# A Forecasting Methodology for Workload Forecasting in Cloud Systems

Francisco Javier Baldán, Sergio Ramírez-Gallego, Christoph Bergmeir, Francisco Herrera, *Member, IEEE*, and José M. Benítez *Member, IEEE*

**Abstract**—Cloud Computing is an essential paradigm of computing services based on the "elasticity" property, where available resources are adapted efficiently to different workloads over time. In elastic platforms, the forecasting component can be considered by far the most important element and the differentiating factor when comparing such systems, with workload forecasting one of the problems to solve if we want to achieve a truly elastic system. When properly addressed the cloud workload forecasting problem becomes a really interesting case study. As there is no general methodology in the literature that addresses this problem analytically and from a time series forecasting perspective (even less so in the cloud field), we propose a combination of these tools based on a state-of-the-art forecasting methodology which we have enhanced with some elements, such as: a specific cost function, statistical tests, visual analysis, etc. The insights obtained from this analysis are used to detect the asymmetrical nature of the forecasting problem and to find the best forecasting model from the viewpoint of the current state of the art in time series forecasting. From an operational point of view the most interesting forecast is a short-time horizon, so we focus on this. To show the feasibility of this methodology, we apply it to several realistic workload datasets from different datacenters. The results indicate that the analyzed series are non-linear in nature and that no seasonal patterns can be found. Moreover, on the analyzed datasets, the penalty cost as usually included in the SLA can be reduced to a 30% on average.

**Index Terms**—Cloud Computing, elasticity, workload forecasting, machine learning, time series forecasting.

## 1 Introduction

Cloud Computing (see, e.g., [1] for an introduction) is a trending topic. In Gartner's 2013 CIO Agenda [2], the term "Cloud" is in the third position in the top ten technology priorities for that year, and the term "Analytics and Business Intelligence" is first. Cloud Computing is changing the way we think about computing, and it is both a new field of research for the scientific community, and a profitable area from which new business models are emerging. In fact, Cloud Computing makes feasible the idea of computing as a utility, comparable to power, water, light, or communications infrastructure [3]. The cloud paradigm can be defined as a new paradigm of computing services, centering its efforts in the characteristics of pay-per-use and the ability to provide seemingly unlimited resources to end-users.

Among the most outstanding underlying properties of this paradigm, you can find "self-scaling" and "elasticity," which are based on the system's ability to adapt available resources to different workloads over time, without resulting in inefficient use of these resources. From a computational complexity point of view, this problem is NP-hard. So the application of heuristic techniques is required to reach an automated and efficient solution. A service that is provisioned in an inadequate manner can yield two kinds of consequences for service providers: over-provisioning (downward forecasting) and under-provisioning (upward forecasting). Since a cloud system often cannot react in an immediate way to changes of variables, a predictive system is necessary to manage resources in an intelligent way. This forecasting component can be considered the most important element in those systems and a good forecaster can be the differentiating factor when comparing elastic systems.

However, most real systems developed until now are purely reactive. This means that these systems work with near-future, ignoring information from past values. Because of the fluctuating resource demand (e.g., e-commerce applications) this kind of system does not obtain satisfactory results, and problems such as sudden *Traffic surge* or *Slashdot effect* [4] arise. Thus, a solution is necessary which is capable of capturing more complex patterns present in real life, and fundamentally focusing attention on capturing trends, seasonality and other features that may be present in real-world workloads.

In the literature, traditional regression methods and more elaborate solutions have been used in many cloud systems to forecast the workload. Moreover, Machine Learning (ML) algorithms have also been used to model this problem. However, current predictive systems offer ad-hoc and specific solutions to forecast the workload of a cloud system, typically applying a set of methods without a thorough, state-of-the-art statistical evaluation of the problem from a time series point of view. This

- *F.J. Baldán, S. Ramírez-Gallego, F. Herrera, and J.M. Benítez are with the Department of Computer Science and Artificial Intelligence of the University of Granada, CITIC-UGR, iMUDS, Granada, Spain, 18071. E-mails: {fjbaldan, sramirez, herrera, J.M.Benitez}@decsai.ugr.es*
- *C. Bergmeir is with the Faculty of Information Technology, Monash University, Melbourne, Australia. E-mail: christoph.bergmeir@monash.edu*

evaluation is important for any forecasting problem (as in workload forecasting) since none of them can be properly addressed without a preliminar stage to analyze and model properly the properties of the series before applying any method. The benefits from the application of a comprehensive forecasting process are well-known in many fields, such as: health care, industry, science or bussiness [5].

While there are many solutions in the literature based on existing forecasting tools, there is no general framework or methodology that addresses the workload forecasting problem analytically from a state-of-the-art time series forecasting perspective. Besides in the cloud field, we perceive a gap between forecasters and domain experts in the sense that well-structured state-of-the-art forecasting methodology is not commonly employed in this area.

For example, one of the main results of the M3 forecasting competition is that simple methods often outperform more complex ones [6], and model performance depends on the underlying data of a forecasting problem. More recently, the NN3 competition [7] yielded as a result that none of the participating Machine Learning methods was able to outperform the theta method, which is equivalent to simple exponential smoothing with drift, which is a relatively simple standard method in time series forecasting.

These findings lead us to affirm that a methodology is necessary that assesses data characteristics and chooses the most appropriate forecasting method accordingly, i.e., that chooses complex methods only where appropriate. So, this paper aims to provide three main contributions in this field. First, this work is aimed at developing a case study in a completely detailed and rigorous way. The second objective is to promote the use and apply a full, sound state-of-the-art forecasting methodology to the workload forecasting problem in Cloud Computing. The methodology entails a deeper study for the workload analysis and modeling of time series by analyzing the statistical properties (stationarity, seasonality, trends, etc.) of the series, performing linearity testing, and analyzing the auto-correlation structure of the series. As for the third contribution the nature of the problem will lead us to the proposal of a specific cost function better suited than other error formulations commonly used. This study will allow us to better understand the properties of the series and to enhance the results through fitter forecasting models.

Finally, to demonstrate the effectiveness of this approach, we apply this methodology to several real world datacenter workload datasets, one of them from Google's clusters.

The rest of the paper is organized as follows. Section 2 presents a brief overview of elasticity and workload forecasting in cloud systems, and reviews the most relevant related works. Section 3 details the case study targeted in this paper. Section 4 introduces a specific cost function. Section 5 presents the proposal of a forecasting methodology to address the workload forecasting problem in elastic cloud systems. Section 6 introduces our experimental framework, and Section 7 discusses the forecasting methodology and the experimental results. Finally, Section 8 concludes the paper.

## 2 Preliminaries

In Section 2.1, we introduce the concept of elasticity applied to Cloud Computing, a concept used to describe the automatic scaling behavior of cloud systems. This elastic reaction cannot be done in an automatic fashion without a forecasting component, which is capable to foretell with some degree of certainty the future demand of resources in a cloud system.

Section 2.2 describes the related work in the literature on the elastic cloud systems topic and how these proposals address the specific problem of workload forecasting through a wide range of forecasting techniques.

### 2.1 Elasticity in Cloud

Frequently, a user of a cloud platform will not need to acquire all the resources initially, but requires a variable amount of resources which will be provided or released by the provider in a dynamic fashion, according to the actual demand. Thus, resources are provided in an elastic manner [8].

It is important not to confuse the terms scalability and elasticity. Scalability requires a manual installation and deployment of the resources. So, it depends on an early detection of problems. Elasticity involves an instantaneous and automatic response to problems, so that elasticity can be seen as a synonym of self-scaling. The resources are increased or decreased in an automatic fashion, without human intervention.

This situation of dynamism becomes especially interesting for systems which in certain times need to face peak demands [9]. In these cases, it is more interesting to have a cloud platform that can embrace this sort of eventualities, instead of having an infrastructure that is mainly underused. Indeed, the key is the ability of the platform to assign or release resources in a timely and fine-grained manner, so that we create a close alignment between actual demand and the resources cap [10].

Accordingly, elasticity bears a great number of advantages, including: smaller self-provisioning cycles, an easier maintenance of server applications, a quicker adoption of new web-business models, a better user experience, a reduced complexity, and higher integrity and security levels.

Additionally, cloud clients, as users of a computing service, sign a Service-Level Agreement (SLA) with a service provider, which guarantees a specific service quality. SLA and costs are typically in direct proportion, but one would like to satisfy the SLA level while also keeping low costs. Hence, we can conclude that capacity and demand in a cloud platform must be planned both

for average load and peak load. Nevertheless, each one of these approaches is somewhat different [11].

When provisioning Cloud resources for an average load (under-estimation) the provider incurs in lower costs because less hardware is acquired, but the performance will suffer when a peak demand occurs (under-provisioning). This bad performance will incur SLA penalties, so that the benefit and prestige of the provider will be undermined. On the other hand, if the system's capacity is established for the maximum (over-estimation) so that a peak load does not affect system performance, the resources will be underused for some the time, incurring unnecessary expenses of infrastructure (over-provisioning). Both problems are among the most important challenges in cloud workload forecasting [1].

If we choose adding or removing resources every time system load changes, this implies a wasteful additional expenditure of resources to boot up or turn off machines. Furthermore, another variable to be considered is the time it takes for a resource to become available or unavailable: latency time. Comparing several public cloud providers [12] [13] [14], we can observe that the average scaling latency time to allocate a new virtual resource instance is less than 10 minutes (typically 5 minutes), which is relatively small compared with non-virtual computer paradigms.

Due to latency time used by leveraging resources, it is mandatory that the elasticity decisions are made as fast as possible. In addition, the decisions not only have to be quick, but also effective, since if the result is wrong, then the latency time is expanded and Quality of Service (QoS) is downgraded. Accordingly, a desirable solution requires to forecast system workload at any time, and provisioning resources in advance and in a fully automatic manner. Thereby, the system will be able to manage input workloads in a near future. So, a well-designed forecasting module that is able to forecast accurately the resources required by the cloud system is necessary.

## 2.2 Related work

There are currently two kinds of solutions to cope with the problem of elasticity in cloud platforms. The first one is basically reactive, based on the current state of the workload in the system. This perspective is controlled making use of scaling rules defined by a human expert or the client in the SLA. The other choice is based on the introduction of a component for workload forecasting.

In [15], a survey and classification of several elasticity solutions, both reactive and predictive, for Cloud Computing is presented. In this work, we center our attention in the systems that may be defined as proactive or predictive, namely solutions that use ML or mathematical/analytical techniques to forecast the system load before making scaling decisions. This section provides an overview of predictive approaches.

### 2.2.1 Traditional regression methods

In the following we describe solutions which use traditional regression methods to forecast the system's workload changes. These solutions are based on traditional regression algorithms like linear or polynomial regression and auto-regressive models like ARMA.

Elastic VM [16] uses Virtual Machine (VM) resizing as a method to maintain the SLA within its bounds. In this case, the forecasting scheme is very simple, it only depends on the last CPU allocation and consumption.

Iqbal *et al.* [17] present a mixed approach using a reactive model for under-provisioning and a predictive model for over-provisioning. Those authors use polynomial regression to forecast the number of web and database server instances for the current observed workload.

Tirado *et al.* [18] present an elastic web infrastructure that adapts to workload variations by self-scaling dynamically. For this purpose, those authors use an ARMA model capable of capturing seasonal patterns. Although those authors present a general solution for elastic web infrastructures, the forecasting part does not represent a general framework for workload forecasting, which is our aim.

Roy *et al.* [11] also use forecasting algorithms like ARMA to forecast the workload of the application and estimate the system behavior. The main objective of that work is to solve a linear equation formed by variables, such as: application QoS, used resources, and associated costs.

### 2.2.2 Machine Learning algorithms

In this part, we present ML-based methods proposed in the elastic cloud predictive literature.

Reig *et al.* [19] propose a system with two modules of forecasting: one for short-term forecastings, and another one for long-term forecasting. The latter makes use of ML based models (M5P, REPTree, and a bagging approach) to forecast CPU load and memory usage.

Moore *et al.* [20] present a system that comprises of three models running in parallel: a time series forecaster based on the SVR algorithm and two incrementally updatable naïve Bayes models, learning online while the system is acting.

In another work, Imam *et al.* [21] use a Time Delay Neural Network (TDNN) which is a special type of ANN for analysis of temporal data. In addition, they also train a set of regression models like a non-linear polynomial regression model. However, the developed models are proven on grid computing workloads, not on cloud traces, which are different in nature.

The closest approach to our work is presented by Islam *et al.* [22], where some inherently efficient and effective ML techniques like Error Correction Neural Network (ECNN), and linear regression are used. Those authors also incorporate additional data partitioning methods in the training stage like sliding window and cross-validation techniques.

#### 2.2.3 Related solutions

Finally, we discuss some other related proposals present in the literature that do not fit the kinds of methods commented on above.

PRESS [23] is a proposal of an elaborate solution based on signal processing techniques, such as Fast Fourier Transform (FFT), to identify repeating patterns called signatures. The signatures are used to perform CPU load forecastings.

CloudScale [24] is built on PRESS and contributes two complementary ideas for the under-estimation error: online adaptive padding and reactive error correction.

Kingfisher [25] is a cost-aware system that integrates multiple elasticity mechanisms. Those authors formulate a provisioning problem as an Integer Linear Programming (ILP) problem to account for both infrastructure and transition cost for deriving appropriate elasticity decisions. Regarding the forecasting part, in their experiments they assume a perfectly accurate (in practice unrealistic) forecaster to evaluate their ILP equation.

A large set of solutions have been discussed above. In contrast to those works, we use a forecasting methodology based on a wide variety of models and statistical tests. Likewise, we follow a statistical analysis and forecasting process instead of simply and blindly applying forecasting models. Although other proposals have also used models considered in our methodology (e.g., ARIMA or ANNs), these techniques have been applied without performing an appropriate analysis or proven methodology; ignoring, for example, some interesting pattern information (seasonality, stationarity, etc.) that could be used to improve the forecasting outcomes.

## 3 A Forecasting Case Study: Workload in Cloud Systems

This section details a forecasting case study for workload on cloud systems. The study highlights the most relevant parameters and considerations to take into account in these problems.

The general structure of an elastic cloud system is depicted in Figure 1. This system has to efficiently manage a set of virtualized resources in an elastic manner. Briefly, an elastic cloud system has as inputs the current platform state (obtained from a real-time monitoring process) as well as the forecast for the following states, made by the workload forecasting module. This module will use historical data to produce forecasts that are sent to the resource provisioning system. In turn, this module will make decisions that are forwarded to the management component. Finally, the resource management component will take actions (turning on/off virtualized resources) on the resources pool according to the received decisions.

The workload forecasting module component can be considered as the most significant part in elastic systems. In Figure 1, we present the diagram of a prototype system, where our forecasting module is included as part of it. Likewise, we describe the design of this important component, detailing its inputs and outputs.

In elastic cloud systems, the main variables to consider are usually related to the use of the main resources offered by the platform, like CPU and memory utilization, disk space, bandwidth, etc. These variables must be properly studied, forecasted and managed so that the system can make appropriate provisioning and management of resources at any moment. CPU load and memory use can be seen as the most important and limited resources in a computer system, and the main bottlenecks in cloud platforms. Due to the limited space available in the paper we will focus on just one of them. The study and subsequently defined methodology can be derived in a similar fashion for other resources.

In this case study, we focus on CPU usage (also known as *CPU rate*) as a measurement in units of CPU core per time unit. We are using seconds as time unit, hence the CPU quantum for billing is a core-second. The values of CPU rate at each second composes a time series, which is fed to our forecasting module.

It is also necessary to decide which is the most appropriate time unit for modeling our time series, as this depends on the latency period of given resources (e.g., VMs). This time unit will be used as the forecasting horizon for one-step-ahead forecasting. As discussed in Section 2.1, VM resources can be generally supplied every 5 minutes. So the speed required in cloud systems limits the granularity of the time series to be coarse-grained, greater than or equal to the latency period. In this paper, we choose two different time units as the most appropriate ones:

1) **5-minute interval**: we consider adjusting to the latency period as much as possible is a good choice to achieve accurate and efficient forecasts. This could be named *operational horizon*.
2) **hourly**: normally, in many forecasting problems hourly data is used to detect time patterns and seasonality.

Hence, the output of the forecasting module can be defined as the total amount of virtualized resources (measured in number of cores for CPU) required by the system in the next time step. This information will be used by the resource provisioning component to manage the available resources. As we measure the output in terms of number of cores, the output is completely independent of the studied system; it does not matter whether it is a small group of machines or a data center with thousands of machines.

Regarding the measurement of error, the loss associated with over-provisioning and under-provisioning will be typically asymmetric. And while both of them could be measured in monetary terms, i.e., penalty cost for under-provisioning and energy cost for idle nodes, their cost are rather imbalanced. Hence, while standardized forecasting measures may provide indicative guidance

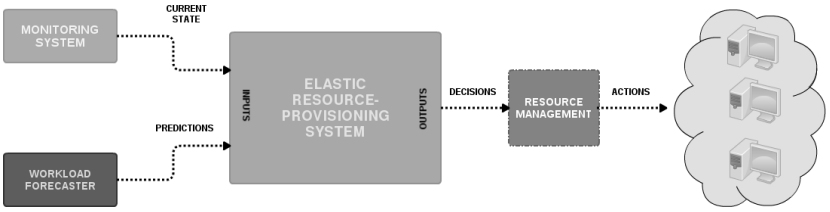


Fig. 1. Elastic Cloud System

for the forecaster evaluation, it is necessary to develop a better suited measure for this problem.

## 4 Evaluation Criteria

A number of different error measures are commonly employed in time series forecasting [26]. To assess the models considered in this work, we employ RelMAE and sMAPE. However, as noted in Section 3 the cost of errors in Cloud system Workload Forecasting (CWF) problems is not symmetric. To have a more appropriate evaluation for this regard, a specific cost measure is formulated in this section. In the following, we describe all error measures used and explain why they are used in the experiments.

### 4.1 Error measures: RelMAE and sMAPE

A problem with error measures such as the root mean square error is that they are not scale free and cannot be used to compare forecasts for different series. For series with strictly positive values, an effective alternative is to normalize by the value of the series, which yields percentage measures. We use the commonly employed sMAPE measure [6]:

$$\begin{aligned} \text{sMAPE} &= \tfrac{1}{n}\textstyle\sum_{t=1}^{n} 100\ \tfrac{|y_t-\hat{y}_t|}{m_t}, \\ \text{with}\ \ m_t &= \tfrac{|y_t|+|\hat{y}_t|}{2}. \end{aligned} \tag{1}$$

Furthermore, we use the RelMAE, which is the Mean Absolute Error (MAE), normalized by the MAE of a benchmark method. We use the naïve forecast (where forecasts are equal to the last observed value) as benchmark $MAE_B$. The RelMAE indicates if the forecast performs better than a trivial method, and therewith performs a sanity check for the forecasts, and methods with a RelMAE greater one are not reasonable for forecasting. The RelMAE can be defined as:

$$\text{RelMAE} = \frac{\text{MAE}}{\text{MAE}_B}. \tag{2}$$

### 4.2 *CWFR* function

As noted in Section 3, the errors in CWF are not symmetric. On the one hand, an under-provisioning error occurs when a user requires an increase in resources and they cannot be provided immediately. In this case, as a mean of prestige many cloud providers agree to pay back a percentage of the fault rate. With respect to the CPU usage, this error happens when the number of cores supplied to a given user is smaller than the number requested.

Let $y_t$ be the total number of cores required at time $t$ and $z_t$ be total number of cores supplied by the provider. Let us assume the period under evaluation is from time 1 up to $m$. Then the cost for under-provisioning, $C_{up}$ in that period is:

$$C_{up}(1,...,m) = \sum_{n=1}^{m} \begin{cases} y_i - z_i, & \text{if } y_i > z_i, \\ 0, & \text{otherwise.} \end{cases} \tag{3}$$

On the other hand, an over-provisioning happens when the number of immediately available cores, i.e., is active cores, is greater than the requested number. In this case, the cloud supplier incurs in exploitation costs, just like energy consumption, equipment degradation, and so forth. Following the notation above, we can define the cost for over-provisioning, $C_{op}$, as:

$$C_{op}(1,...,m) = \sum_{n=1}^{m} \begin{cases} 0, & \text{if } y_i > z_i, \\ z_i - y_i, & \text{otherwise.} \end{cases} \tag{4}$$

Now, after establishing the corresponding weight we we can add both terms together. Let $p$ be the unit penalty imposed when the number of cores is smaller than the requested one —for example, it could be a percentage (over 100% of the rate). Let $r$ be the rate for unit time usage, and let $e$ be the cost of having one core active —but not used— for one second. This would include energy consumption —for both operation and cooling, degradation cost, etc. Then the overall CWF cost function is defined as:

$$\mathit{CWFc}(1..m) = p \cdot r \cdot C_{up}(1..m) + e \cdot C_{op}(1..m) \tag{5}$$

*CWFc* is defined for the actual usage, but it can be readily adapted to obtain the performance of a forecaster f, $\mathit{CWFc}(f)$ by changing $y_i$ for the value predicted by the forecaster. Finally, we can compare the relative performance of the forecaster with the reactive approach (no forecasting) by computing the ratio:

$$\mathit{CWFR} = \frac{\mathit{CWFc}(f)}{\mathit{CWFc}} \tag{6}$$

This measure allows us to obtain a fast monetary cost of the forecasting errors. Let us assume a platform

datacenter of size $nc$, number of cores. The amount saved by using the forecaster f with respect to using just a reactive approach is:

$$MC = CWFR \cdot nc. \quad (7)$$

Similar ratio definitions associated to $C_{up}$ and $C_{op}$ can be straightforwardly defined, $C_{up}R$ and $C_{op}R$.

## 5 A Complete Forecasting Methodology

Once the problem is well understood and a convenient error measure is defined, we proceed to set the forecasting methodology. This section presents a thorough forecasting methodology, based on the state-of-the-art forecasting methodology to analyze and predict time series data described in Hyndman and Athanasopoulos [5]. We enhance and complete the methodology to include, e.g., non-linear and ML forecasting procedures and statistical tests for linearity to determine when to use these more complex procedures.

The general scheme of this process is depicted in Figure 2. In the following, we present the different steps being part of this methodology:


Fig. 2. Forecasting Methodology - General Scheme

1) Visualize the time series, and analyze Auto-Correlation Function (ACF) and Partial Auto-Correlation Function (PACF) plots (Section 5.2).
2) Perform a non-seasonal study, constructing ETS and ARIMA models as a first approach (Section 5.3), and using the obtained results to build other regression models.
3) Perform a similar study but focused on seasonality (Section 5.4).

So, several models are built and evaluated on a given dataset, to select and build the best final model.

Before we explain the aforementioned steps, we present the forecasting methods (Section 5.1) which, like the evaluation criteria previously seen (in Section 4), are used in this methodology. All of them are employed in both the non-seasonal and seasonal studies.

### 5.1 Forecasting models

In this section, we describe the different forecasting algorithms employed in this framework. Since there is no single universally valid model and the time series to be analyzed have different properties, we use a wide range of forecasting models to obtain the fittest model according to their features.

Firstly, we describe thoroughly the ARIMA and ETS methods, which can be considered the *de-facto* standard models in time series forecasting [27]. Besides these state-of-the-art models, we also use an additional set of general regression methods for comparison purposes.

#### 5.1.1 Time series models: ARIMA & ETS

##### *5.1.1.1 ARIMA*

Combining an ARMA model and differencing, we obtain the non-seasonal ARIMA model [28]. It is a workhorse in time series analysis and modelling. ARIMA models are denoted as ARIMA($p$,$d$,$q$), where $p$ is the number of lagged values considered in the autoregressive part, $q$ is the number of lagged values considered for the moving average part, and $d$ is the number of differences considered.

There is an established methodology to choose all possible parameters of an ARIMA model fully automatically. In R [29], a language and environment for statistical computing and graphics, this methodology, known as the Hyndman-Khandakar algorithm [5], is implemented in the *auto.arima* function from the forecast package [27].

##### *5.1.1.2 Exponential Smoothing*

The abbreviation ETS [5], which will be used in the following, stands both for exponential smoothing and for error, trend, and seasonality. ETS forecasts are computed as weighted averages of past observations. They can be applied to a wide set of series, so the key to properly do so is to recognise key components of the series (trend and sesonality) and how they are employed in the smoothing method (in an additive or multiplicative way).

Nowadays, there is a whole family of ETS models, with a solid theoretical foundation. The models can be distinguished by the type of error, trend, and seasonality they use. In total there are 15 models with different combinations of trend and seasonality. For more detailed descriptions, we refer to Goodwin [30].

In R, exponential smoothing is implemented in the function *ets* from the *forecast* package, which can be used to automatically fit models to the data provided. The smoothing parameters and initial conditions are optimized for maximum likelihood with a simplex optimizer. Then, the best model is chosen using AIC.

#### 5.1.2 Standard regression methods

Besides these state-of-the-art forecasting models, we also use a set of general regression models such as:

##### 5.1.2.1 *Linear auto-regression*

Linear Auto-Regression (AR) is based on considering each variable value as a linear combination of the previous ones [28], with the introduction of some noise. This method is considered a special case of ARIMA and is normally used as a benchmark method. We refer to this as an $AR(p)$ model.

##### 5.1.2.2 *Lasso*

Lasso regression as proposed by Efron *et al.* [31], is a well-known shrinkage method used to estimate coefficients in linear models, which pursues the removal of the maximum number of coefficients turning them zeros, as well as the minimization of the error.

##### 5.1.2.3 *Multi-adaptive regression splines*

Multi-Adaptive Regression Splines (MARS), as proposed by Friedman [32], performs a regression process on input data obtaining an approximative function. MARS uses some base functions, such as: constant function, maximum function, or product of the previous ones.

#### 5.1.3 Machine learning methods

##### 5.1.3.1 *Multi-layer perceptron with back-propagation*

Multi-Layer Perceptron (MLP) is a feedforward neural network formed by multiple layers, which allows it to solve non-linearly separable problems [33].

Learning occurs in the perceptron by updating connection weights after each instance has been processed, based on the output error compared to the real value. This is conducted by the backpropagation process: a generalization of the least squares algorithm in the linear perceptron.

##### 5.1.3.2 *Multi-layer perceptron trained with the BFGS algorithm*

A multi-layer perceptron trained with the BFGS algorithm [34] (we call this NNET in the following). The BFGS algorithm is a Quasi-Newton second-derivative line search family method and one of the most powerful methods to solve unconstrained optimization problem. The BFGS method approximates Newton's method, a class of hill-climbing optimization techniques that seeks a stationary point of a (preferably twice continuously differentiable) function.

##### 5.1.3.3 *Elman recurrent network*

Elman recurrent networks [35] have the same topology as the aforementioned perceptron-based methods, however, they introduce recurrence on their layers. They have a context layer available, which is a layer fed by the hidden layer output. The Elman network then stores these values and outputs them into the next run of the neural network. These values are sent, using a trainable weighted connection, back into the hidden layer. They are very useful in forecasting since they implement a limited short-term memory.

##### 5.1.3.4 *Support vector regression*

Support Vector Machines (SVMs) are supervised learning models, based on Statistical Learning Theory, with associated learning algorithms that analyze data and recognize patterns, used for classification and regression tasks. Their main objective is to solve a quadratic optimization problem finding a maximum separability hyperplane in a higher dimensional space. In our experiments, we use $\epsilon$-SVR [36], which inherits an SVR standard implementation where the cost function ignores any data according to a given threshold $\epsilon$.

### 5.2 Time series analysis

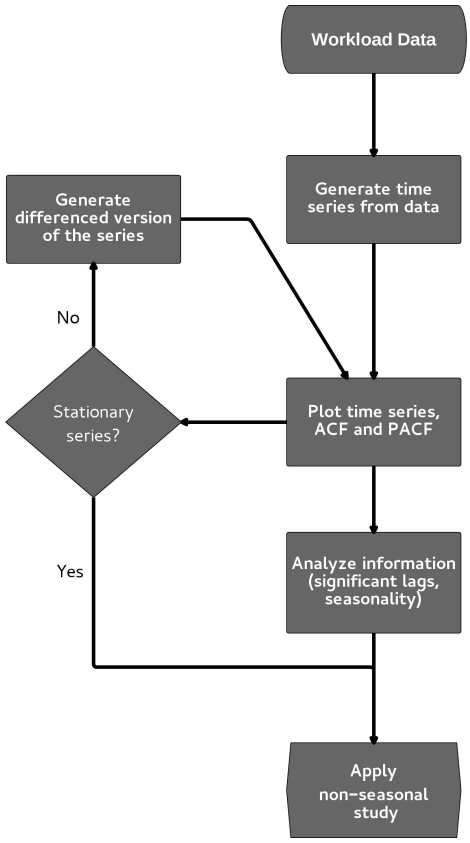


Fig. 3. Forecasting Methodology - Analytical Process

In time series forecasting, it is important to analyze the data before applying any method. There is a great amount of information that can be used to improve the forecasting performance. By plotting a series and its ACF, we can look for significant lags to model our series and extract other information from patterns like stationarity, seasonality and trends. We can use this information to model the series accordingly. In this section, we present the first step of the methodology: an analytical process which aims to analyze the series to extract relevant information to the modeling stage.

This analytical process is described as follows: first of all, we plot all time series in consideration and their corresponding ACF and PACF to visually examine them,

looking for significant lags. The significant lags found will be used to configure the forecasting methods. If the plots do not show significant lags, we use a default maximal lag value to explore the data. A frequent default value is 5 (e.g., the *auto.arima* function uses this as a default), so that we stick to this default value in our non-seasonal studies. For seasonal studies, we can employ typical seasonal lags, e.g., lags 12 or 24 for hourly data, or 30 or 60 for minutely data. The forecasting methods test if these potentially interesting lags are reasonable, and therewith, effective for our forecasting problem. A scheme of the whole process is depicted in Figure 3.

On the other hand, if we notice in the ACF that the series are non-stationary (have an integrated behavior), differences on series can be used. We construct models that use differencing by applying the following procedure: firstly, we create the differenced time series applying the *diff* function (also in R) on the original time series. After that, we construct complementary models using both the original and the differenced data, so that we obtain as forecasts the difference of each value from the last known value. To transform these values to the input domain, we add to each differenced value the corresponding $t-1$ value in the original time series, thus obtaining the correct forecasts. We repeat the previous process over the differenced version to find interesting lag values.

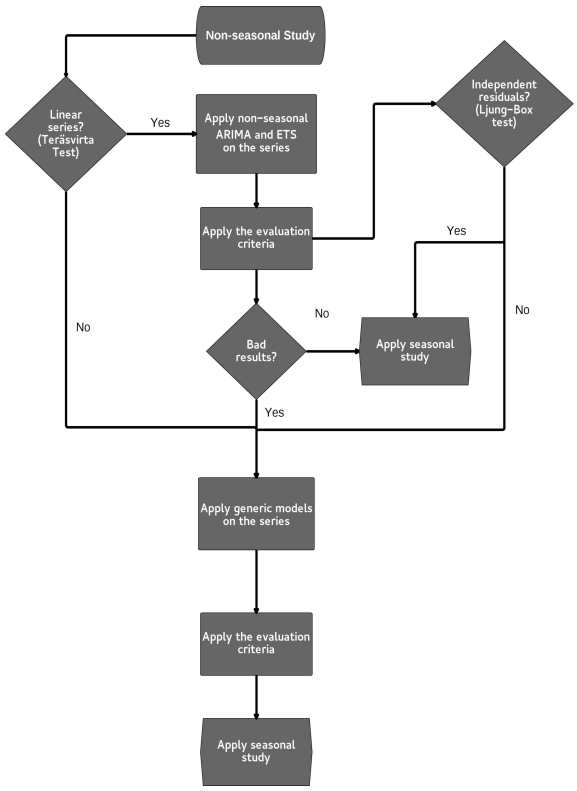


Fig. 4. Forecasting Methodology - Non-seasonal Study

### 5.3 Non-seasonal study

After analyzing the series, we use ETS and ARIMA to construct models without seasonality as a first approach. The entire process is represented in Figure 4.

As stated before, ARIMA parameter values can be set automatically through *auto.arima* in R. This process is based on the AICc measure. It is important to note that non-seasonal models are incorporated at this stage in the selection process. ETS has a similar process to choose among models, which we also restrict to non-seasonal models here. After that, we apply the evaluation criteria to assess the quality of these models.

Furthermore, it is worthwhile to evaluate the independence of the residuals. A common test for residual diagnostics is the Box-Pierce test [37], and based on it the Ljung-Box test [38]. Both of them test if the first $h$ auto-correlations are significantly different from a white noise process.

If there is still auto-correlation left in the residuals, it is highly recommended to continue the exploration of models (for example, with non-linear alternatives).

We then check the linearity of the original time series, performing a linearity test over all the series. This procedure is advisable as fitting flexible non-linear models may lead to overfitting if there is no non-linearity present or if it is not strong [39].

We use the function *terasvirta.test* from the package *tseries* in R, which is an implementation of a well-known non-linearity test of Teräsvirta [39]. The test is based on a single hidden-layer feedforward neural network. It performs a regression of a Taylor series expansion of the non-linear activation function of the neural network to the data. Then, it tests the null hypothesis of linearity of the data by testing if the coefficients of the non-linear terms of this expansion are zero.

A key advantage of the test is that it has power against many types of nonlinearity [39], not only against the type of nonlinearity modeled by the neural network, so that it can be seen as a general test for non-linearity.

If the performance results are then unsatisfactory, we can try to improve the results obtained so far using an additional set of methods (general regression methods). Thereby, we can use the number of lagged values derived from the ARIMA results, or the significant (non-seasonal) lags extracted from the analysis to create an embedding scheme of lags; and afterwards, to build the models. Likewise, we can use the degree of differencing to create differenced versions of the models.

Finally, we apply again the evaluation criteria to assess the quality of the new models.

### 5.4 Seasonal study

As a daily seasonality may be present in the series, we perform additional experiments around seasonality. We follow the same study used in the above part (Figure 2), but focusing on seasonality and without applying a

further study in case of unsatisfactory results. Thereby, we allow for choosing seasonal models, if appropriate.

The automatic building procedure for ETS does not choose seasonal models. However, ARIMA can choose new seasonal and differenced models. After modeling, we use again the Ljung-Box test to evaluate the independence of residuals, and we compare the new models with the previous ones using the evaluation criteria. If the results are unsatisfactory or the test rejects the null hypothesis, we try to improve the results with the set of general methods. Likewise, if the original series are non-stationary, we also perform the seasonal process on the differenced version of the series.

In this way, we can leverage the information from ARIMA, or the significant seasonal lags (e.g., 24-lag value for hourly data) obtained from the analysis to create new models with a different embedding of lags.

At last, we repeat the calculation of evaluation criteria over the new models to compare their performances with those obtained from previous steps.

## 6 Experimental framework

This section describes the experimental framework used in our experiments to demonstrate the feasibility of our proposal. First of all, we present the data used in our experiments as well as the preprocessing applied. Afterwards, we describe the experimental setup, including the parameters for the used methods.

### 6.1 Dataset description

To evaluate the proposal we have gathered datasets from four different cloud platforms. These are from Google clusters [40], LANL Origin 2000 Cluster (Nirvana), University Gaia Cluster of Luxemburg, and Sharnet Whale, [41]. Summary information of the datasets is shown in Table 1. The datasets are original logs of these clusters, which we have preprocessed to extract the data relevant for the CPU usage, namely core-second values per minute.

Log files are composed of historical resource usage data. Measurements are usually taken at one second intervals.

In the case of the Google dataset, the task resource usage table contains the following fields: start and end time, job ID, task index, machine ID CPU usage (mean), memory usage, assigned memory, unmapped page cache memory usage, page cache memory usage, maximum memory usage, disk I/O time (mean), local disk space used (mean), CPU usage (max), disk IO time (max), Cycles Per Instruction (CPI), Memory Accesses per Instruction (MAI), sampling rate and aggregation type. The other three datasets have a similar log format.

As noted in Section 3, we focus on CPU usage. Data are processed so that the total number of cores used by all the tasks at a given one-second period is summed. The resulting series represents the workload of the system per minute. Finally, the data is normalized to zero mean, variance one before applying methods, which has practical considerations as many of the methods only work well on normalized data. Predictions and error information are then de-normalized and analyzed in the input domain.

### 6.2 Experimental setup

All experiments are performed using R implementations of the algorithms presented. Most of the methods used, except ARIMA and ETS that fit their parameters in an automatic way, have meta-parameters to set. We define parameter grids for each method. For the ANN models, we consider as parameters the size of the hidden layer and the weight decay rate with all combinations of size={3, 5, 9, 15} and decay={0.00316, 0.0147, 0.1}. SVR has the parameters cost, gamma, and epsilon. The parameter grid for SVR contains the following configuration: cost={10, 100}, gamma={0.001, 0.01, 0.2}, and epsilon=0.1. LASSO only has one free parameter (fraction), which is chosen from fraction={0.1, 0.36, 0.63, 0.9}.

TABLE 2
Parameters chosen using parameter grids

| Method | Parameters |
|---|---|
| SVR | cost = 100, gamma = 0.001, epsilon = 0.1 |
| MLP | size = 5, decay = 0.00316, maxit = 1,000 |
| NNET | size = 3, decay = 0.1, maxit = 1,000 |
| ELMAN | size = 5, decay = 0.00316, maxit = 1,000 |
| LASSO | fraction = 0.9 |
| MARS | – |
| AR | – |

To choose the most suitable parameter combination for each method, we perform an initial study with a smaller, comparable dataset from a similar domain. For each method the parameter combination yielding the lowest cumulative fitting error on all evaluation sets is chosen. Furthermore, we also apply all forecasting methods with all parameter combinations defined in the parameter grids to train on all time series, to verify on the training set that the initially chosen parameters are reasonable choices. In this way, we avoid over-fitting without loosing data for an additional validation set.

The models with the chosen parameter configurations are executed over the respective test data. See Table 2 for the parameter configurations.

For every dataset the first 80% of data are used for model training, and the remaining 20% compose the test set. In addition, regarding the forecasting horizon, since the most common usage scenario is one-step-ahead prediction, every model has been designed for forecasting with this horizon.

## 7 Experimental analysis

This section presents the experimental results as well as their analysis. Due to space constraints, we show

TABLE 1
Summary information of the used datasets

| Dataset name | Google | LANL O2K 1999 | UniLu Gaia 2014 | Sharnet Whale |
|---|---|---|---|---|
| Dataset file name | clusterdata-2011-2 | LANL-O2K-1999-2.swf | UniLu-Gaia-2014-2.swf | SHARCNET-Whale-2005-2.swf |
| Number of samples | 42761 | 198352 | 128137 | 524608 |
| Sampling interval | 1 minute | 1 minute | 1 minute | 1 minute |
| Sampling size | 40GB | 2.1MB | 837KB | 8.06MB |
| Length (s) | 2565600 (29.69 days) | 11901074 (137.74 days) | 7694207 (89.95 days) | 17529163 (202.88 days) |

the most illustrative results and summaries of all other results[1].

### 7.1 Time series analysis

Figure 5 depicts the complete workload system series for the Google dataset hourly CPU load [2]. In the figure, we see an oscillating pattern with a period length of approx. one day, which could indicate a seasonal pattern of daily fluctuation. However, in the ACF and PACF plots this pattern is not reflected, which suggests that the cycles change in length. In the PACF we see that lags 1, 2, and 5 are significant. In the differenced version, we get significant lags up to 12 and 16, respectively, in the ACF and PACF plots.

The CPU 5-minute series is similar to the hourly version; it shows an integrated behavior. Significant lags are only present up to lag 10. Indeed, we do not expect the 5-minute series to show a seasonality at lag 24.

To summarize, we observe in the plots that seasonal patterns may be present in the time series, and we use a lag value of 24 to evaluate daily patterns in the seasonal study. ARIMA and ETS assess if there is daily seasonality in the series and therewith if this lag choice is reasonable. We also use a value of 5, as this is a maximal lag choice commonly used in time series forecasting (as already discussed in Section 5.2) and as most of the series' autocorrelation can be covered using the first 5 lags.

With respect to the other datasets, their plots display shapes similar to the Google case. In general, the average time series value is much smaller and peaks are more scattered and higher, which affects negatively its predictability. The conclusion is again that no seasonality pattern is observed.

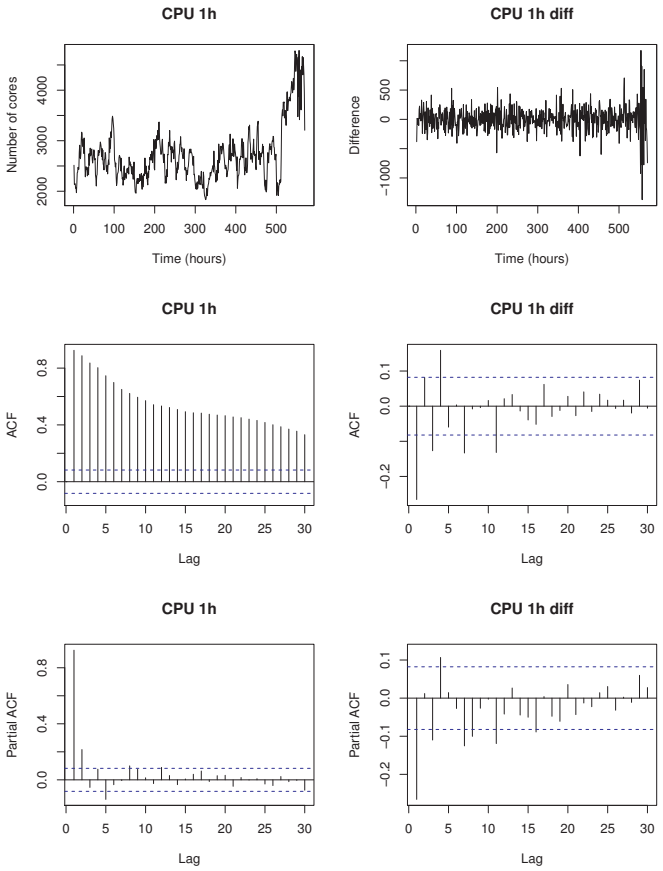


Fig. 5. Google CPU 1h time series. The first row depicts the original workload system series ("tserie" label) and its differenced version ("diff" label). Here, the x-axis represents the time index (1h), and the y-axis represents the CPU rate prediction. The second and third rows show the corresponding ACF and PACF plots.

### 7.2 Non-seasonal study

After having analyzed the series, we use ETS and ARIMA to construct models without seasonality as a first approach and use the Ljung-Box [38] test to evaluate the independence of the residuals and therefore check the quality of the models. Then, we check the linearity of the original time series, performing a Teräsvirta [39] test over all the time series.

After that, we try to improve the results obtained so far, using an additional set of general forecasting models. We use the number of lagged values derived from the ARIMA results and the Ljung-Box test on the residuals derived.

As stated before, ARIMA is a method which does not need any parameter configuration in its automated version (*auto.arima*) in R. ETS has a similar process to choose among models which we also restrict to non-seasonal models here. Table 3 shows non-seasonal models which have been chosen using the automated selection process.

Regarding the models obtained, ETS does not choose models with trend in any time series, and ARIMA chooses generally models with the maximal lag 5 (the

1. Detailed results as well as additional information is available at the paper webpage: http://dicits.ugr.es/papers/CWF.

2. The full series plots, as well as the full ACF and PACF plots were omitted here due to space constraints.

maximum is a parameter of the method) and with first differencing. Table 3 shows the parameters used to build the models that we use in the following sections.

TABLE 3
ARIMA/ETS results without seasonality

| | Google | Lanl | UniLu | Sharnet |
|---|---|---|---|---|
| | | **ARIMA** | | |
| | CPU 5min | CPU 5min | CPU 5min | CPU 5min |
| RelMAE | **0.972** | **0.987** | 1.259 | **0.952** |
| sMAPE | **3.150** | 13.473 | 22.900 | **3.961** |
| Models | ARIMA(4,1,4) | ARIMA(1,1,2) | ARIMA(5,1,2) | ARIMA(5,1,3) |
| | | **ETS** | | |
| RelMAE | 1.000 | 1.000 | **1.219** | 0.992 |
| sMAPE | 3.221 | **10.017** | **21.925** | 4.159 |
| Model | ETS(M,N,N) | ETS(A,N,N) | ETS(A,Ad,N) | ETS(A,Ad,N) |

Table 3 displays the RelMAE and sMAPE values for ARIMA and ETS. The RelMAE indicates that ARIMA models outperform the naïve forecaster for the Google, Lanl and Sharnet datasets. In addition, ARIMA provides a lower RelMAE and a lower sMAPE than ETS in the Google and Sharnet cases. ETS values indicate that this technique is not suitable for the CWF series, and can be safely discarded. As noted above the peculiar peak distribution of UniLu hinders ARIMA and ETS from performing well.

As regards the Ljung-Box test, results are shown in Table 4. We see that both for ARIMA and ETS, there is still auto-correlation left in the residuals. Thereby, it is highly recommended to continue the exploration of models. On the one hand, we have to check linearity of the series, and on the other hand, the lag structure needs to be expanded.

TABLE 4
Ljung-Box test

| | Google | Lanl | UniLu | Sharnet |
|---|---|---|---|---|
| | | $p$-value | | |
| | | **ARIMA** | | |
| CPU 5min | 0.0002893 | $2.2 \cdot 10^{-16}$ | $2.2 \cdot 10^{-16}$ | $2.2 \cdot 10^{-16}$ |
| Models | ARIMA(4,1,4) | ARIMA(1,1,2) | ARIMA(5,1,2) | ARIMA(5,1,3) |
| | | **ETS** | | |
| CPU 5min | $2.2 \cdot 10^{-16}$ | $2.2 \cdot 10^{-16}$ | $2.2 \cdot 10^{-16}$ | $2.2 \cdot 10^{-16}$ |
| Model | ETS(M,N,N) | ETS(A,N,N) | ETS(A,Ad,N) | ETS(A,Ad,N) |

So, we use the Teräsvirta test to assess the linearity of the original time series. As we can see in Table 5, the null hypothesis is rejected in all cases. Thus, we can state that it is highly desirable to use a complementary set of general methods.

TABLE 5
Teräsvirta test

| | Google | Lanl | UniLu | Sharnet |
|---|---|---|---|---|
| | | $p$-value | | |
| | | **ARIMA** | | |
| CPU 5min | $2.2 \cdot 10^{-16}$ | $2.2 \cdot 10^{-16}$ | $2.2 \cdot 10^{-16}$ | $2.2 \cdot 10^{-16}$ |

Using 5 lagged values and differenced time series to train complementary regression methods, we obtain the results shown in Table 6. To construct models that use differencing, we have followed the procedure described in Section 5.2.

TABLE 6
Non-seasonal results for general methods (model order 5)

| **Model** | Google | Lanl | UniLu | Sharnet |
|---|---|---|---|---|
| | | **RelMAE** | | |
| AR | 0.983 | 0.983 | 1.213 | **0.937** |
| LASSO | 0.987 | 0.982 | 1.194 | 0.961 |
| MARS | 0.979 | 0.978 | 1.167 | 0.972 |
| NNET | 0.973 | **0.969** | **1.109** | 0.973 |
| ELMAN | 1.308 | 0.997 | 1.288 | 1.560 |
| MLP | 0.977 | 0.983 | 1.122 | 0.956 |
| SVR | **0.969** | 1.001 | 1.447 | 1.762 |
| | | **sMAPE** | | |
| AR | 3.164 | **12.720** | 22.998 | 4.532 |
| LASSO | 3.177 | 13.715 | 22.617 | 4.463 |
| MARS | 3.157 | 13.211 | 21.264 | 4.218 |
| NNET | 3.131 | 13.001 | 21.170 | **4.188** |
| ELMAN | 4.269 | 13.383 | 27.393 | 10.613 |
| MLP | 3.145 | 13.238 | **20.929** | 4.463 |
| SVR | **3.125** | 14.379 | 34.409 | 10.482 |

In Table 6, we can see how there is no method that always outperforms the others. We can see how methods that offered very good results both in terms of RelMAE and sMAPE on the Google dataset, namely SVR, show the worst performance in almost all cases.

### 7.3 *CWFR* cost

Table 7 displays under-provisioning and over-provisioning costs, $C_{up}$ (upper part of the table) and $C_{op}$ (lower part of the table) for each model expressed as ratio with respect to reactive behavior without forecasting. The results are clear: for all the datasets an improvement in $C_{up}$ is reached by using a forecaster. Even the simple naïve forecast is better than a primitive reactive approach. There is improvement in both, $C_{up}$ and $C_{op}$, with a better achievement for $C_{up}$, whose average is around 30%. This is even more interesting since the component of the cost with the higher economic weight is $C_{up}$.

### 7.4 Seasonal study

As a daily seasonality may be present in the series, we perform additional experiments with the hourly series. We follow the same process used in the previous part, but focus on seasonality.

A daily seasonality in hourly series means a lag difference of 24. So, we allow for choosing seasonal models, if appropriate. The automatic building procedure for ETS does not choose seasonal models in our cases, so that Table 8 shows only the newly chosen models for ARIMA. We can observe that automatic ARIMA now chooses seasonal models for three series, as well as first differencing for the "CPU 1h" series.

TABLE 7
Under-provisioning and over-provisioning costs (model order 5) 5 min

| **Model** | Google | Lanl | UniLu | Sharnet |
|---|---|---|---|---|
| **Under-provisioning cost ratio $C_{up}R$** | | | | |
| AR | 89.117 | 83.997 | 92.639 | 80.131 |
| LASSO | 89.095 | 83.690 | 92.166 | 83.595 |
| MARS | 88.134 | 83.554 | 94.129 | 84.802 |
| NNET | 87.675 | 83.684 | 90.959 | 85.154 |
| ELMAN | ***73.482*** | 92.518 | 76.329 | 103.212 |
| MLP | 87.820 | 95.905 | 92.774 | 81.603 |
| SVR | 90.898 | 81.603 | ***71.886*** | 128.993 |
| ARIMA | 83.406 | 82.664 | 98.189 | 84.033 |
| ETS | 85.582 | ***74.387*** | 106.410 | 90.916 |
| Naïve | 85.584 | 74.398 | 77.244 | ***60.225*** |
| No prediction | 100 | 100 | 100 | 100 |
| **Over-provisioning cost ratio $C_{op}R$** | | | | |
| AR | 80.360 | 82.781 | 101.947 | 85.283 |
| LASSO | ***80.225*** | 101.752 | 98.892 | 84.828 |
| MARS | 81.374 | 82.697 | 99.680 | 86.105 |
| NNET | 87.431 | 81.865 | 93.482 | 89.029 |
| ELMAN | 120.577 | 77.884 | 129.72 | 119.784 |
| MLP | 85.131 | 75.705 | 89.573 | 94.336 |
| SVR | 84.931 | 86.197 | 138.591 | 93.520 |
| ARIMA | 86.613 | 82.767 | 99.041 | 84.063 |
| ETS | 86.300 | 75.427 | 106.636 | 90.947 |
| Naïve | 86.373 | ***74.410*** | ***77.297*** | ***60.247*** |
| No prediction | 100 | 100 | 100 | 100 |

TABLE 8
ARIMA results with seasonality

| | **ARIMA** | | | |
|---|---|---|---|---|
| | Google | Lanl | UniLu | Sharnet |
| | CPU 1h | CPU 1h | CPU 1h | CPU 1h |
| RelMAE | 0.921 | 1.062 | 1.134 | 1.021 |
| sMAPE | 6.315 | 30.222 | 44.390 | 30.070 |
| Model | ARIMA(5,1,4)(1,0,0)[24] | ARIMA(4,1,4)(0,0,2)[24] | ARIMA(1,1,1) | ARIMA(2,1,1)(2,0,0)[24] |

In Table 8, we see how LANL and Sharnet show better RelMAE and sMAPE in the non-seasonal case, on the other hand UniLu has a better RelMAE in the non-seasonal case although in the seasonal case UniLu has a higher sMAPE, as occurs in the case of the Google data.

TABLE 9
Ljung-Box test

| | $p$-value | | | |
|---|---|---|---|---|
| | **ARIMA** | | | |
| | Google | Lanl | UniLu | Sharnet |
| CPU 1h | $2.2 \cdot 10^{-16}$ | $2.28 \cdot 10^{-11}$ | 0.2134 | $3.13 \cdot 10^{-06}$ |

Furthermore, Ljung-Box test results for the residuals are shown in Table 9. We see that there is a lot of auto-correlation left which the models do not adequately exploit. These results show that there seems not to be a clear 24-hour seasonality present in the data. However, as it was shown before a lag structure of 5 may not be sufficient as there is auto-correlation also in higher lags. Thereby, we decided to use the general models in a further experiment with a lag structure of 24. The results for this experiment show a slight improvement in terms of RelMAE and sMAPE of the models for Google and Lanl with respect to the corresponding non-seasonal results displayed in Table 6.

Next, we evaluate the impact of seasonal models with increased lag terms. Table 10 presents the cost of under-provisioning and over-provisioning in terms of ratios for these models. Again, a clear reduction in both $C_{up}$ and $C_{op}$ is reached when using forecasting models with respect to the reactive approach. The size of reduction, however, is smaller than in the non-seasonal model cases.

TABLE 10
Under-provisioning and over-provisioning costs (model order 24) 5 min

| **Model** | Google | Lanl | UniLu | Sharnet |
|---|---|---|---|---|
| **Under-provisioning cost ratio $C_{up}R$** | | | | |
| AR | 93.206 | 95.340 | 99.436 | 98.775 |
| LASSO | 92.972 | 95.288 | 99.072 | 98.755 |
| MARS | 92.996 | 95.371 | 101.005 | 98.425 |
| NNET | 91.224 | 95.675 | 98.352 | 98.772 |
| ELMAN | ***89.603*** | 100.416 | ***72.750*** | 99.767 |
| MLP | 91.022 | 93.700 | 93.237 | 99.896 |
| SVR | 94.996 | 95.301 | 90.714 | 105.016 |
| ARIMA | 90.172 | ***84.732*** | 89.905 | 96.712 |
| ETS | 96.305 | 96.190 | 94.224 | 98.193 |
| Naïve | 98.123 | 98.001 | 98.550 | ***95.516*** |
| No prediction | 100 | 100 | 100 | 100 |
| **Over-provisioning cost ratio $C_{op}R$** | | | | |
| AR | 91.485 | 94.932 | 101.475 | 99.014 |
| LASSO | 91.593 | 94.497 | 101.177 | 99.006 |
| MARS | 92.583 | 94.957 | 101.772 | 98.844 |
| NNET | 93.722 | 95.246 | 99.416 | 98.988 |
| ELMAN | 112.496 | 92.982 | 154.536 | 114.268 |
| MLP | 99.074 | 95.782 | 106.863 | 98.343 |
| SVR | 94.008 | 97.732 | 115.831 | 99.167 |
| ARIMA | ***90.546*** | ***84.922*** | ***89.931*** | 96.730 |
| ETS | 96.834 | 94.242 | 94.227 | 98.216 |
| Naïve | 98.796 | 98.028 | 98.715 | ***95.535*** |
| No prediction | 100 | 100 | 100 | 100 |

In conclusion, the application of the proposed methodology has led us to gain some useful insight regarding CWF time series properties, i.e., no seasonality, nor trend is found, and they are non-linear. This helps to discard some models and focus on others. In addition, by using the forecasting models developed here, a significant reduction in under-provisioning and over-provisiong costs can be achieved.

### 7.5 Forecasting for longer time horizons

While the concern of this paper is providing an effective method for the short-term workload forecasting (so-called *operational horizon*), at the request of one of the referees, we have performed some experiments for longer horizons. Hence, we have defined similar prediction problems considering 10, 15, 30, 45 and 60 minutes ahead and built the corresponding models. Due to space constraints, we cannot include complete empirical results, just some conclusions. The general conclusion is that, as

expected, the cost increases as the time horizon grows longer. Although not in every case, because the average cost at 15 minutes is smaller than at 10 minutes. The averages cost increase percentage, expressed in % with respect to the cost obtained at a 5-minute, are 43.211, 44.083, 20.077, 15.597, and 19.664, for 60, 45, 30, 15, and 10 minutes, respectively.

## 8 Conclusions

In this paper, we have discussed the concept of elasticity in Cloud Computing, as well as the relevance of having a proper forecasting module, as part of a complete elastic system. We have seen that workload forecasting in cloud platforms is a problem that needs a preliminary stage to analyze and model properly the properties of the time series.

As there is no general framework or methodology in the literature that addresses the CWF problem analytically from a time series point of view, we propose a combination of tools based on a state-of-the-art forecasting methodology, which we have enhanced and completed. This methodology comprises a deeper study through the analysis and the modeling of workload series to better understand the intrinsic properties of the series.

Since the error in CWF is clearly asymmetric we have proposed a new measure to assess the cost of under-provisioning and over-provisioning. These measures allow for a better visualization of the performance of the elastic provisioning module of cloud platforms.

The effectiveness of this approach has been evaluated through its application to workloads of four different cloud datacenters. The insights gained from the empirical results have allowed to find the best model architecture, so that the best forecasts from the viewpoint of the current state of the art in time series forecasting have been achieved.

Finally, we found that the use of forecasting algorithms leads to significant cost reductions in both under-provisioning and over-provisioning, which is of great importance to the industry that provides cloud computing. A reduction around 30% was observed in our study cases.

## Acknowledgments

The authors would like to thank the associate editor and anonymous reviewers for their insightful comments, which have led to a much improved version of the paper.

This work was partially supported by the Spanish Ministry of Science and Technology under projects TIN2011-28488, TIN2013-47210, and the Andalusian Research Plans P11-TIC-7765, P10-TIC-6858 and P12-TIC-2958.

## References

[1] R. Buyya, J. Broberg, and A. M. Goscinski, *Cloud Computing Principles and Paradigms*. Wiley Publishing, 2011.

[2] Gartner, "Gartner's 2013 CIO agenda report," 2013. [Online]. Available: http://www.gartner.com/imagesrv/cio/pdf/cio_agenda_insights2013.pdf

[3] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, and I. Brandic, "Cloud Computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility." *Future Generation Comp. Syst.*, vol. 25, no. 6, pp. 599–616, 2009.

[4] A. Halavais, "The Slashdot Effect: analysis of a large-scale public conversation on the World Wide Web," Ph.D. dissertation, University of Washington, 2001.

[5] R. Hyndman and G. Athanasopoulos, "Forecasting: Principles and practice," 2013. [Online]. Available: http://otexts.org/fpp/

[6] S. Makridakis and M. Hibon, "The M3-competition: Results, conclusions and implications," *International Journal of Forecasting*, vol. 16, no. 4, pp. 451–476, 2000.

[7] S. Crone, M. Hibon, and K. Nikolopoulos, "Advances in forecasting with neural networks? empirical evidence from the NN3 competition on time series prediction," *International Journal of Forecasting*, vol. 27, no. 3, pp. 635–660, 2011.

[8] K. Konstanteli, T. Cucinotta, K. Psychas, and T. Varvarigou, "Elastic admission control for federated cloud services," *IEEE Transactions on Cloud Computing*, vol. 2, no. 3, pp. 348–361, 2014.

[9] S. Tang, B.-S. Lee, and B. He, "DynamicMR: A dynamic slot allocation optimization framework for mapreduce clusters," *IEEE Transactions on Cloud Computing*, vol. 2, no. 3, pp. 333–347, 2014.

[10] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, "A view of Cloud Computing," *Commun. ACM*, vol. 53, no. 4, pp. 50–58, 2010.

[11] N. Roy, A. Dubey, and A. Gokhale, "Efficient autoscaling in the Cloud using predictive models for workload forecasting," in *Proceedings of the 2011 IEEE 4th International Conference on Cloud Computing*, ser. CLOUD '11, 2011, pp. 500–507.

[12] A. Li, X. Yang, S. Kandula, and M. Zhang, "CloudCmp: comparing public cloud providers," in *Proceedings of the 10th ACM SIGCOMM conference on Internet measurement*, ser. IMC '10, 2010, pp. 1–14.

[13] S. Ostermann, A. Iosup, N. Yigitbasi, R. Prodan, T. Fahringer, and D. Epema, *A Performance Analysis of EC2 Cloud Computing Services for Scientific Computing*. Springer Berlin Heidelberg, 2010, Online ISBN 978-3-642-12636-9.

[14] M. Mao and M. Humphrey, "A performance study on the vm startup time in the cloud," *Cloud Computing (CLOUD), 2012 IEEE 5th International Conference on*, pp. 423–430, 2012.

[15] G. Galante and L. C. E. d. Bona, "A survey on cloud computing elasticity," in *Proceedings of the 2012 IEEE/ACM Fifth International Conference on Utility and Cloud Computing*, ser. UCC '12, 2012, pp. 263–270.

[16] W. Dawoud, I. Takouna, and C. Meinel, "Elastic VM for cloud resources provisioning optimization," in *ACC (1)*, ser. Communications in Computer and Information Science, A. Abraham, J. L. Mauri, J. F. Buford, J. Suzuki, and S. M. Thampi, Eds., vol. 190, 2011, pp. 431–445.

[17] W. Iqbal, M. N. Dailey, D. Carrera, and P. Janecek, "Adaptive resource provisioning for read intensive multi-tier applications in the Cloud," *Future Generation Computer Systems*, vol. 27, no. 6, pp. 871 – 879, 2011.

[18] J. M. Tirado, D. Higuero, F. Isaila, and J. Carretero, "Predictive data grouping and placement for cloud-based elastic server infrastructures," in *Proceedings of the 2011 11th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, ser. CCGRID '11, 2011, pp. 285–294.

[19] G. Reig, J. Alonso, and J. Guitart, "Prediction of job resource requirements for deadline schedulers to manage high-level slas on the Cloud," in *Proceedings of the 2010 Ninth IEEE International Symposium on Network Computing and Applications*, ser. NCA '10, 2010, pp. 162–167.

[20] L. R. Moore, K. Bean, and T. Ellahi, "Transforming reactive auto-scaling into proactive auto-scaling," in *Proceedings of the 3rd International Workshop on Cloud Data and Platforms*, ser. CloudDP '13, 2013, pp. 7–12.

[21] M. Imam, S. Miskhat, R. Rahman, and M. Amin, "Neural network and regression based processor load prediction for efficient scaling of grid and cloud resources," in *Computer and Information Technology (ICCIT), 2011 14th International Conference on*, 2011, pp. 333–338.

[22] S. Islam, J. Keung, K. Lee, and A. Liu, "Empirical prediction models for adaptive resource provisioning in the Cloud," *Future Gener. Comput. Syst.*, vol. 28, no. 1, pp. 155–162, 2012.
[23] Z. Gong, X. Gu, and J. Wilkes, "PRESS: predictive elastic resource scaling for cloud systems," in *CNSM*, 2010, pp. 9–16.
[24] Z. Shen, S. Subbiah, X. Gu, and J. Wilkes, "CloudScale: elastic resource scaling for multi-tenant cloud systems," in *Proceedings of the 2nd ACM Symposium on Cloud Computing*, ser. SOCC '11, 2011, pp. 5:1–5:14.
[25] U. Sharma, P. Shenoy, S. Sahu, and A. Shaikh, "A cost-aware elasticity provisioning system for the Cloud," in *Proceedings of the 2011 31st International Conference on Distributed Computing Systems*, ser. ICDCS '11, 2011, pp. 559–570.
[26] R. Hyndman and A. Koehler, "Another look at measures of forecast accuracy," *International Journal of Forecasting*, vol. 22, no. 4, pp. 679–688, 2006.
[27] R. Hyndman and Y. Khandakar, "Automatic time series forecasting: The forecast package for R," *Journal of Statistical Software*, vol. 27, no. 3, pp. 1–22, 2008.
[28] G. Box and G. Jenkins, *Time series analysis: Forecasting and control*. Holden-Day, 1970.
[29] R Development Core Team, *R: A Language and Environment for Statistical Computing*, R Foundation for Statistical Computing, Vienna, Austria, 2009. [Online]. Available: http://www.R-project.org
[30] P. Goodwin, "The Holt-Winters approach to exponential smoothing: 50 years old and going strong," *Foresight: The International Journal of Applied Forecasting*, vol. 19, pp. 30–33, 2010.
[31] B. e. Efron, "Least angle regression," *Annals of Statistics*, vol. 32, no. 2, pp. 407–499, 2004.
[32] J. H. Friedman, "Multivariate adaptive regression splines," *The Annals of Statistics*, vol. 19, pp. 1–67, 1991.
[33] C. Bergmeir and J. M. Beníez, "Neural networks in R using the Stuttgart Neural Network Simulator: RSNNS," *Journal of Statistical Software*, vol. 46, no. 7, pp. 1–26, 2012.
[34] W. N. Venables and B. D. Ripley, *Modern Applied Statistics with S*. Springer, 2002.
[35] J. L. Elman, "Finding structure in time," *COGNITIVE SCIENCE*, vol. 14, no. 2, pp. 179–211, 1990.
[36] C.-C. Chang and C.-J. Lin, *LIBSVM: a library for support vector machines*, 2001, software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.
[37] S. Makridakis, S. Wheelwright, and R. Hyndman, *Forecasting: Methods and Applications*. Wiley, 1997, ISBN 0471532339.
[38] G. M. Ljung and G. E. P. Box, "On a measure of lack of fit in time series models," *Biometrika*, pp. 297–303, 1978.
[39] T. Teräsvirta, C.-F. Lin, and C. W. J. Granger, "Power of the neural network linearity test," *Journal of Time Series Analysis*, vol. 14, no. 2, pp. 209–220, 1993.
[40] Google, "Google Cluster Data: traces of Google workloads," 2013. [Online]. Available: http://code.google.com/p/googleclusterdata/wiki/ClusterData2011_1/
[41] T. R. The Hebrew University of Jerusalem, S. B. S. of Computer Science, and Engineering, "Logs of real parallel workloads from production systems," 2015. [Online]. Available: http://www.cs.huji.ac.il/labs/parallel/workload/logs.html

**Francisco Javier Baldán** Telecommunications engineer in 2014 from the University of Granada, Spain. He received the Official Master degree in Data Science and Computer Engineering in 2015 from the University of Granada, Spain. He is currently a member of the Department of Computer Science and Artificial Intelligence, University of Granada, Spain. His research interests include time series data mining, big data and cloud computing with parallel and distributed computing.

**Sergio Ramírez-Gallego** received the M.Sc. degree in Computer Science in 2012 from the University of Jaén, Spain. He is currently a Ph.D. student in the Department of Computer Science and Artificial Intelligence, University of Granada, Spain. His research interests include data mining, cloud computing, big data and parallel and distributed computing.

**Christoph Bergmeir** received the M.Sc. degree in Computer Science from the University of Ulm, Germany, in 2008, and the Ph.D. degree from the University of Granada, Spain, in 2013. He is currently working at the Faculty of Information Technology, Monash University, Melbourne, Australia, as a Research Fellow (Applied Machine Learning). He has published in journals such as IEEE Transactions on Neural Networks and Learning Systems, Journal of Statistical Software, Computer Methods and Programs in Biomedicine, and Information Sciences.

**Francisco Herrera** received his M.Sc. in Mathematics in 1988 and Ph.D. in Mathematics in 1991, both from the University of Granada, Spain. He is currently a Professor in the Department of Computer Science and Artificial Intelligence at the University of Granada. He has been the supervisor of 36 Ph.D. students. He has published more than 290 papers in international journals. He currently acts as Editor in Chief of the international journals "Information Fusion" (Elsevier) and "Progress in Artificial Intelligence" (Springer). He acts as editorial member of a dozen of journals. His current research interests include among others, soft computing (including fuzzy modeling and evolutionary algorithms), decision making, bibliometrics, biometric, data preprocessing, data mining, cloud computing and big data.

**José Manuel Benítez** (M'98) received the M.S. and PhD. degrees in Computer Science both from the Universidad de Granada, Spain. He is currently an Associate Professor at the Department of Computer Science and Artificial Intelligence, Universidad de Granada. He is the head of the Distributed Computational Intelligence and Time Series (DiCITS) lab. His research interests include Cloud Computing and Big Data, Data Science, Computational Intelligence and Time Series. He has published in the leading journals of the "Artificial Intelligence" and Computer Science field.