Hindawi
Mobile Information Systems
Volume 2022, Article ID 3725657, 13 pages
https://doi.org/10.1155/2022/3725657

Research Article

Towards SLA-Driven Autoscaling of Cloud Distributed Services for Mobile Communications

Carlos Miguel[1] **Víctor Rampérez,**[1] **Javier Soriano,**[1] **and Shadi Aljawarneh**[2]

[1]*Department of Computer Languages and Systems and Software Engineering, Universidad Politecnica de Madrid (UPM), Madrid, Spain*
[2]*Department of Software Engineering, Jordan University of Science and Technology (JUST), Irbid, Jordan*

Correspondence should be addressed to Carlos Miguel; carlos.miguel.alonso@alumnos.upm.es

Received 21 June 2022; Revised 14 September 2022; Accepted 19 September 2022; Published 3 October 2022

Academic Editor: Claudio Agostino Ardagna

In recent years cloud computing has established itself as the computing paradigm that supports most distributed systems, which are essential in mobile communications, such as publish-subscribe (pub/sub) systems or complex event processing (CEP). The cornerstone of cloud computing is elasticity, and today's autoscaling systems leverage that property by making scaling decisions based on estimates of future workload to satisfy service level agreements (SLAs). However, these autoscaling systems are not generic enough, as the workload definition is application-based. On the other hand, the workload prediction needs to be mapped in terms of SLA parameters, which introduces a double prediction problem. This work presents an empirical study on the relationship between different types of workloads in the literature and their relationship in terms of SLA parameters in the context of mobile communications. In addition, more than 30 prediction models have been trained using different techniques (time series analysis, regression, random forests) to test which ones offer better prediction results of the SLA parameters based on the type of workload and the prediction horizon. Finally, a series of conclusions on the predictive models to be used as a first step towards an autonomous decision system are presented.

1. Introduction

In recent years, cloud computing has become an essential technology in our daily lives due to the constant connection we maintain with the Internet through mobile systems and the elastic capabilities of this computing technique. These capabilities allow users to use and pay for the resources needed by acquiring and releasing them on-demand (pay-per-use or pay-as-you-go model), decreasing the cost of using this infrastructure [1]. Despite these model benefits, the service level agreements (SLAs), an agreement between the cloud customer and the cloud service provider, composed of service level objectives (SLOs) (we will use "SLA parameters" in the rest of this document), that meeting certain application performance levels [2, 3] must. The service must find the point at which the applications/services use the minimum amount of resources to fulfil the established obligations, which, in turn, minimizes both cost and energy consumption [4, 5]. This leads to a double-edged problem since both under- and over-provisioning (either does not have enough resources to process all tasks/requests on time, which is essential for mobile communications, or it has more resources than required to process the current load, respectively) could lead to saturation of the system or waste of resources, respectively [1, 6]. Both of these situations lead to the system operating outside the agreed SLA parameters, which constitutes an SLA violation.

Predictive autoscaling systems can use one or more variables to predict the system behavior and use this prediction to decide if it needs more resources, is using too many, or requires no changes in resource allocation. This decision triggers the appropriate scaling action.

In the literature, the workload itself is used to predict the system behavior by identifying different parameters (e.g., its trend to check how much the load is changing) and using them to predict the system load in the immediate future.

Despite this, the system could use other variables to predict its behavior, e.g., its level of saturation (e.g., CPU and RAM consumption) or its current performance and efficiency (e.g., the mean response time of the events received), resulting in a prediction based on results that might reflect the system status more accurately.

One of the major problems of using the workload as the predictor is the need for a precise definition due to the ambiguity of the term itself [7]. The definition of workload varies between systems, and so do its contents and structure. The workload of a publish/subscribe (pub/sub) system (e.g., a sensors system [8], a messaging service, etc.) workload is composed of publications, subscriptions, and unsubscriptions; a batch application workload is composed of processing orders; and a complex event processing system (CEP) workload that contains different events to be processed (e.g., text messages, weather reports, monetary transactions). This problem has a direct impact on any predictive autoscaling system not tuned to the expected workload pattern, for example, a system adjusted to receive a periodic workload will underperform if it changes to an unpredictable workload. Creating a sufficiently generic predictive autoscaling system capable of using any workload on any cloud service is a very complex task [9] due to the limited capabilities of this type of system caused by this ambiguity.

Furthermore, workload prediction might not produce direct information about the possibility of an SLA violation of the predicted workload, which is essential for the decision-making process of triggering a scaling action and the number of resources this action will manage. Also, this prediction might not reflect the number of resources used by the system, which does not provide a snapshot of its performance based on its SLA parameters (high-level metrics).

The autoscaling system will benefit from having additional information related to the SLA parameters (i.e., throughput and response time) to decide about said scaling action.

To get this extra information the autoscaling system must predict its SLA parameters to check if any will not be met with the forecasted workload it will receive, which creates a "double-prediction" problem (predicting the workload and SLA violations) that could lead to problems, mainly related to the dependence of the SLA prediction on the workload prediction. For example, the decision made by the SLA prediction might be incorrect due to an incorrect workload prediction, triggering a scaling action that might waste resources or produce an SLA violation since an imprecise workload prediction might lead to an unreliable prediction of the SLA parameters.

A solution to these problems might be a system that manages this "double-prediction" which might make it less efficient than a system that does not use the workload as the predictor and hence, does not have these problems. Furthermore, the complexity of implementing a system that treats these problems might be an obstacle to its development in comparison with a generic autoscaling system that does not have them.

To summarise, the main disadvantages of workload prediction-based predictive autoscaling systems are the following:

> Problem 1: the definition of workload is strongly coupled to the type of system (a workload is not defined in the same way for a pub/sub system as for a batch processing system), therefore, autoscaling systems that rely on workload prediction suffer from this same limitation, and therefore cannot be generic as they are system-dependent.
>
> Problem 2: autoscaling systems based on workload prediction need to map these workload predictions to the SLA parameters, which is not trivial and generates a double prediction problem in many cases.

To solve these limitations, we propose an autoscaling strategy to predict SLA parameters (e.g., throughput and response time), avoiding the "double-prediction" problem and making the system sufficiently generic to handle any workload on any cloud platform. This approach will use the most accurate prediction model for the three main workload patterns to forecast the system SLA parameters and produce a well-informed answer that will trigger any required scaling action. We have used 30 trained prediction models to measure their accuracy with each workload type and choose the most accurate one for further research.

This approach results in the following hypothesis:

Hypothesis: the accuracy of predictive autoscaling systems can be improved by choosing the appropriate prediction model to forecast the most relevant SLA parameters.

Due to the great relevance of event-driven architectures in current technologies, solving these problems is this research next natural step, to reduce the aforementioned cost while increasing the quality of service for end-users. In order to achieve this, this paper aims to empirically prove these limitations and find a prediction model that fits the requirements of the proposed autoscaling strategy which allows the further development of this autoscaling system.

We have conducted several experiments to gather information and empirically prove this information and its effects on the system related to its performance metrics and behavior to prove this hypothesis. (i) a study on the effects of different workload patterns (i.e., growing, periodic, unpredictable) have on different systems in terms of SLA parameters and SLA violations; (ii) an experiment to empirically prove the behavior and effects (i.e., changes in its performance metrics, saturation point) of different workload patterns on different systems, to solve the first problem previously mentioned; (iii) an empirical evaluation of the results of predicting the system SLA parameters using different predictive models and workload patterns on a cloud platform to find the best predictive model for each case.

Based on the results obtained from these experiments and evaluations, we elaborate a conclusion on which model is the best predictive model depending on some system parameters, such as environment, configuration, and workload pattern.

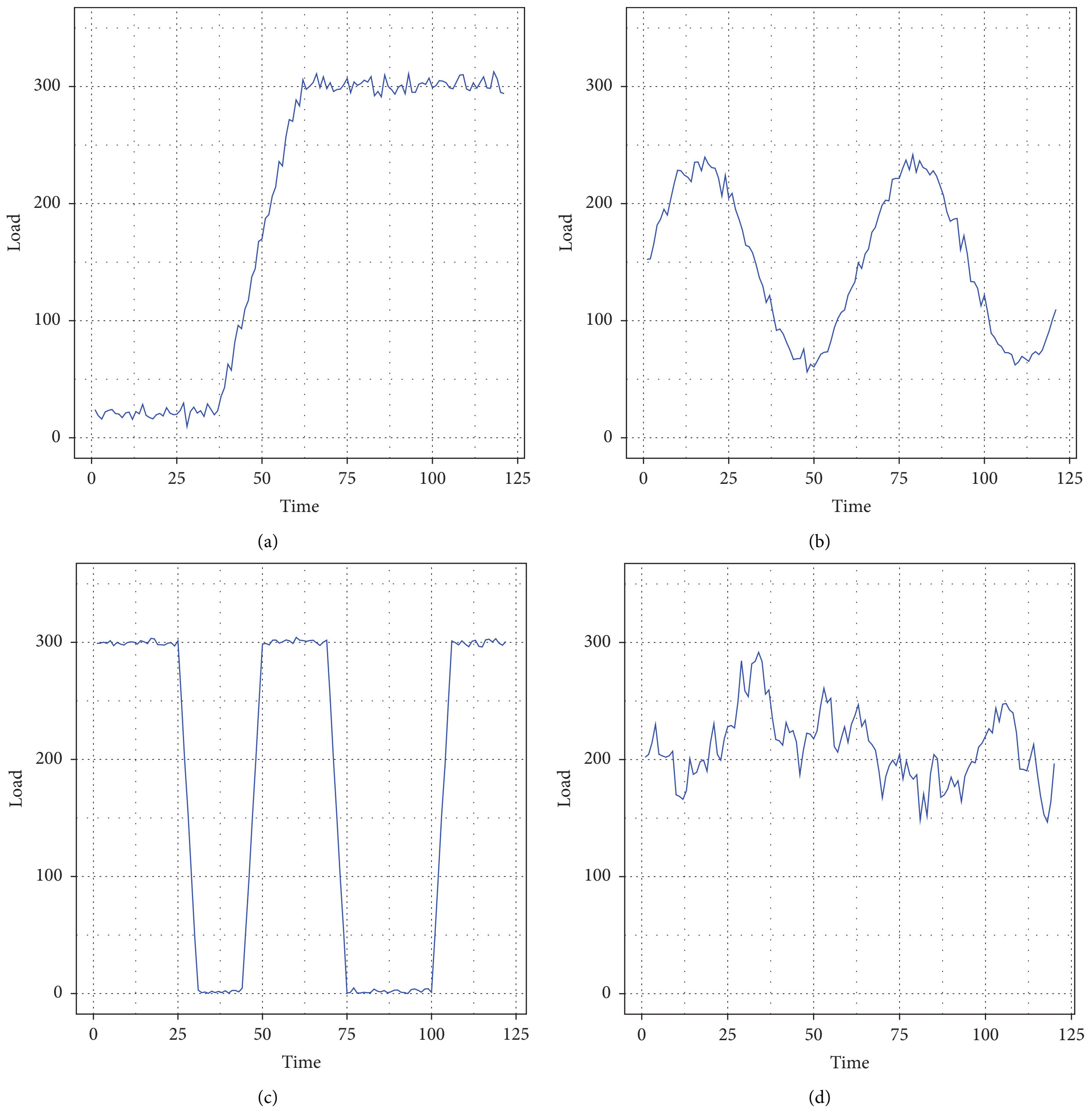


Figure 1: Workload patterns, left to right: growing, on-and-off, periodic, static, unpredictable.

The main contributions of this work are as follows.

(i) An empirical study on the relationship between different types of workloads in the literature and their relationship in terms of SLA parameters in the context of mobile communications.

(ii) An empirical study composed of more than 30 prediction models has been trained using different techniques (time series analysis, regression, random forests, etc.) to test which ones offer better prediction results of the SLA parameters based on the type of workload and the prediction horizon.

(iii) A set of conclusions on the predictive models to be used as the first step toward an autonomous autoscaling decision system is presented.

The remainder of this paper is organised as follows: Section 2 background and related work, introduces essential information about workloads and predictive autoscaling systems; Section 3 SLA parameters prediction approach, explains the relevant concepts of SLA parameters, our approach to predicting them and the experiments we have used to prove out hypothesis; Section 4 introduces the experiment environment and its results with a brief analysis of them;

Section 5 conclusions, presents the knowledge obtained from these experiments and the overview of this work; and Section 6 future work, explains what is ahead of this project and what are our next steps on this subject.

2. Background and Related Work

This section briefly introduces the essential concepts and related work used in this paper. Section 2.1 workload presents an overview of the concepts of workload and its different patterns present in the literature. Section 2.2 autoscaling systems introduces and explains the main principles of these systems and the types there are. Section 2.3 predictive autoscaling approaches presents the different autoscaling techniques and brief analysis of the available predictive models.

2.1. Workload. A generic yet inaccurate definition of the term workload is a series of events sent to a system of some sort that processes them or does something with them. Each type of system has a specific and well-defined workload which can be defined with great precision. According to [6, 7, 10], there are five major workload patterns in cloud computing environments.

(i) Growing workload is characterised by a rapid and constant load increase (Figure 1(a)).

(ii) Periodic workload is characterised by changes in the load spaced at regular time intervals (Figure 1(b)).

(iii) Unpredictable workload represents the load that cloud service will face once deployed, with constant fluctuations without a recognizable pattern or seasonal changes (Figure 1(d)).

(iv) Static workload is characterised by a load with no (or very small) changes or fluctuations (i.e., constant number of events).

(v) On-and-off workload represents a load with regular or occasional intervals without load (i.e., no user is interacting with the cloud service) (Figure 1(c)).

In this paper, we focus on growing, periodic, and unpredictable workloads, since they represent environments in which an autoscaling action might be required. Both static and on-and-off (batch application) workloads fall outside the scope of our work since applications with this type of workload do not require any autoscaling actions.

2.2. Autoscaling Systems. Autoscaling systems take advantage of cloud computing's key feature, elasticity, to automatically balance the resource allocation, complying with the SLA obligations by adapting the system resources to the demand at all times. The SLA parameters are the cornerstone of autoscaling systems since they are the reason these systems are necessary in the first place. In previous works [11], the authors used the system's high-level metrics (SLA parameters) together with its low-level metrics to build a model that maps these two metrics, which makes predictions and eventually generates a scaling decision. In [12], the authors use response time and throughput to make a more robust decision about the scaling actions.

According to [13], there are three main autoscaling systems based on the technique used to perform the scaling actions: reactive, proactive, and predictive. Reactive autoscaling systems are the most popular ones used on cloud computing and scale in/out according to the current system performance. Despite this, when they detect an SLA violation is occurring (or is about to occur), it might be too late for a scale action to avoid this SLA violation since his action takes some time, and during that time, the violation is well-underway [6, 10]. Proactive autoscaling systems allow the user to pre-define a scaling system schedule, which could solve some SLA violations but requires a predictable environment. That is not the case with predictive autoscaling systems that solve this problem by predicting its behavior and adjusting its resources to comply with SLA obligations [14].

2.3. Predictive Autoscaling Approaches. A predictive autoscaling system predicts future system behavior to adjust the application resources allocated in advance of changes in the environment (i.e., an increase in the workload) to meet its SLA obligations and minimise the hosting cost of the cloud application.

As shown in Figure 2, a predictive autoscaling system is composed of monitor, predictor, and decision maker components, which measure the chosen variable, such as the workload, predicts future values of it, and makes decisions based on the predicted values, respectively [14].

Autoscaling systems use different resource allocation techniques to manage the computing resources assigned to the system based on the decisions made by the decision maker component.

Resource allocation techniques can be classified into horizontal scaling (that is, adding new operator instances) and vertical scaling (that is, increasing the resources assigned to an already running system instance, such as memory or CPU). According to [10], since most of the cloud service providers do not support changing the resources allocated to an already running instance without rebooting it and the operating systems (OS) do not support this action, the most used resource allocation technique (and the one we will focus on) is horizontal scaling.

Lorido-Botran et al. [10] state that classifying this technique is also a difficult task due to the wide diversity of approaches found in the literature and because some techniques are a combination of two or more methods. The main techniques are (i) static threshold-based policies, (ii) reinforcement learning, (iii) queuing theory, (iv) control theory, (v) time-series analysis.

The static threshold-based policies are based on a set of rules, usually two, one for scaling out and one for scaling in, that use one or more performance metrics, involving several user-defined parameters: an upper and lower threshold, thUp and thDown, respectively, and two time-values, vUp and vDown, which denote the time interval in which the

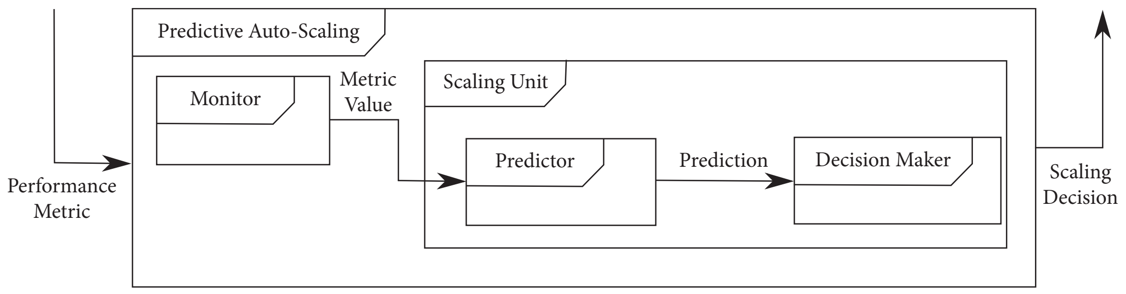


FIGURE 2: Architectural overview of a predictive autoscaling system. Extracted from [14].

condition must be met to trigger a scaling action. The user must define a fixed amount of resources (i.e., VMs or operators) to be allocated or deallocated once the corresponding scaling action is triggered. If a scaling action is performed, another will not be triggered for tUp/tDown seconds, respectively.

To perform a scale in action and allocate s resources, the performance metric must be greater than thUp for vUp seconds. Another scale-in action will not be triggered for tUp seconds, as shown in equation.

$$\begin{aligned} &\text{if } s > th\,\text{Up for } v\,\text{Up then}\\ &\quad \text{allocates}\\ &\quad \text{wait } t\text{Up}\end{aligned} \tag{1}$$

On the other hand, to perform a scale-out action to relocate s resources, the performance metric must be less than *thDown* for *vDown* seconds. If triggered, another next scale-out action will not occur for *tDown* seconds, as shown in equation.

$$\begin{aligned} &\text{if } s < th\text{Down for } v\,\text{Down then}\\ &\quad \text{de allocate s}\\ &\quad \text{wait } t\text{Down}\end{aligned} \tag{2}$$

For example, the autoscaling system will allocate 2 new instances if the CPU and RAM consumption of it is above 75% for more than 2 minutes.

Reinforcement learning techniques are based on learning through interaction between an agent, in this case, the autoscaler, and its environment, to automate the goal-directed learning and decision-making process. *Queuing theory* uses the mathematical study of queues to estimate the main system performance metrics, i.e., the response time or the average waiting time for requests. *Control theory* techniques aim to automate the management of systems such as data centres or storage systems to reduce the need for human input. These control systems are mainly reactive, but there are also some proactive approaches. *Time-series analysis* is mainly used to find a (repeating) pattern in the input or to forecast future values. This input (the time-series) is a sequence of values measured at sequential and evenly spaced time instants.

In [7], Masdari and Khoshnevis performed an in-depth analysis of the cloud computing workload prediction methods used in the literature, showcasing mathematical, data mining, and machine learning methods. Despite using different algorithms and techniques, these methods face the problems showcased in this work, such as detailed knowledge of the system to predict system variables (e.g., CPU, memory, and disk) using workload prediction. Furthermore, historical data are required to train models, and some methods take long periods to train and tune to the specific environment the system will face (e.g., twin support vector machine for regression (TSVR) [15]).

In [16], Nikravesh et al. proposed a new self-adaptive prediction technique focused on improving the accuracy of predictive autoscaling systems by choosing the appropriate prediction algorithm according to the incoming workload pattern. To this end, the workload pattern is identified by decomposing it into its *seasonal*, *trend*, and *remainder* components, using the LOESS [17] *R* package. Each workload pattern marks the usage of a different algorithm to predict its respective time series. To perform this prediction, the authors have used three different artificial neural network (ANN) algorithms; multilayer perceptron (MLP), multilayer perceptron with weight decay (MLPWD), and support vector machine (SVM). The system will identify the pattern of the received workload and automatically chooses the most accurate model to perform the workload prediction. The test metrics show that using the appropriate prediction algorithm for the incoming workload improved the prediction accuracy of the autoscaling system. Despite dealing with some problems mentioned previously, this solution still uses the workload as the predictor.

Predicting SLA violations is another method found in the literature that avoids the problems inherited by using the workload as the predictor. In [18], Leitner et al. proposed a predictive system that monitors and predicts SLA violations through machine learning and prevents them by triggering scaling actions when needed, called prevent. Through the regression used on the monitored data, the system can generate values that can be used to trigger the appropriate scaling action. These values are adjusted with *mean prediction error* (arithmetic average of the differences between predicted and monitored values for a given number of instances) and *prediction error standard deviation* (variability of the prediction error). Low predictive precision is expected until the system has enough data to make the necessary corrections. Despite this, the results of the experiments show that between 12% and 40% SLA violations could not be prevented, depending on the method and strategy used.

Table 1: Comparison of related works key characteristics.

References	Prediction algorithm	Technique	Performance Metric	Advantages	Disadvantages
[16]	Workload prediction	MLP, MLPWD, SVM	MAE, RMSE, PRED, R2PA	Uses best prediction algorithm automatically	Uses the workload as predictor
[18]	SLA violation prediction	Regression and MPE	Number of violations	Reduced impact of SLA violations	Low violation prevention rate (78% maximum)
[12]	Predict resource demand	SVM, NN, LR	MAPE, RMSE, PRED	Resources need prediction without workload as predictor	Complex system with 11 input parameters
[19]	Resource usage prediction	Bayesian information	MAE, RMSE, MAPE	Prediction based on resource usage history	Only applied to CPU load
[21]	Workload clustering, resource usage prediction	*K*-means, Bayesian learning	Delay, cost, SLA-violation rate, energy consumption	Considers SLA cost on the decision-making process	The prediction only takes one performance metric (response time)
[22]	Resource usage prediction	Fuzzy C-means, gray wolf	Energy consumption, execution time and cost, SLA violation and failure rates	Does not use the workload as the predictor	Two-stage prediction algorithm (clustering and GWO)
[20]	Resource need prediction	MVA	*Unknown* ([20] does not give enough information about performance metrics and evaluation)	Does not use workload as predictor	The prediction only takes one performance metric (response time)

A more direct approach to dealing with an unpredictable environment is by predicting the SLA parameters, also found in the literature, giving clear answers if a scale action is required. In [12], Bakole and Ajila developed and evaluated cloud client prediction models using support vector machine (SVM), neural networks (NN), and linear regression (LR), using 11 input parameters to predict CPU utilisation, response time and throughput of the system. For CPU utilisation and throughput, SVM shows the best results, and for response time, LR is the best prediction model. The resulting system can manage and predict the three key parameters correctly but results in a complex system that handles and monitors 11 input parameters, making it difficult to integrate them into a real system. In [19], Tofighy et al. proposed an ensemble CPU load prediction model that chooses the best prediction model using a Bayesian information criterion based on the resource usage history which, with the proposed framework for cloud resource management, achieved greater prediction accuracy that similar algorithms. In [20], Kouki and Ledoux proposed an SLA-driven autoscaling model that uses response time, service abandon rate, and service financial cost, combined with the SLA requirements, workload, and infrastructure information to compute the required scale actions using mean value analysis to achieve the required system configuration to meet the SLA requirements and expected QoS.

In [21], Ghobaei-Arani proposed a workload clustering-based resource provisioning mechanism that uses biogeography optimization with *K*-means clustering to classify the workload according to the system quality of service requirements. Additionally, the system uses a Bayesian learning technique to make the scaling decisions to fulfil the requirements. The test results show a reduced cost (due to using the SLA cost on the decision-making process), energy consumption, and SLA violations. Following a similar procedure, in [22], Ghobaei-Arani and Shahidinejad proposed a metaheuristic-based clustering mechanism used in combination with fuzzy *C*-means technique to find the clusters according to the quality-of-service requirements. These clusters are later used by a gray wolf optimizer to produce the appropriate scaling decision. The tests and simulations show that CPU utilisation efficiency, elasticity, and response time improved compared to other solutions.

To conclude this section, Table 1 contains a summary of the most relevant works mentioned previously comparing their key characteristics: (1) utilized prediction algorithm, (2) utilized technique, (3) performance metrics, (4) advantages of the model, and (5) disadvantages of the model.

3. SLA Parameters Prediction Approach

This section introduces the concepts related to SLA parameters and their prediction, along with a detailed overview of the empirical analysis that proves the statements about the SLA parameter-related effects of workload patterns on predictive autoscaling systems.

3.1. SLA Parameters. Sufficient prediction accuracy is necessary so the system SLA parameters (high-level metrics), for example, the response time and throughput, are within the constraints specified in the SLA throughout its execution. These values are agreed upon between the cloud customer and the cloud service provider and must be met to max the quality of service (QoS) offered to the end-users while minimising the cost of running the cloud application.

Following our experience from previous works [11], we have focused our efforts on the three most common high-level metrics, that are also the most relevant to our work; (i) response time, which is the time to process a message, which depends on the occupation of the system, since if the system

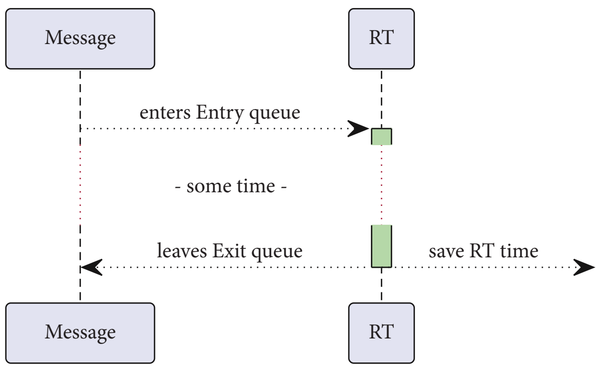


Figure 3: Measurement of response time.

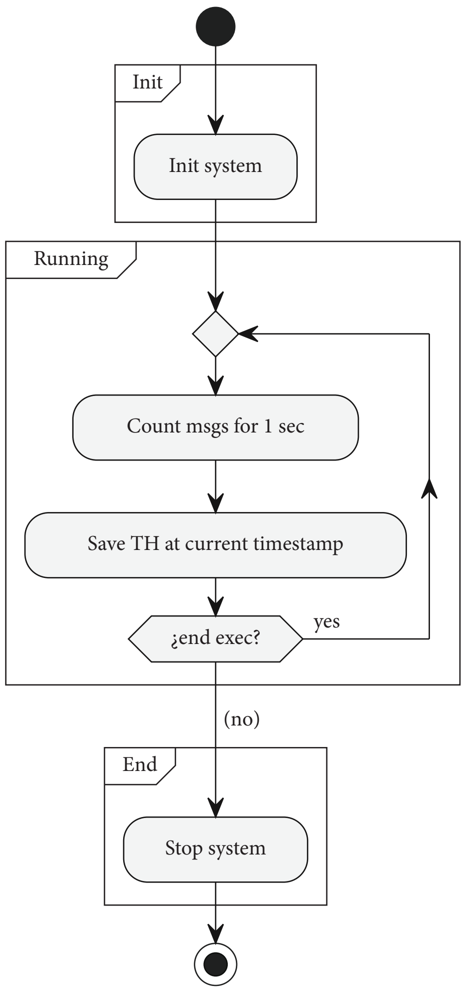


Figure 4: Measurement of throughput.

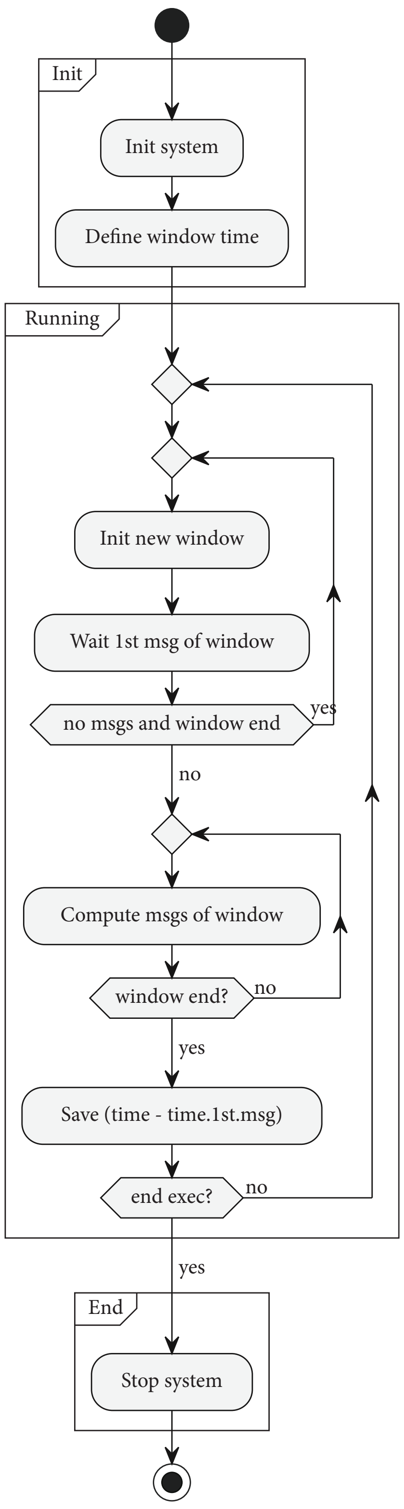


Figure 5: Measurement of window processing time.

is saturated, the response time will be higher because it has, in addition to the processing time (PT, the time required to process a single message), to wait for some time in the queue before being processed; (ii) throughput, the number of messages that the system can process per unit of time, also affected by system saturation, since a saturated system will output fewer messages per unit of time due to the higher processing time; and (iii) the window processing time (WPT), the time from the first message of the window to the moment the system closes the window and produces a result, over a defined window of time, which is also affected by system saturation.

Despite the diverse ecosystem of this type of system and its prediction methods and techniques, the type (pattern) of the workload it receives is also a key factor in its behavior since the same workload affects the SLA parameters of different systems in various ways. If a system adjusted to a workload pattern receives a different one, it will behave erratically and may trigger out-of-the-ordinary scaling actions, which may cause SLA violations and affect the QoS of the end-users. Choosing a prediction algorithm that fits the expected workload pattern and produces the best results is imperative to obtain the best prediction accuracy, as proven in [7, 10, 16].

3.2. Prediction of SLA Parameters. As mentioned above, the method for predicting the SLA parameters of the system are different in each predictive autoscaling system [11, 12, 14], despite this, the use of a monitor component is essential to measure the chosen prediction parameter(s), which, in our case, are the window processing time (WPT) [23], the response time (RT), and the throughput (TH). Together, these measurements represent a high-fidelity snapshot of the system status.

Each measurement is implemented with a different technique since they occur and start at different times and places. We attach a sequence number to each message so that each measurement can be used by the prediction algorithm later: (i) the response time (RT) measurement starts once the message enters the entry queue and stops when it exits the exit queue (see Figure 3); (ii) the throughput (TH) is measured each second, counting the number of messages output since the last measurement (see Figure 4); and (iii) the window processing time (WPT) is measured every N seconds, N being the time defined for the window, processing the messages on each window (see Figure 5).

The prediction algorithm will use these measured values to estimate future values and cross this knowledge with previous knowledge of the system behaviour using machine learning (ML) and deep learning (DL) methods, through which the autoscaling system will decide on the necessity of a scaling action.

3.3. Empirical Analysis. To prove our hypothesis, we studied the relation of the workload pattern of a system with its SLA parameters using different workload patterns in a single system, and each workload pattern in different systems. Through this technique, we have collected data on the SLA parameters of the system throughout this test execution to compare their results and to prove that workloads affect both SLA parameters and low-level metrics.

Furthermore, we have also conducted an empirical analysis with various workload patterns and predictive models, using multiple ML and DL methods to predict the SLA parameters and compare the results of each model to find which predictive model suits each environment in terms of system SLA parameters and behaviour.

For these tests, we have used three of the previously mentioned workloads (growing, periodic, and unpredictable) since they represent situations the system might face in a real environment. Each pattern helps measure the changes of the SLA parameters being the unpredictable workload closer to a real environment due to the sudden changes in

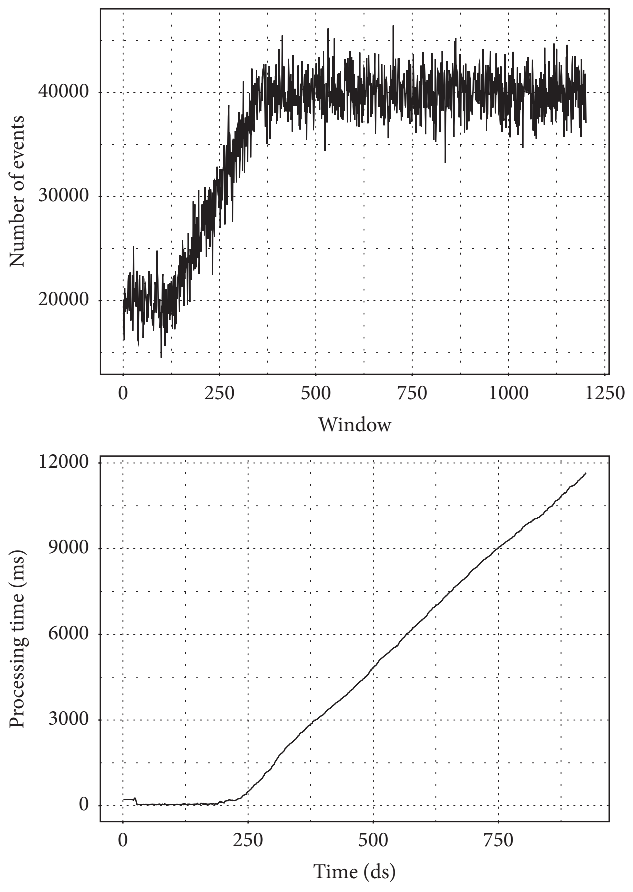
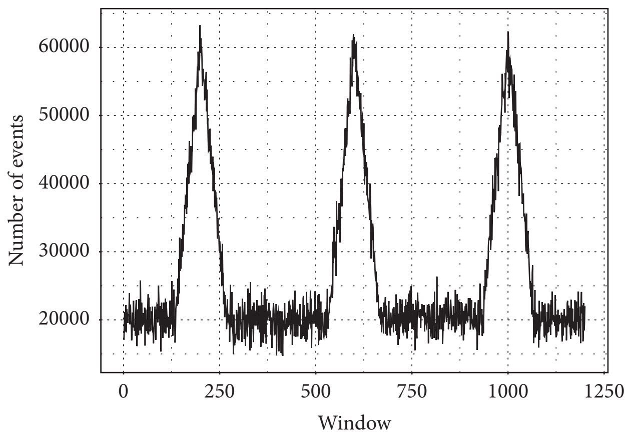


FIGURE 6: Growing workload and system WPT.

the load, testing the prediction models and their ability to keep the SLA parameters on the constraints defined by the SLA. Growing workloads stress the system to find its limits, whereas periodic workloads oscillate the system load to test if it can keep up with the constant changes.

The study results (see Section 4.2) have been gathered by measuring the system throughput and response time in the same configuration while applying different workload patterns and comparing their effects on these values to prove that they do affect them. We have used an analogous technique to measure the results of a workload pattern on different systems.

For the empirical analysis, based on the positive results observed in some previous and similar works [6, 11, 14, 16], we used time series (e.g., roller forecasting origin, ARIMA, and STL + ETS models) and machine learning and deep learning methods (e.g., linear regression, linear congruential generators, random forest, and neural network models). To evaluate each method's root mean squared error (RMSE) and mean absolute error (MAE) metrics and prove which predictive model is the most accurate, we have performed a k-fold cross-validation (https://otexts.com/fpp2/accuracy.html) ($k = 10$) comparing the deviation of predicted values from the actual values. Once trained, these models should predict the SLA parameters with the highest possible prediction based on the model and training data.



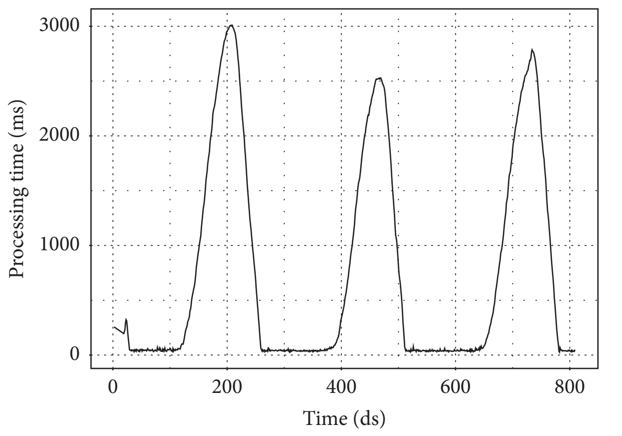


Figure 7: Periodic workload and system WPT.

Table 2: RMSE and MAE of prediction models with growing workload.

model	meanRMSE	meanMAE
Cubist	18.96	17.19
Gam	25.64	24.17
Bam	25.65	24.18
Qrf	61.41	56.99
gamLoess	65.96	64.74
parRF	67.97	63.96
Rf	67.98	63.97
Cforest	129.2	126.8
M5	134.2	131.5
M5Rules	134.2	131.5
svmRadialSigma	194.6	193.8
svmPoly	207.5	206.9
Nnet	292.4	291.3
svmRadialCost	313.1	305.2
svmRadial	317	309.1

A trade-off between training time and energy consumed is required to achieve the desired accuracy. The training phase may take more time than similar algorithms but reduces the energy consumed and time taken for each prediction while working in a live environment, which helps reduce the cost of running this system.

For each predictive model, we have used the SLA parameters measured from our system, running each of the chosen workload patterns, and applied these results to each predictive algorithm, measuring its RMSE and MAE and saving these measurements for later comparison and analysis.

Table 3: RMSE and MAE of prediction models with periodic workload.

model	meanRMSE	meanMAE
Cubist	32.42	28.46
Qrf	102.7	95.51
M5	104.8	97.69
M5Rules	105.4	98.23
parRF	114.3	107.7
Rf	114.3	107.7
svmRadialSigma	174.3	167.9
svmRadialCost	177	170.3
svmRadial	178	171.1
Gam	321.2	316.8
Bam	323.3	318.9
Cforest	369.1	365.3
gamLoess	648.1	644.1
Nnet	663.7	659.9
svmPoly	709.6	706.3

Table 4: RMSE and MAE of prediction models with unpredictable workload.

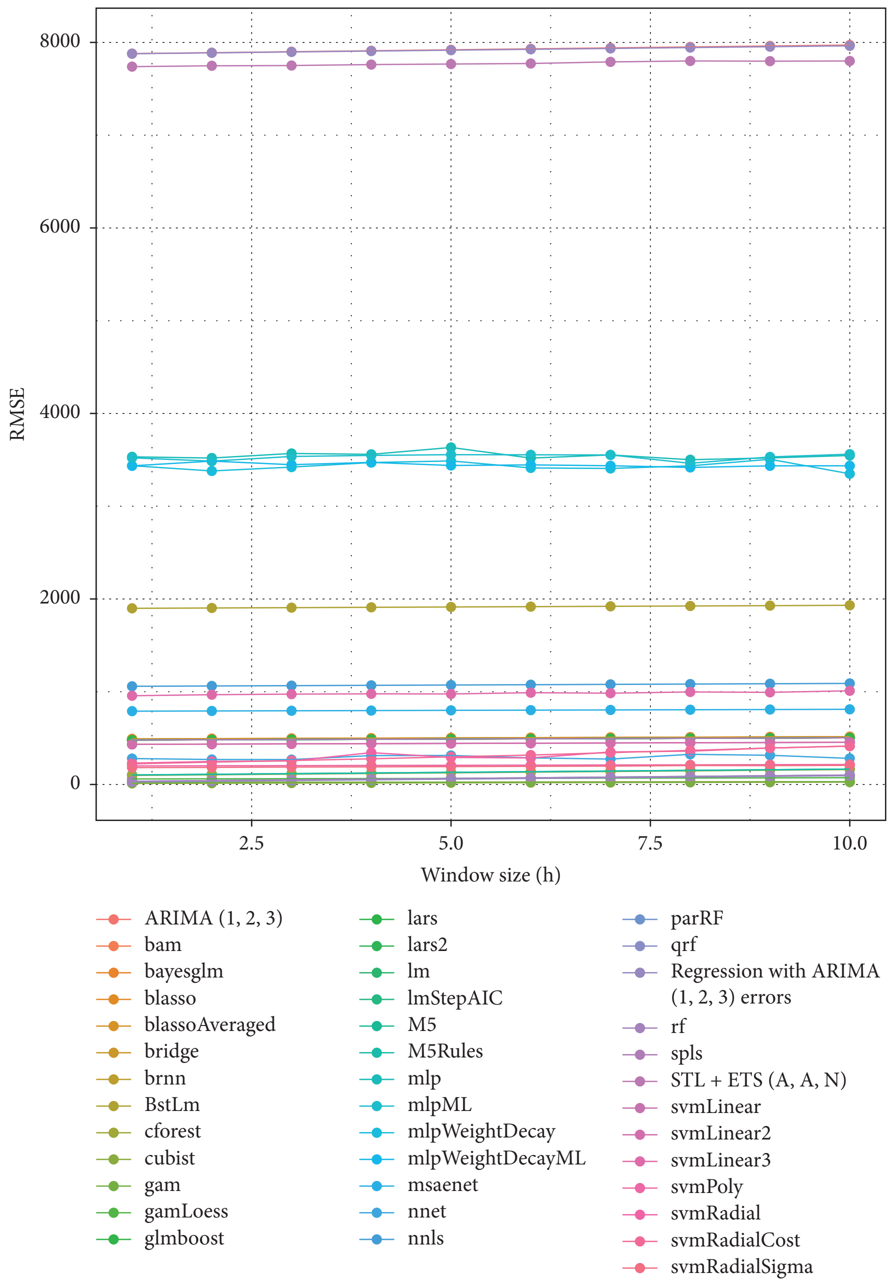
model	meanRMSE	meanMAE
Cubist	24.56	22.29
Qrf	27.09	24.47
parRF	29.1	26.59
Rf	29.11	26.6
M5	30.81	28.41
M5Rules	31.81	29.44
Bam	48.02	46.03
Gam	48.02	46.03
Cforest	81.52	79.77
svmRadialCost	88.72	84.74
svmRadial	90.79	86.79
svmRadialSigma	93.83	89.66
gamLoess	155	152.5
svmPoly	222.1	220.8
Nnet	233.2	231.8

4. Experimental Evaluation

This section presents information related to the experimental environment, relevant concepts, and the experiment results as well as their analysis.

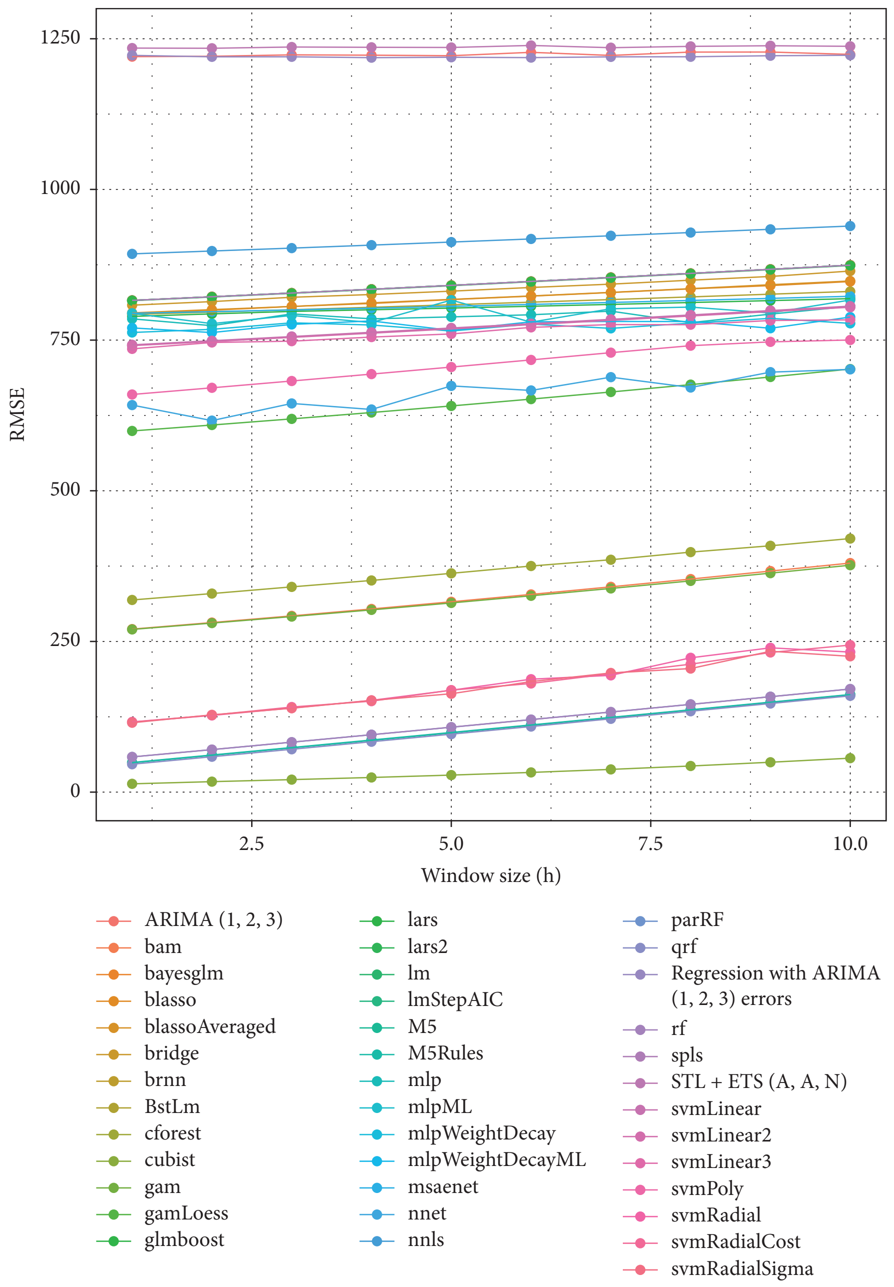
4.1. Experimental Environment.

4.1. Experimental Environment. The system we have used for the experiments, a complex event processing (CEP) distributed system, that calculates the topk using the MinTopK + N algorithm developed in [23] has been deployed using Apache Kafka and Zoopeker. For these experiments, we have used two T2 EC2 instances of Amazon Web Services cloud, each with 16 GB of RAM, 4 vCPUs running the 4.14 Amazon Linux kernel and Java OpenJDK 11.0.7 2020-04-14 LTS.



(a)

Figure 8: Continued.



(b)

FIGURE 8: Comparison of RMSE of prediction models with growing workload based on window size (h).

As a workload, we have used an over-sampled real dataset that contains tweets mentioning 400 verified users of Twitter, collected using the Twitter streaming API. We increased the number of users to 200 k, randomized the number of followers of each user, and added more user interactions to include some saturation points while keeping the load true to the original. For more information, see [23].

4.2. Experimental Results. Figures 6 and 7 show the effects of a workload on the SLA parameters, in this case, the window processing time. We have used growing, periodic, and unpredictable workload patterns, that, as figures show, affect SLA parameters.

In Figure 6, when the load increases (top graph), the window processing time increases constantly, even when the load stops increasing. This is due to the number of messages queued waiting to be processed since the system is saturated and cannot process messages fast enough to return to normal operations. Furthermore, in Figure 7, the spikes of the periodic load cause the window processing time to increase drastically in a very small window of time since the messages are queued very rapidly. Once the load spike ends, the system can process all those queued messages and return to its normal state. These spikes may cause SLA violations, since the system saturates and is unable to process the messages fast enough, which decreases the QoS for the end-users.

Tables 2–4 show the results of the empirical analysis that represent the root mean squared error (RMSE) and mean absolute error (MAE), which are the deviation of the predicted values from the expected ones of each model. Out of the 30 models we have tested, these tables show the 15 models with the least mean RMSE and MAE.

As shown in Table 2, the most accurate prediction model, with the least RMSE and MAE, is the cubists model, a rule-based regression model, followed by the gam and bam generalised additive models and quantile random forest, M5, and M5 rules random forest models. In contrast, in Table 3, all models show worse accuracy, with much higher RMSE and MAE. While being the most accurate, the cubist model shows worse accuracy than the previous, and the same happens with the gam and bam, and M5, and M5 rules. These random forest models are now more accurate than the previous quantile random forest model, using a periodic workload. We have also measured the RMSE and MAE of the predictive models using an unpredictable workload (Table 4) with similar results to the ones obtained with a periodic workload. The cubist model is still the most accurate, followed by the M5 and M5 rules models, also followed closely by the bam and gam models.

With these results, we see that the random forest cubist model is the most accurate predictive model regardless of the workload pattern, and the one to be used for a generic autoscaling system since our system will be workload-independent.

Furthermore, we have analyzed the effects of changing the prediction window on each predictive model. This window represents how far in the future the prediction is made. For example, a prediction window of 1 means that the predicted values correspond to the immediate next value; but if the prediction window is 5, the predicted values will correspond to the fifth next value. In Figure 8(a), we have measured the prediction accuracy for each predictive model using a growing workload, and as can be observed, the prediction error remains stable despite the increase of the prediction window. This means that the prediction window does not affect the prediction results significantly. Therefore, the forecasting models are robust enough to predict values over a distant forecast horizon with the same accuracy as the more immediate forecasts.

On the other hand, Figure 8(b) shows that the prediction models are not as robust for periodic workloads, as the prediction error increases slightly as the prediction horizon gets further away. This decrease in prediction accuracy is not the same for all models. For example, the cubist model has worse accuracy when the higher the prediction window, but this change is not significant enough.

To conclude this analysis, we have repeated this comparison with an unpredictable workload with the same results as the predictions with the previous workloads (Figure 8).

The results obtained from these experiments show that the workload does have a direct influence on the system SLA parameters, with the risk of SLA violations if the load spikes are not managed correctly; and the most suitable and accurate model for a generic predictive autoscaling system is the cubist random forest model, disregarding the received workload.

5. Conclusions

In this work, we have raised some problems with the predictive autoscaling approach based on workload prediction. In addition, we have proposed an alternative approach based on the prediction of the SLA parameters, thus avoiding double prediction problems and offering a solution less coupled to the type of system to be scaled and, therefore, more generic.

For this purpose, this work has developed an empirical experiment that has allowed us to analyze the impact of different workload patterns on the SLA parameters, demonstrating the problems of the classical approaches of autoscaling based on workload prediction. In addition, we have presented an experiment in which more than 30 predictive models were trained to predict the window processing time with different workloads of a CEP used in mobile communications. Finally, we have analyzed the results of this experiment to offer a set of conclusions and recommendations for the development of autonomous predictive autoscaling systems.

6. Future Work

One feature that would add great value would be to extend the study of prediction models. To this end, we consider several lines of action: testing whether using low-level metrics as predictors improves model predictions and predicting not only SLA parameters but their first derivative as well, to take into account the trend. Furthermore, we want to extend the study and experimentation shown in this work to other distributed cloud-based systems relevant to mobile communications, such as pub/sub systems. We also want to extend our evaluation to other workloads to study our findings in different environments.

Moreover, it would be of a great value to create an autonomous system capable of deciding which prediction model to use based on the conclusions of this work. In this way, the autoscaling system would choose the best prediction model for the SLA parameters based on the type of workload, system, and similar parameters. This autonomous system will be designed and implemented using the knowledge obtained from this and previous works to further improve our research.

Data Availability

The data used to support the findings of this study can be obtained from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

References

[1] S. Mustafa, K. Sattar, J. Shuja et al., "Sla-aware best fit decreasing techniques for workload consolidation in clouds," *IEEE Access*, vol. 7, pp. 135256–135267, 2019.

[2] J. Gao, H. Wang, and H. Shen, "Machine learning based workload prediction in cloud computing," in *Proceedings of the 2020 29th International Conference on Computer Communications and Networks (ICCCN)*, pp. 1–9, Honolulu, HI, USA, September 2020.

[3] M. R. Raza and A. Varol, "Qos parameters for viable sla in cloud," in *Proceedings of the 2020 8th International Symposium on Digital Forensics and Security (ISDFS)*, pp. 1–5, Beirut, Lebanon, June 2020.

[4] M. Shaukat, W. Alasmary, E. Alanazi, J. Shuja, S. A. Madani, and C.-H. Hsu, "Balanced energy-aware and fault-tolerant data center scheduling," *Sensors*, vol. 22, no. 4, p. 1482, 2022.

[5] M. Ghobaei-Arani, M. Shamsi, and A. A. Rahmanian, "An efficient approach for improving virtual machine placement in cloud computing environment," *Journal of Experimental & Theoretical Artificial Intelligence*, vol. 29, no. 6, pp. 1149–1171, 2017.

[6] A. Y. Nikravesh, S. A. Ajila, and C.-H. Lung, "Measuring prediction sensitivity of a cloud auto-scaling system," in *Proceedings of the 2014 IEEE 38th International Computer Software and Applications Conference Workshops*, pp. 690–695, Vasteras, Sweden, July 2014.

[7] M. Masdari and A. Khoshnevis, "A survey and classification of the workload forecasting methods in cloud computing," *Cluster Computing*, vol. 23, no. 4, pp. 2399–2424, 2020.

[8] R. M. A. Haseeb-Ur-Rehman, M. Liaqat, A. H. M. Aman et al., "Sensor cloud frameworks: state-of-the-art, taxonomy, and research issues," *IEEE Sensors Journal*, vol. 21, no. 20, pp. 22347–22370, 2021.

[9] A. Bahga and V. K. Madisetti, "Synthetic workload generation for cloud computing applications," *Journal of Software Engineering and Applications*, vol. 4, no. 7, pp. 396–410, 2011.

[10] T. Lorido-Botran, J. Miguel-Alonso, and J. A. Lozano, "A review of auto-scaling techniques for elastic applications in cloud environments," *Journal of Grid Computing*, vol. 12, no. 4, pp. 559–592, 2014.

[11] V. Rampérez, J. Soriano, D. Lizcano, J. A. Lara, and Flas, "FLAS: a combination of proactive and reactive auto-scaling architecture for distributed services," *Future Generation Computer Systems*, vol. 118, pp. 56–72, 2021.

[12] A. A. Bankole and S. A. Ajila, "Cloud client prediction models for cloud resource provisioning in a multitier Web application environment," in *Proceedings of the 2013 IEEE Seventh International Symposium on Service-Oriented System Engineering*, pp. 156–161, San Francisco, CA, USA, March 2013.

[13] V. Ramperez, J. Soriano, D. Lizcano, and C. Miguel, "Automatic Evaluation and Comparison of Pub/sub Systems Performance Improvements," *Journal of Web Engineering*, vol. 16, 2022.

[14] A. Y. Nikravesh, S. A. Ajila, and C. H. Lung, "Towards an autonomic auto-scaling prediction system for cloud resource provisioning," in *Proceedings of the 2015 IEEE/ACM 10th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, pp. 35–45, Florence, Italy, May 2015.

[15] X. Peng, "Tsvr: an efficient twin support vector machine for regression," *Neural Networks*, vol. 23, no. 3, pp. 365–372, 2010.

[16] A. Y. Nikravesh, S. A. Ajila, and C.-H. Lung, "An autonomic prediction suite for cloud resource provisioning," *Journal of Cloud Computing*, vol. 6, no. 1, p. 3, 2017.

[17] Statsmodels Org, "Seasonal-trend decomposition using loess (Stl)," 2022, https://bit.ly/3lykgJf.

[18] P. Leitner, A. Michlmayr, F. Rosenberg, and S. Dustdar, "Monitoring, prediction and prevention of sla violations in composite services," in *Proceedings of the 2010 IEEE International Conference on Web Services*, pp. 369–376, IEEE, Miami, FL, USA, July 2010.

[19] S. Tofighy, A. A. Rahmanian, and M. Ghobaei-Arani, "An ensemble cpu load prediction algorithm using a bayesian information criterion and smooth filters in a cloud computing environment," *Software: Practice and Experience*, vol. 48, no. 12, pp. 2257–2277, 2018.

[20] Y. Kouki and T. Ledoux, "Scaling: sla-driven cloud autoscaling," in *Proceedings of the 28th Annual ACM Symposium on Applied Computing, SAC'13*, pp. 411–414, Association for Computing Machinery, New York, NY, USA, June 2013.

[21] M. Ghobaei-Arani, "A workload clustering based resource provisioning mechanism using biogeography based optimization technique in the cloud based systems," *Soft Computing*, vol. 25, no. 5, pp. 3813–3830, 2021.

[22] M. Ghobaei-Arani and A. Shahidinejad, "An efficient resource provisioning approach for analyzing cloud workloads: a metaheuristic-based clustering approach," *The Journal of Supercomputing*, vol. 77, no. 1, pp. 711–750, 2021.

[23] V. Ramperez, S. Zahmatkesh, and E. D. Valle, "Scaling the monitoring of approximate top-K queries in streaming windows," in *Proceedings of the 2021 IEEE International Conference on Big Data (Big Data)*, pp. 181–189, IEEE, Orlando, FL, USA, December 2021.