-world scenarios. Our solution analyzes a particular system rather than specific users, without categorizing them into specific behavior types, which makes it easier to collect the data, as opposed to the solution presented.

In [19], the authors extensively analyzed the anomaly detection problem. The approach they propose is based on dynamic limits, which are calculated with the Kalman Filter and clustering methods. The resulting mechanism enables great detection accuracy as well as adaptation to system changes and the proper modeling of recurrent patterns, although the algorithm can only detect individual cycles within the data collected. Our solution can detect multiple overlapping cycles and act accordingly as opposed to the solution presented, which is limited to a single cycle.

Another method described in [12] approaches the problem from the host machine's perspective. By analyzing the workloads of various hosts, it detects over- and under-utilized hosts and tries to migrate virtual machines to keep balance. The algorithm helps to reduce the total electricity usage of a data center as well as upholding the service level agreement. However, due to the fact that it analyzes the system as a whole, it cannot detect anomalies in virtual machines themselves. Test results showed a 93% decrease in over- and under-utilized hosts as well as a drop of 5% in overall energy consumption. On the other hand, the analysis of individual virtual machines did not yield significant changes due to the algorithm's inability to properly balance host machines. This method does not detect anomalies in individual machines, which is necessary for our use case, and therefore will not be further discussed. Our solution works on specific instances instead of hosts.

The most accurate and most flexible group of anomaly detection methods utilize machine learning techniques. One such method presented in [6] leverages an auto-encoder deep neural network. The proposed method was first trained to recognize the normal state on data obtained from a supercomputer in an unsupervised manner and then tested on data containing anomalies. The authors have carried out multiple experiments, achieving 88%-96% accuracy in detecting anomalies. The strength of their method is the ability to work with relatively small amounts of training data, which do not require labeling. Our WHA solution similarly does not require labeled data to determine the normal state of the machine.

In [5], the authors propose an anomaly detection approach that is also based on deep learning. The method utilizes generative adversarial networks, which are used to detect anomalies by analyzing multivariate time series in an unsupervised manner. The networks learn the normal state for a given system and are able to detect anomalies that differ from this state. Our solution is also able to work in an unsupervised manner, although it does not use adversarial networks.

In [15], the authors propose a prediction-based anomaly detection method, which utilizes a convolutional LSTM neural network to detect anomalies in video segments by analyzing them frame by frame. Their solution utilizes a set of predictors to determine what the frame

Table 1 Anomaly detection methods

Article	Method type	Year published
[1]	Rule-based	2015
[13]	Perception-based	2015
[28]	Statistics-based	2013
[14]	Statistics-based	2019
[10]	Adaptive	2010
[23]	Adaptive	2018
[3]	Behavior-based	2009
[19]	Adaptive	2018
[12]	Usage-based	2020
[6]	Machine learning-based	2019
[5]	Machine learning-based	2020
[15]	Machine learning-based	2019
[24]	Machine learning-based	2018
[8]	Machine learning-based	2018

following the currently examined one should look like and then compare the actual next frame with the predicted one to decide whether that frame is anomalous. Our solution compares current data with the normal state instead of predicting what the value should be.

Another method was presented in [24], where the authors propose a machine learning algorithm which analyzes every virtual machine instance separately based on its previous usage data. This allows for greater accuracy in pinpointing possible anomalies, although it disregards the system's behavior as a whole. Presently, in most cases only basic solutions are employed, which do not have adaptive capabilities. Our solution analyzes the data supplied by individual resource measurements, which allows it to correlate anomalies with a certain resource type, as opposed to the approach presented.

In [8], the authors extensively analyze the anomaly detection problem. They propose a machine learning-based solution that can dynamically adapt to an unknown dataset and find anomalies within it. It does not require any prior knowledge or labeling and does not use any hard thresholds for detection, but instead uses a neural network to decide whether the data contain anomalies or not. The authors noted the high accuracy of their solution during evaluation against the anomaly detection framework called EGADS[1] on Yahoo Webscope Benchmark. Our solution analyzes the data using multiple provided algorithms and does not include any machine learning-based detectors.

The solutions presented above are summarized in Table 1. The anomaly detection problem has many possible solutions. However, there is room for improvement in the area of adaptive mechanisms that are specifically tailored for use in usage prediction. Our solution (WHA) is designed to increase the accuracy of long-term predictions through anomaly detection. Moreover, it includes mechanisms for modeling recurrent anomalies and handling them properly, which is not provided for in any existing methods.

[1] EGADS (Extensible Generic Anomaly Detection System)—https://github.com/yahoo/egads.

2.2 Cloud resource usage planning

The methods most commonly used in cloud resource usage planning are rule-based and ML ones. However, systems using predefined rules [11, 17] do not take into consideration dynamic changes in the environment as well as ignoring the aspect of prediction altogether. That is why, they are becoming less and less used. Currently, prediction methods based on machine learning algorithms (that allow for dynamic resource usage planning) are gaining more and more popularity due to their relatively high accuracy, flexibility and ease of use after being set up. Therefore, in our analysis we will focus on solutions using machine learning algorithms. In this section, we analyze cloud resource usage planning solutions that do not involve detecting anomalies.

In [22], the authors describe an algorithm based on learning automata, which accurately predicts the virtual machines' loads. The algorithm assembles together multiple machine learning algorithms for better prediction. The solution was tested on data collected on more than 1,000 virtual machines from almost 500 host machines. The data were sampled with a 5-minute interval for 24 hours from machines with a high variety of use cases. The solution performed best in comparison with four different prediction algorithms tested.

In [18], the authors present an ensemble model for workload prediction and comparison with the baseline. The use of machine learning algorithms (KNN and decision tree) in the solution developed allowed for good accuracy and the ensemble achieved through these has also shown an improvement in results. The baseline study showed an RMSE (Root Mean Square Error) of 0.37 and the mechanism proposed has demonstrated a 6.22% reduction in the CPU usage error and a 17.3% reduction in the memory usage error. The experiments conducted were carried out for predictions amounting to a few minutes ahead.

An approach for predicting cloud resource utilization on a per-task and per-resource level is presented in [7]. The authors apply machine learning-based prediction models and they experimentally show that these reduce (in the context of one hour) the prediction error by 20% in a typical case, and best-case scenario improvement exceeds 89%.

In [16], the authors explore the efficiency of neural networks to predict multi-resource utilization. They propose a model which uses a hybrid Genetic Algorithm (GA) and Particle Swarm Optimization (PSO) to train network weights and uses a Functional Link Neural Network (FLNN) for prediction (a few minutes ahead). The results of experiments show that the hybrid model proposed yields better accuracy compared to traditional techniques.

A deep learning-based workload prediction model for resource provisioning in an autonomic cloud computing environment was presented in [2]. The authors focused on forecasting CPU usage for virtual machines in cloud data centers and proposed an efficient deep learning model based on diffusion convolutional recurrent neural networks (DCRNN). An essential aspect of the work was the possibility to capture important topological properties of the network that significantly and positively influence forecasting accuracy. Also, various experiments (for time horizons of 5, 15, 30 and 60 minutes) were conducted to validate the effectiveness of the proposed model. Experimental results showed that the proposed DCRNN-based forecasting model was useful for resource provisioning in the autonomic cloud computing platform.

[4] is one of the few works that deal with predicting cloud resources (using machine learning) for longer than a day. The authors of the article present theoretical considerations related to predictions in terms of days and months, but experiments confirming the operation of the model proposed are performed only for 30 days. The model proposed by the authors does not involve anomaly detection, which may significantly affect the accuracy of long-term prediction using machine learning algorithms.

Table 2 Usage prediction methods

Article	Method type	Prediction	Year published
[11]	Rule-based	–	2013
[17]	Rule-based	–	2017
[22]	Machine learning-based	Short-term	2018
[18]	Machine learning-based	Short-term	2018
[7]	Machine learning-based	Short-term	2016
[16]	Machine learning-based	Short-term	2022
[2]	Machine learning-based	Short-term	2021
[4]	Machine learning-based	Long-term	2021

The analysis of the existing prediction solutions (summarized in Table 2) shows that rule-based prediction methods were used quite frequently in the past. However, machine learning algorithms are used to predict resource consumption nowadays. The analysis of the latest publications on prediction has shown that a significant majority of solutions concern short-term prediction. Therefore, in our research, we dealt with long-term prediction, which has not been widely studied so far, and which seems to be more and more important in the context of the increasing use of cloud resources, including by mobile network operators. Also very importantly, there is scarcely any research that uses anomaly detection methods in the process of predicting and then planning resource consumption. This may be especially important for predicting resource consumption using ML, where the noise introduced into the data often decreases the accuracy of the prediction. Our solution avoids this by using an anomaly detection mechanism as a filter. It also enables responding to specific user needs by modeling the users' behavior.

2.3 Usage prediction with anomaly detection

Very few works incorporate anomaly detection mechanisms into resource usage prediction. An example is [25], in which the authors propose a novel method used to build a long-term virtual machine reservation plan in cloud computing using machine learning algorithms. The solution suggested enables autonomous plan adaptation and verification through analyzing data on system usage. It also accounts for abnormal usage and may be dynamically updated where problems are predicted. Our solution focuses more on the anomaly detection aspect as opposed to the solution presented.

In [27], the author analyzes prediction possibilities in periods ranging from a few minutes to several months; however, he focuses on weekly cycles. In the first stage of the research, the author used simulation to provide a proof of concept. In the second stage, the time series analysis was extended from short-term to long-term resource utilization prediction using trace data. At the same time, the author only uses a very basic anomaly detection mechanism [9], leaving this issue for future research.

In [21], the authors describe a system for automatic resource scaling in a cloud environment. The solution proposed analyzes the system over a specified period and selects the appropriate usage plan from available options. The solution collects data from specified sources, such as the CPU, memory or network, which are then analyzed for anomalies. The anomalies detected are disregarded to improve prediction accuracy. The remaining data are processed further—smoothed using a median filter to further increase accuracy by removing

Table 3 Prediction usage solutions with anomaly detection mechanism

Article	Method type	Prediction	Year published
[27]	Machine learning-based	Short/long-term	2013
[25]	Machine learning-based	Long-term	2019
[21]	Machine learning-based	Long-term	2020
[20]	Machine learning-based	Long-term	2021

momentary changes which could affect the prediction. Predictions are made on the basis of an analysis covering a specified time period. As a result of these observations, usage characteristics are discovered. The solution is capable of choosing from different scaling options depending on the values predicted and the cost of individual plans. Our approach is similar in terms of data preprocessing; however, our anomaly detection mechanism does not filter out anomalies which occur with a certain constant frequency, as this indicates that these changes are in fact not anomalous and should be taken into account when predicting future usage.

The authors of [20] present an approach that uses anomaly detection and machine learning to achieve cost-optimized and QoS-constrained cloud resource configuration. The solution developed was tested using a system located in Microsoft's Azure cloud environment. Experiment results show a cost reduction for PaaS/IaaS over the tested period. However, in their research the authors focus on maintaining QoS parameters in the context of predicting resource consumption and use only basic, simple anomaly detection mechanisms without analyzing this topic in more detail.

The analysis of existing prediction solutions with anomaly detection (summarized in Table 3) shows that there still has been little research on this important topic. All analyzed articles are primarily focused on the long-term prediction of resource consumption, for which the detection of anomalies is a significant problem.

2.4 Summary

The solutions described above have addressed the issue of detecting the machines' anomalous states. However, they are most often limited to analyzing present anomalies and do not try to incorporate them in their prediction methods. Moreover, most cloud resource usage prediction and planning methods do not model the user's behavior in the long term where such factors as holidays and weekdays/weekends begin to matter. It should be noted that even if the existing research on long-term prediction includes the detection of anomalies, it is carried out using simple methods without more in-depth research in this area. The contribution of this study is to present a solution which incorporates the cloud resource usage prediction/planning and complex anomaly detection aspects and test it on real-life data as well as discussing possibilities of long-term usage prediction.

3 Anomaly detection in the context of cloud usage

In our work, we focus on two distinct types of anomalies: Point Anomalies (PA) and Recurrent Anomalies (RA). Point anomalies can also be thought of as true anomalies, since they occur completely randomly or in a manner we cannot predict (without any emerging pattern). They have the form of sudden spikes in resource usage, which shortly disappear. However, due to the dynamic nature of the cloud, it is possible for a system to encounter a change in usage

pattern. If such a change remains in place for a longer period, it should not be considered anomalous. The term "recurrent anomaly" was adopted to describe an environment in which it is possible to observe usage changes that would be considered anomalous by themselves, but when combined, they begin to form a pattern, which makes it possible to predict the next occurrence. This type of anomaly is usually strongly tied to usage patterns, which in turn may be related to such factors as the time of day, the day of the week, the month of the year, etc. By analyzing previous usage history, it may be possible to properly model these kinds of anomalies. This difference is important in the context of resource prediction, which is discussed in detail in Sect. 4.

Two major types of anomaly detectors are used—static and adaptive ones, the latter also encompassing methods based on machine learning.

The static approach, as the name suggests, does not account for changes over time. It is usually implemented as a set of rules which determine the state of the system and decide what action to take in the event of an anomaly. One such example of a static approach is a Limit-based algorithm (LA), which can rely on soft or hard limits in attempt to minimize false positives (an example is presented in [1]). When the values measured exceed the limits, the system triggers an alarm. The main reason for the popularity of this approach is the ease of configuration and deployment; in many cases, it may prove sufficient, especially in systems which are not subject to drastic changes. However, it also has many downsides such as low accuracy in dynamic environments due to its inability to respond to changes in usage patterns and poor scaling with larger systems, since administrators may need to configure limits on a per-instance basis. This approach is one of the most commonly used throughout the industry due to the simplicity of its deployment and configuration. However, managing it accurately or for individual instances quickly becomes infeasible.

Solutions with the highest accuracy utilize some form of dynamic profiling: either of an individual virtual machine or of the entire system. The main idea behind anomaly detection is finding data that do not match the expected patterns. Therefore, an intuitive idea is to determine the pattern during normal usage and compare it against any new data in order to find outliers. In the literature, this is called the "normal state." Many approaches to calculating the normal state are discussed in literature. The simplest among these are Dynamic Limits-based algorithms (DLAs), which calculate the normal state dynamically, and subsequently use the result to derive limit thresholds, similarly to the static approach. An example of a DLA approach is to use the average of last observed samples, called the Simple Moving Average (SMA):

$$SMA = \frac{p_{n-k+1} + p_{n-k+2} + \cdots + p_n}{k} = \frac{1}{k} \sum_{n-k+1}^{n} p_i \tag{1}$$

where p denotes the sample, n the total number of samples, and k the window size. By only calculating the result using a certain number k of last states p instead of all the samples n, the algorithm becomes adaptive, since if the usage pattern changes over time, this will be reflected by the new normal state. It is also possible to calculate the median in a similar way, which results in values being used that are closer to actual usage. This Median Algorithm method is later used for evaluation. As with the static approach, a state is considered abnormal when the usage values measured differ significantly from those expected. A more advanced solution would incorporate data gathered from multiple metrics.

An example of such approach is [3] which uses the data to generate a tensor, which is subsequently used to create a fingerprint characterizing the user's behavior patterns. Such approach can model predictable users better, but it requires a larger number of samples and

does not lend itself well to random patterns. After initial training, it is easy to deploy and manage, since manual configuration on a per-instance basis is not required due to its dynamic nature. Another very good method of obtaining the normal state is Kalman filtering, and this is extensively analyzed in [19]. The Kalman filter is a statistical method of predicting the value of an unknown variable based on previous measurements. The method assumes that all measurements are prone to error and aims to minimize this error. Its strength lies in its ability to correct itself dynamically when new measurements are made, which increases accuracy over time and adapts to changes in usage patterns. Similarly, it is very easy to deploy and does not require large data sets to start either, which makes it a good choice for a highly accurate monitoring solution. What makes it most appealing in resource planning is its ability to model events that occur regularly, which makes it possible to predict recurrent anomalies with ease. However, this only applies to individual recurrent anomalies, and thus, accuracy will decrease where multiple overlapping frequencies are present.

Finally, the solution presented in [8], called DeepAD, achieves great accuracy in detecting anomalies through the use of machine learning algorithms. It follows the dynamic pattern by analyzing the data and not setting any hard thresholds.

Our approach to detecting anomalies (WHA) falls into the dynamic limits category and operates similarly to the DeepAD solution in terms of architecture; however, it does not use machine learning algorithms and employs an additional weighting mechanism to control the influence of each algorithm's output. It uses multiple dynamic solutions such as the Moving Median, the Kalman filter and the Savitzky-Golay filter, which allows it to achieve higher accuracy. It also uses an internal mechanism for detecting recurrent anomalies. The approach is designed to work with longer time spans and to detect many distinct recurrent anomalies, thereby circumventing the Kalman filter's limitation to a single recurrent cycle. This makes it suitable for use cases in which usage patterns are predictable or based on periodic events related to real-world functioning. The algorithm calculates the confidence level based on the sub-algorithm's outputs and the weights supplied, which denote confidence in the accuracy of child algorithms. The full equation used to obtain confidence levels is as follows:

$$\mathrm{WHA}(x) = \frac{F_1(x) * W_1 + F_2(x) * W_2 + \cdots + F_n(x) * W_n}{n} = \frac{1}{n} \sum_{1}^{n} F_i(x) * W_i \qquad (2)$$

where filters are denoted as F, weights as W, the sample as x and the total number of filters used as n. The algorithm acts as a noise filtering module for the resource usage prediction mechanism to improve the accuracy of the system. This solution allows accurate load prediction and by extension reduces the need to over-provision resources and the cost of virtual machine upkeep.

Algorithm 1: Recurrent anomaly detection algorithm

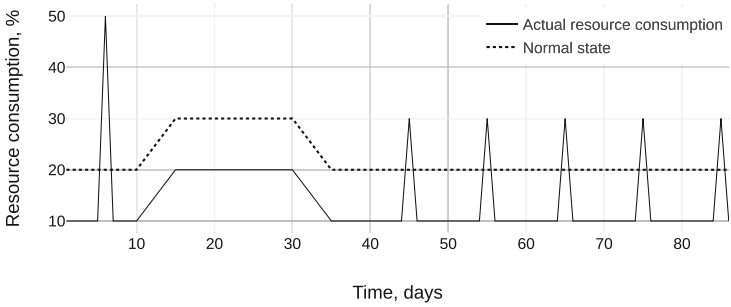
```
1  while True do
2      data = read_csv('vm_resource_usage.csv');
3      update_state(data);
4      calculate_normal_state_wha();
5      if data within Normal State limits then
6          continue;
7      else
8          if similar anomaly has occurred earlier then
9              register_recurrent_anomaly(data);
10         else
11             register_point_anomaly(data);
12         end
13     end
14 end
```

The detection mechanism is presented in Algorithm 1 and visualized in Fig. 1. It is deployed in the form of multiple analyzers set up for each measured resource. These calculate the "normal" state for given machine, which describes typical usage values for it. First, the data are loaded from the csv file generated by a periodic job running on the VM, which reads resource usage and appends it to the file with a timestamp. The normal state is dynamically updated based on new values (lines 3–4, Fig. 1a), which allows for changes in the system's usage characteristic. All the values contribute to the overall score with weights based on the date of measurement. Older values have smaller impact on the final score. Subsequently, the measurements are checked against the normal state (line 5), and when they deviate by more than the allowed tolerance (provided as an additional parameter), an anomaly is registered (line 11, Fig. 1b).

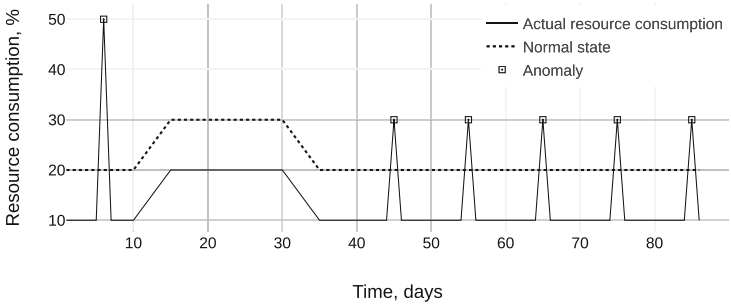
Whether the anomaly is recurrent is determined depending on whether it occurs at certain intervals. On every occurrence of the anomaly, the system registers the date and time of the event. A single occurrence is not a basis for assuming that the anomaly is recurrent, but every consecutive occurrence within a specific interval will increase the confidence considerably. If the anomaly does not occur within that interval, the system's confidence will drop. After multiple occurrences (at least two), the system will not treat new similar anomalies as one-off events and will instead pass them on to the prediction module so that they can be modeled at the planning stage (line 9, Fig. 1c). It is worth noting that the following can also be achieved with an FFT-based algorithm, but for our use-case we have decided to go with the custom detector described before due to its simplicity and because our main objective was to determine whether including recurrent cycles has an impact on overall accuracy.

All of the methods described are compared in Table 4. As a result of analyzing our solution's needs, we have selected two approaches to compare with our method. These approaches are based on static limits because this is the most frequently used method and therefore serves as a useful baseline, and on dynamic limits because this method ensures flexibility (minus recurrent anomaly detection).

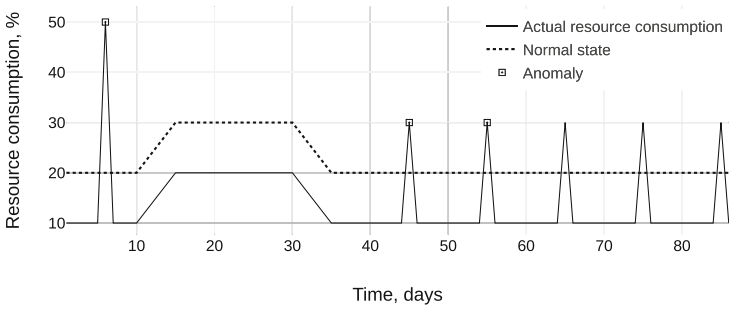
The selected anomaly detection methods have been implemented in our system, which is described in detail in Sect. 4. In Sect. 5, the solution's performance is measured and compared against the selected methods on data collected from live systems.



(a) Step 1 — dynamic calculation of the normal state



(b) Step 2 — detecting point anomalies



(c) Step 3 — including recurrent anomalies in the normal state

Fig. 1 Visualization of steps taken by the anomaly detection algorithm

4 Cloud resource usage planning with anomaly detection and machine learning

The resource usage planner analyzes virtual machines through the data gathered from them, which may consist of CPU, memory, network or disk usage. Based on historical data, it

Table 4 Comparison of selected anomaly detection algorithms

Algorithm	Cloud implementation	Configuration	Accuracy	Scalability	Type
Limit-based [1]	Easy	Easy	Low	None	PA
Statistical [28]	Easy	Easy	Low	None	PA
Dynamic limits [10]	Easy	Easy	Medium	High	PA
Fingerprinting [3]	Medium	Medium	Medium	High	PA
Kalman filter [19]	Easy	Easy	High	High	PA&RA
Host load [12]	Hard	Hard	Medium	Medium	PA
ML-based [21] [24]	Easy	Medium	High	High	PA
Weighted hybrid	Easy	Easy	High	High	PA&RA

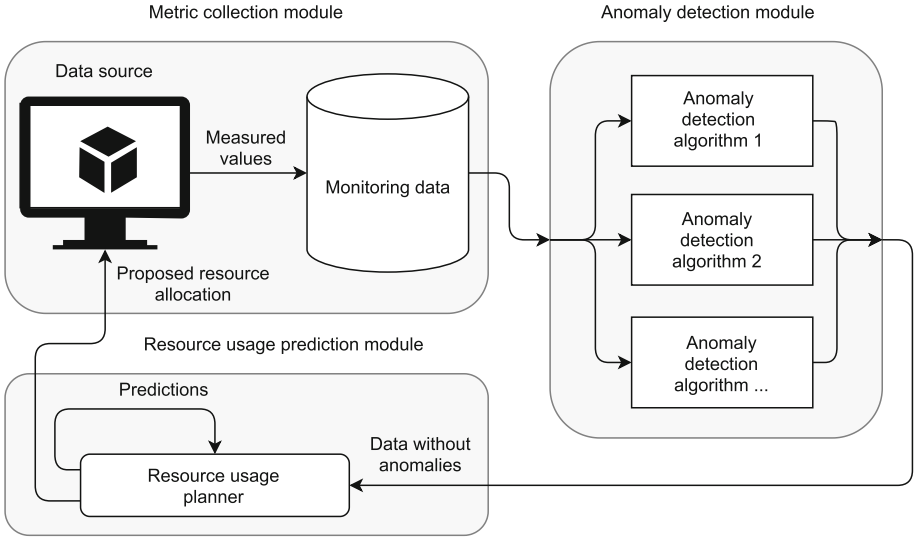


Fig. 2 Prediction system architecture

generates predictions of future demand for virtual machines. Such predictions can then be used to adjust actual capacities of the machines managed. The prediction horizon can be adjusted through configuration, although longer-term forecasts are less reliable.

The overall system architecture (Fig. 2) consists of three main modules —the *Metric collection module*, *Anomaly detection module* and *Resource usage prediction module*. Together they form the entire planning system, which can be deployed to individual instances.

The *Metric collection module* is the source of data gathered from the virtual machine's measurement module. These data can be stored in a central database or processed directly if the planner is deployed to a virtual machine. The collected data may include any measurable system property that changes over time, such as CPU, memory, network or disk usage. Subsequently, the data containing the measured values over time together with the timestamps are sent to the *Anomaly detection module*.

The *Anomaly detection module* uses one or more detection algorithms and can be expanded with additional ones. Each of these algorithms is individually provided with collected data to analyze, which allows the algorithm to determine whether the virtual machine is operating

Algorithm 2: Prediction algorithm with Anomaly Detection as a filter

```
1  const predict_next_days = Integer();
2  LSTM_network = Pretrained_LSTM();
3  anomaly_filter = ADFilter();
4  while True do
5      data = read_csv('vm_resource_usage.csv');
6      anomaly_filter.update(data);
7      filtered_data = anomaly_filter.get_filtered_data();
8      if filtered_data != null then
9          prediction = LSTM_network.update(filtered_data);
10         LSTM_copy = copy(LSTM_network);
11         predicted_usage = Array();
12         yield predicted_usage;
13     end
14 end
```

within normal limits. During the testing, we use our WHA algorithm. Additionally, each method outputs a confidence metric, which allows the detector module to determine the most accurate assessment of the machine's state. The accuracy and effectiveness of individual methods are evaluated and tested in Sect. 5. The module outputs to the *Resource prediction module* data that are not anomalous. The goal of removing the data is to improve accuracy by assuming that anomalies are noise and should not be included in the predictions. Similar to the *Metric collection module*, this module holds usage measurements for individual resources together with timestamps.

The *Resource usage prediction module* is based on a Long-Short Term Memory (LSTM) recurrent neural network. This type of neural network was chosen due to its ability to accurately model changes in a system in which the current state is dependent on the previous one. The network is constantly updated with values obtained from the *Anomaly detection module* and outputs the prediction for the following day. By feeding it with its own predictions, it is possible to predict future usage, although this leads to noise accumulation, so the further in future the prediction, the less accurate it is going to be. This module outputs the data containing predicted values for the data source, which can be used to change the amount of resources allocated.

The full algorithm of the planning system is presented as Algorithm 2. Lines 5–7 are an example how the metric and anomaly detection modules are used, and lines 8–17 show network updating and prediction generation. Since updating the network changes its internal state, it is necessary to use a copy to retain the current state of the network before the prediction is generated. The predictor can utilize a separate network for each available metric or a single network that analyzes the combined data from the virtual machine.

The long-term accuracy of the system is primarily dependent on the data it is provided with. However, real-life systems do not always behave as expected, introducing anomalies to the system. Making predictions with such data results in increased noise and negatively impacts prediction quality because the system will try to model both normal machine behavior and abnormalities. To mitigate that, the anomaly detection module is used. Its role is to filter out any erroneous data, leaving only accurate values, which are later used by the resource planner. It does not filter out recurrent anomalies, however, since these should be included in predictions. The anomaly detection module can be selected from multiple available algorithm implementations, with the possibility to add more.

Measuring the accuracy of the prediction module alone is typical of neural networks—this makes it possible to check how well the network can model actual usage using unknown data. However, this is impossible when anomalies are ignored, since it is no longer desirable to model the usage in an exact manner. Therefore, as presented in Theorem 1, the overall performance metric for the problem consists of two components—the underestimation metric combined with the accuracy metric. The goal is to minimize underestimates, since in cloud computing it is important to uphold the service level agreement, while minimizing over-provision at the same time. Individually, neither of these metrics fit the criteria, since optimizing them leads to incorrect planning. Due to the nature of anomalous behavior, measuring the mean error itself is not sufficient, since large usage spikes may severely bias the result, even if they only last for a moment. To accommodate this, the final metric is based on negative surface area difference and cross-correlation. For additional analysis, the other possible metrics such as overestimation are also collected.

5 Evaluation

Evaluation is divided into multiple segments. First, we evaluate the prediction without the anomaly detection mechanism to establish a baseline and present its impact on the accuracy. Next, we evaluate prediction with different anomaly detection methods employed to compare prediction accuracy when using our anomaly detection algorithm, presenting its strengths against existing solutions.

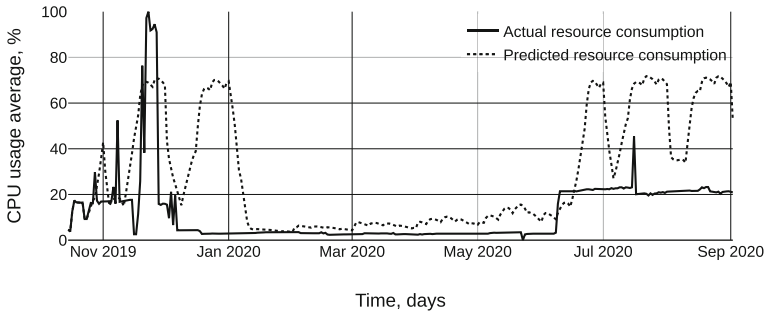
The system was implemented in the Python3 language with the Keras framework used for machine learning and the resulting code has been published to GitHub.[2] The testing dataset consisted of historical usage metrics collected from over 1700 virtual machines from the Nokia Solutions Krakow cluster[3] and over 100 virtual machines from the Polcom cluster.[4] The virtual machines were grouped by deployment project, since their use cases are similar and this allows easier analysis. The data are available in multiple lengths and resolutions, from a week to a year, with less frequent sampling in the longer sets. History over the entire year is sampled every day, while data from a single month are sampled every hour. The metrics available contain information about CPU usage in the form of percentages and frequency in MHz, memory usage in bytes, swap usage, uptime, etc. Due to legal restrictions, we cannot share the main datasets used in our experiments, which means that they cannot be reproduced. However, to demonstrate that our solution works and not just on this specific dataset, we have prepared additional data that are available in our project repository, as well as instructions on how to carry out the same experiments so that it is possible to validate them. We have performed these experiments as well and will analyze them separately in this section.

All of the tests were run on the historical data from the initial dataset. The testing of the anomaly detection module alone does not yield significant results, since there is no way to compare the actual accuracy between the methods. Therefore, to test the performance, the algorithms are tested together with the prediction module. By using identical predictors over all tests and changing only the anomaly detection algorithms, it is possible to check the overall system accuracy and determine which anomaly detection method works best for the given use case.

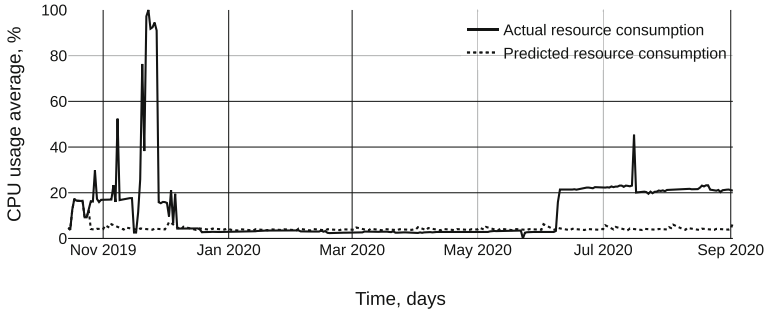
[2] Cloud Prediction Framework—https://github.com/Fruktus/CloudPredictionFramework.

[3] Nokia R&D Centre in Krakow—https://nokiakrakow.pl/.

[4] Polcom—https://polcom.com.pl/.



(a) Long-term prediction generated without anomaly detection algorithm



(b) Long-term prediction generated with WHA

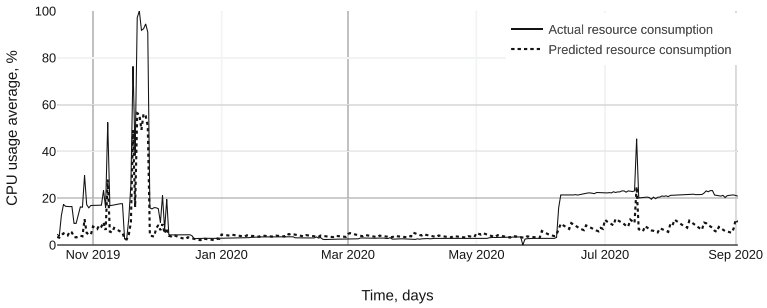
Fig. 3 Initial test of long-term prediction using WHA

For initial testing of the prototype system, the ten most stable and ten most unstable histories were chosen. The goal was to determine the accuracy of the system both visually and numerically as well as to fine-tune its parameters. The stability of the VM can be understood as the number of anomalies detected and the overall standard deviation of the sample. Choosing such virtual machines makes it possible to test the relatively "normal" scenario, which is assumed to be easier to predict, and also more random ones, which are believed to be more difficult to accurately model.

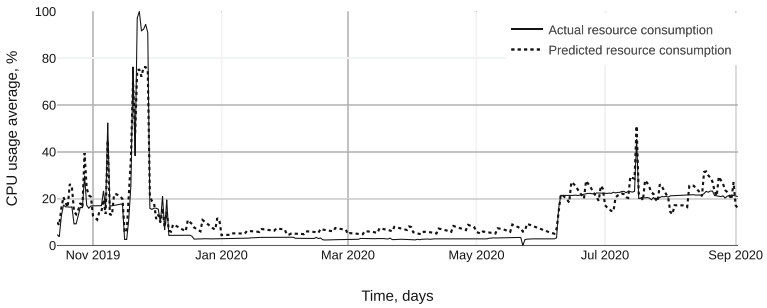
During initial testing, the trained prediction module was used to generate a resource usage prediction for a year ahead without (Fig. 3a) and with (Fig. 3b) the anomaly detection algorithm. The impact of the anomaly detection algorithm is clear, as the system predicts usage more accurately, although as expected, the accuracy decreases for long-term forecasts; however, large usage spikes are disregarded, which would lead to greater cost savings in real life.

The main experiments were run on all available data, with the aforementioned evaluation metric. The algorithms selected for comparison were the LA, the DLA and the WHA, with the latter developed by us. Moreover, the experiment was run without the anomaly detection mechanism as well.

Figure 4a, b shows the actual and predicted usage values obtained while running the system and constantly updating it with new data, as opposed to the previous experiment shown in Fig. 3, in which the system was not updated during prediction generation. The unstable usage



(a) Prediction system without anomaly detection algorithm



(b) Prediction system with WHA

Fig. 4 Comparison of prediction quality with and without the anomaly detection mechanism

pattern leads to significant difficulties in accurate prediction. By using the anomaly detection filter, we achieved noticeably better results, although in many cases underprovisioning would still occur in real-life scenarios. The difficulty occurs due to large spikes in usage at the beginning, which cause the solution to (inadequately) adapt to the dynamic changes. By removing these spikes using the anomaly detection algorithm, the solution can better adjust to the virtual machine's needs. In the end, predicting entirely random behavior is impossible, and in such cases the system can only attempt to mitigate the problems.

The experiments were run for varying numbers of virtual machines taken from the dataset, grouped by the deployment project. The purpose of using multiple machines was to average the accuracy and reduce potential errors from individual machines. The results gathered in Table 5 contain results for different metrics obtained from 5, 22, 49, 94 and 380 virtual machines (from the dataset used) while using the selected anomaly detection algorithms. The metrics present the system's ability to achieve results as close to the actual usage as possible, while minimizing any underestimates; therefore, a lower score means that the prediction module was able to more accurately model the actual behavior of the virtual machine. 100% represents 365 days of working time of a given machine, so an underestimate of 1% can be understood as about 3 days of under-provisioned system operation. The analysis of obtained results presented in Table 5 demonstrates that our approach (WHA) achieves the best results for both metrics in most cases. In the first case, the WHA exhibited an unusually high standard deviation of the Average Underestimate Frequency, which may be caused by the high amount of randomness in virtual machine usage patterns and the relatively low number

Table 5 Test results for different anomaly detection algorithms

Virtual machines	Algorithm	Average difference (%)	Stdev	Average underestimate frequency, (%)	Stdev
5	–	4.72	2.31	53.42	55.40
	LA	6.44	3.79	60.16	55.45
	DLA	6.83	4.66	43.84	58.10
	WHA	4.24	1.62	42.90	107.36
22	–	7.01	5.26	39.17	81.44
	LA	13.22	12.85	51.53	71.56
	DLA	7.13	5.05	39.49	52.34
	WHA	6.71	5.42	38.70	64.91
49	–	5.62	5.20	38.90	70.70
	LA	10.13	12.90	52.90	88.37
	DLA	6.74	7.72	41.66	58.81
	WHA	5.61	5.60	37.85	59.85
94	–	6.66	5.71	25.44	64.09
	LA	10.62	14.54	31.03	79.28
	DLA	6.61	5.75	28.51	69.10
	WHA	6.70	5.49	25.30	66.19
380	–	4.09	5.31	37.11	64.51
	LA	6.68	8.83	44.00	72.94
	DLA	4.15	5.27	44.97	74.11
	WHA	4.18	5.34	36.74	66.61

of virtual machines in the experiment, which results in a higher sensitivity to errors. In other experiments, the standard deviation for the WHA was relatively low.

We have also performed the experiments for the publicly available dataset.[5] The results are gathered in Table 6. The experiment contains just one set consisting of 98 virtual machines. As it can be seen, there is no clear best algorithm judging by the values alone, although our WHA algorithm has achieved almost exactly the same result as when no algorithm was employed; however, it also exhibited a lower deviation from forecast usage, which would lead to some savings. Our WHA and DLA solution performed similarly, which is due to WHA using an instance of DLA internally, although our algorithm can also detect recurrent anomalies, which DLA cannot. This particular dataset does not contain many such cases, which leads to the performance of both algorithms being similar.

To better visualize the performance of different methods, one experiment (for 22 VMs) was selected for visualization using a heat map, which includes additional metrics. The metrics presented are: Difference, Overestimate, Underestimate and Underestimate Frequency. All the metrics presented are colored from 1 to 4 based on their scores (a lower score is better). The values presented are meant for comparison only—they do not represent any specific units. The new metrics—Overestimate and Underestimate—show by how much the system underestimated machine requirements, as opposed to Underestimate Frequency, which shows how often the algorithm underestimated machine requirements. Those metrics are omitted in

[5] Dataset (examples)—https://github.com/Fruktus/CloudPredictionFramework.

Table 6 Public dataset results for different anomaly detection algorithms

Algorithm	Average difference, %	Stdev	Average underestimate, (%)	Stdev
–	15.40	14.67	17.76	37.00
LA	19.66	16.18	23.86	49.31
DLA	13.81	13.70	18.00	42.41
WHA	14.44	14.01	17.88	39.90

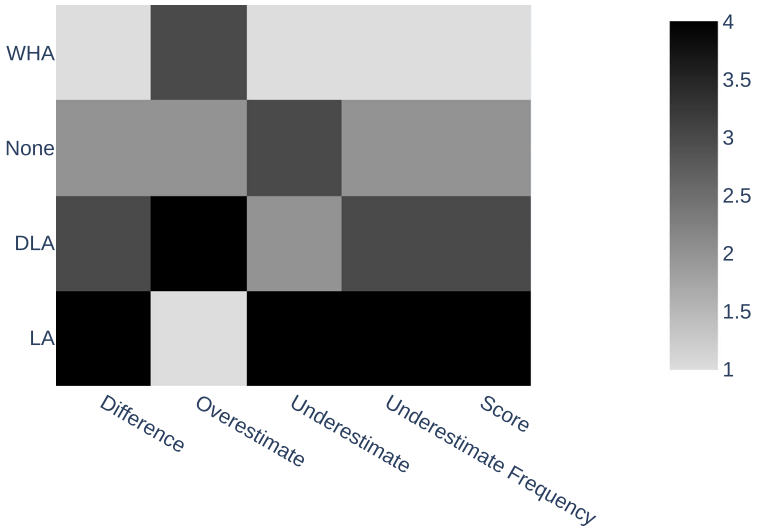


Fig. 5 Visualized metrics of the first experiment after scaling

Table 7 Price comparison of full-time VM use and when using WHA for reservations

Provider	Cost of full-time use	Cost with using WHA	Savings, (%)
Google Cloud	103,307$	49,497$	52.09
Microsoft Azure	25,458$	12,455$	51.08
Amazon AWS	32,068$	15,364$	52.09

Table 5 due to the mixed impact they have on overall performance, because in some cases less frequent, but more serious underestimates have a greater impact on the system than frequent but negligible ones. The analysis of the Heat Map in Fig. 5 (lower is better) reveals that our solution (WHA) achieves the best results (Score). As expected, the LA proved to be the least accurate solution. Interestingly, the absence of the anomaly detection module did not impair the predictions significantly in many cases, which was also noticeable in experiments, and often was the result of a highly anomalous system in which proper anomaly detection did not work reliably.

As a final benchmark, we used our solution to generate memory reservation plans for seven-day periods during which the settings of the virtual machine did not change. We gathered the data on the resources used as the basis for calculating the price of using all the 22 virtual machines for 24 hours each day for a year with the settings in question and subsequently compared it to the case in which the VM had the maximum amount of resources assigned

constantly. The experiment was carried out on the data from the experiment involving 22 virtual machines as presented in Table 5. We calculated the estimated cost based on the flat prices quoted by the three major cloud computing platforms (Google Cloud, Microsoft Azure and Amazon AWS). The prices are based on the cost of individual instances of on-demand virtual machines. The estimate is calculated through assigning 1, 2, 4, 8 or 16 gigabytes of memory to the machine on a week-by-week basis, depending on its predicted needs. The other hardware parameters of the machine were kept as similar as possible within and between services, but there were certain differences like the number of CPU cores. The results are presented in Table 7. It is important to note that these results were achieved for relatively stable virtual machines, with the assumption that it is possible to dynamically change the amount of memory assigned, which is often not the case. The presented figures are very rough estimates due to different pricing models and discounts available when reserving machines for longer time periods; however, the cluster deployment cost figure would remain roughly the same with and without our solution employed.

6 Conclusions

By analyzing real-world data, we demonstrated a new approach to dynamic resource usage prediction and planning, using our custom-built novel hybrid WHA anomaly detection algorithm to improve the accuracy of long-term predictions. Through our experiments, we evaluated the accuracy of our solution and presented its advantages over traditional methods. Experiments have shown that without our anomaly detection methods, over- and under-provision are often encountered, while our solution minimized both these problems and achieved the best results in most of the cases tested. Also, we presented a possible business use case scenario alongside the potential savings when using services from major cloud computing providers (Google Cloud, Microsoft Azure and Amazon AWS), which are made possible by improving the utilization of available resources. When analyzing one of the experiments (containing 22 virtual machines, which were running full-time for a year), we achieved a noticeable decrease in cost when using our solution compared to the price when maximum resources were assigned. For Google Cloud, the decrease was 52.09%, for Microsoft Azure it was 51.08% and for Amazon AWS, it was 52.09%.

The primary area of further research concerning our system is the possibility of building a prediction system that handles random usage characteristics as well as the impact of holiday-related anomalies on machine usage patterns. The second area could be replacing the default recurrent anomaly detection method with an FFT-based algorithm, which could potentially enable easier configuration at the expense of higher implementation complexity. Another opportunity for further research involves dynamically calculating confidence based on each sub-algorithm's performance, which could possibly lead to greater accuracy.

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

References

1. Ahad R, Chan E, Santos A (2015) Toward autonomic cloud: automatic anomaly detection and resolution. In: 2015 International conference on cloud and autonomic computing, pp 200–203
2. Al-Asaly MS, Bencherif MA, Alsanad A, Hassan MM (2022) A deep learning-based resource usage prediction model for resource provisioning in an autonomic cloud computing environment. Neural Comput Appl 34(13):10211–10228. https://doi.org/10.1007/s00521-021-06665-5
3. Albayrak S, Camtepe SA, Edman M, Yener B (2009) Host-based anomaly detection via resourceusage signatures. Technical report, Distributed Artificial Intelligence Laboratory - Technische Universitat Berlin, Berlin, Germany
4. Anupama K, Shivakumar B, Nagaraja R (2021) Resource utilization prediction in cloud computing using hybrid model. Int. J. Adv. Comput. Sci. Appl. (IJACSA) 12:373–382
5. Audibert J, Michiardi P, Guyard F, Marti S, Zuluaga MA (2020) Usad: unsupervised anomaly detection on multivariate time series. In: Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery and data mining, KDD '20. Association for Computing Machinery, New York, , pp 3395–3404. https://doi.org/10.1145/3394486.3403392
6. Borghesi A, Bartolini A, Lombardi M, Milano M, Benini L (2019) Anomaly detection using autoencoders in high performance computing systems. Proc AAAI Conf Artif Intell 33(01):9428–9433. https://doi.org/10.1609/aaai.v33i01.33019428
7. Borkowski M, Schulte S, Hochreiner C (2016) Predicting cloud resource utilization. In: Proceedings of the 9th international conference on utility and cloud computing, UCC '16. Association for Computing Machinery, New York, pp 37–42. https://doi.org/10.1145/2996890.2996907
8. Buda TS, Caglayan B, Assem H (2018) DeepAD: a generic framework based on deep learning for time series anomaly detection. In: Phung D, Tseng VS, Webb GI, Ho B, Ganji M, Rashidi L (eds) Advances in knowledge discovery and data mining. Springer, Cham, pp 577–588
9. Chandola V, Banerjee A, Kumar V (2009) Anomaly detection: a survey. ACM Comput Surv (CSUR) 41(3):1–58
10. Chengwei Wang, Talwar V, Schwan K, Ranganathan P (2010) Online detection of utility cloud anomalies using metric distributions. In: 2010 IEEE network operations and management symposium-NOMS 2010, pp 96–103. https://doi.org/10.1109/NOMS.2010.5488443
11. Grewal RK, Pateriya PK (2013) A rule-based approach for effective resource provisioning in hybrid cloud environment. Springer, Berlin, Heidelberg, pp 41–57
12. Hieu NT, Francesco MD, Ylä-Jääski A (2020) Virtual machine consolidation with multiple usage prediction for energy-efficient cloud data centers. IEEE Trans Serv Comput 13(1):186–199. https://doi.org/10.1109/TSC.2017.2648791
13. Kim J, Kim HS (2015) Pbad: perception-based anomaly detection system for cloud datacenters. In: 2015 IEEE 8th international conference on cloud computing, pp 678–685. https://doi.org/10.1109/CLOUD.2015.95
14. Lindh F (2019) Machine learning to detect anomalies in datacenter. Ph.D. thesis. http://urn.kb.se/resolve?urn=urn:nbn:se:uu:diva-388616
15. Liu W, Luo W, Li Z, Zhao P, Gao S (2019) Margin learning embedded prediction for video anomaly detection with a few anomalies. In: Proceedings of the twenty-eighth international joint conference on artificial intelligence, IJCAI-19. International joint conferences on artificial intelligence organization, pp 3023–3030. https://doi.org/10.24963/ijcai.2019/419
16. Malik S, Tahir M, Sardaraz M, Alourani A (2022) A resource utilization prediction model for cloud data centers using evolutionary algorithms and machine learning techniques. Appl Sci. https://doi.org/10.3390/app12042160
17. Mamatha E, Sasritha S, Reddy C (2017) Expert system and heuristics algorithm for cloud resource scheduling. Roman Stat Rev 65(1):3–18
18. Mehmood T, Latif S, Malik S (2018) Prediction of cloud computing resource utilization. In: 2018 15th international conference on smart cities: improving quality of life using ICT IoT (HONET-ICT), pp 38–42. https://doi.org/10.1109/HONET.2018.8551339
19. Mendoza, MA, Amistadi HR (2018) Machine learning for anomaly detection on VM and host performance metrics. Technical report, The MITRE Corporation

20. Nawrocki P, Osypanka P (2021) Cloud resource demand prediction using machine learning in the context of qos parameters. J Grid Comput 19(2):1–20
21. Osypanka P, Nawrocki P (2020) Resource usage cost optimization in cloud computing using machine learning. IEEE Trans Cloud Comput 1:251. https://doi.org/10.1109/TCC.2020.3015769
22. Rahmanian AA, Ghobaei-Arani M, Tofighy S (2018) A learning automata-based ensemble resource usage prediction algorithm for cloud computing environment. Future Gener Comput Syst 79:54–71. https://doi.org/10.1016/j.future.2017.09.049
23. Ravichandiran R, Bannazadeh H, Leon-Garcia A (2018) Anomaly detection using resource behaviour analysis for autoscaling systems. In: 2018 4th IEEE conference on network softwarization and workshops (NetSoft), pp. 192–196. https://doi.org/10.1109/NETSOFT.2018.8460025
24. Sauvanaud C, Kaâniche M, Kanoun K, Lazri K, Da Silva Silvestre G (2018) Anomaly detection and diagnosis for cloud services: practical experiments and lessons learned. J Syst Softw 139:84–106. https://doi.org/10.1016/j.jss.2018.01.039
25. Sniezynski B, Nawrocki P, Wilk M, Jarzab M, Zieliński K (2019) VM reservation plan adaptation using machine learning in cloud computing. J Grid Comput 17:797–812
26. Stalcup K (2021) Cloud computing trends 2021: optimization is #1 priority. https://www.parkmycloud.com/blog/cloud-computing-trends-2021/
27. Yoas DW (2013) Using forecasting to predict long-term resource utilization for web services. Ph.D. thesis, Nova Southeastern University
28. Zhang Y, Hong B, Zhang M, Deng B, Lin W (2013) ecad: Cloud anomalies detection from an evolutionary view. In: 2013 international conference on cloud computing and big data, pp 328–334. https://doi.org/10.1109/CLOUDCOM-ASIA.2013.57

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

Wiktor Sus is a PhD candidate and has received a Master of Science in Computer Science at the AGH University of Science and Technology. The principal areas of research he works in are anomaly detection and machine learning.