Journal of Network and Computer Applications ■ (■■■■) ■■■–■■■

Contents lists available at ScienceDirect

Journal of Network and Computer Applications

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

# AME-WPC: Advanced model for efficient workload prediction in the cloud

Q1 Katja Cetinski, Matjaž B. Jurič

Q3 *University of Ljubljana, Faculty of Computer and Information Science, Tržaška cesta 25, 1000 Ljubljana, Slovenia*

ARTICLE INFO

*Article history:*
Received 8 July 2014
Received in revised form
15 March 2015
Accepted 2 June 2015

*Keywords:*
Infrastructure management
Cloud computing
IaaS
Resource auto-scaling
Workload prediction
Random forest

ABSTRACT

Workload estimation and prediction has become a very relevant research area in the field of cloud computing. The reason lies in its many benefits, which include QoS (Quality of Service) satisfaction, automatic resource scaling, and job/task scheduling. It is very difficult to accurately predict the workload of cloud applications if they are varying drastically. To address this issue, existing solutions use either statistical methods, which effectively detect repeating patterns but provide poor accuracy for long-term predictions, or learning methods, which develop a complex prediction model but are mostly unable to detect unusual patterns. Some solutions use a combination of both methods. However, none of them address the issue of gathering system-specific information in order to improve prediction accuracy. We propose an Advanced Model for Efficient Workload Prediction in the Cloud (AME-WPC), which combines statistical and learning methods, improves accuracy of workload prediction for cloud computing applications and can be dynamically adapted to a particular system. The learning methods use an extended training dataset, which we define through the analysis of the system factors that have a strong influence on the application workload. We address the workload prediction problem with classification as well as regression and test our solution with the machine-learning method Random Forest on both – basic and extended – training data. To evaluate our proposed model, we compare empirical tests with the machine-learning method kNN (k-Nearest Neighbors). Experimental results demonstrate that combining statistical and learning methods makes sense and can significantly improve prediction accuracy of workload over time.





## 1. Introduction

The area of cloud computing is relatively new and has expanded considerably in the past few years. Usage of dynamically scalable and often virtualized computing resources that are available as services over the Internet have gained a lot of attention from both industry and academia.

The evolution of cloud computing IT services took a step forward in the efficient use of hardware resources through the use of virtualization. In traditional hosting services the user receives a static amount of hardware resources. In contrast, the cloud computing approach offers Q4 on-demand virtualized resources to its users (Buyya et al., 2009). Because virtual resources can be added or removed at any time during the lifetime of the application hosted in a cloud, the possibility of dynamic scaling arises, along with the need for more advanced resource management systems (Manvi and Shyam, 2014). For systems to work seamlessly, decisions about dynamic resource scaling should be carefully scrutinized – because they are influenced by a lot of factors (e.g. current state of the system, number of users, projects, upcoming events and projects such as software development projects or delivering a service to a customer). These decisions can be identified on the basis of future workload predictions, which can be achieved through identification of historical usage patterns, analysis of historical data or current state of the system. To enable more reliable decisions about current and future resource scaling, it is important to establish an effective workload prediction mechanism. Why cannot resources be increased exactly when we need them to? Why is it better to have this knowledge in advance? The answer to those questions relies on the fact that initializing additional virtual resources in a cloud is not instantaneous – cloud-hosting platforms introduce several minutes delay in the hardware resource allocation (Islam et al., 2012), which could cause a lot of inconveniences for the end users (interruptions, operational costs, utilization of resources, loss of clients, etc.). Finally, appropriate workload prediction mechanisms would not only resolve the problem with the shortage of hardware resources, but also the problem with unused resources, which make the cloud costly and inefficient.

In this research paper we present an Advanced Model for Efficient Workload Prediction in the Cloud (AME-WPC) which combines statistical and learning methods. In order to improve capabilities of

*E-mail addresses:* katja.cetinski@cloud.si (K. Cetinski), matjaz.juric@cloud.si (M.B. Jurič).

http://dx.doi.org/10.1016/j.jnca.2015.06.001
1084-8045/

Please cite this article as: Cetinski K, Jurič MB. AME-WPC: Advanced model for efficient workload prediction in the cloud. Journal of Network and Computer Applications (2015), http://dx.doi.org/10.1016/j.jnca.2015.06.001

the learning method, we propose domain-specific database extensions, which we define through analysis of the system factors that have a strong influence on the application workload (e.g. part of day, holidays and weekends). Additional database extensions are defined using a novel Two-phase Pattern Matching method (TPM), which covers two phases: it recognizes similar patterns based on workload value and similar patterns based on workload fluctuation. TPM can be repeatedly applied to the most recent historical workload data in order to regularly improve the prediction model. We address the workload prediction problem with classification as well as regression and test our solution with the machine learning method Random Forest on both – basic (contains only attributes given in source data) and extended (contains an additional set of attributes) – training data. Finally, we demonstrate capabilities of AME-WPC on AuverGrid workload data series, obtained from Grid Workloads Archive (http://gwa.ewi.tudelft.nl/). For evaluation purposes, we include the machine-learning method kNN. Experimental results demonstrate that the use of combined methods significantly improves prediction accuracy of workload over time.

The rest of the paper is organized as follows: Section 2 discusses related work. Section 3 defines a problem domain and presents data collection and processing. The proposed model is highlighted in Section 4, which is followed by experimental results in Section 5. Section 6 discusses the conclusion of the work presented.

## 2. Related work

Methods for ensuring efficient workload prediction in the cloud have yet to be addressed in a way comparable to the approach proposed in this paper. Workload of infrastructure resources can be presented as a time series, which is a sequence of data points typically measured at successive points in time spaced at uniform time intervals. Time series forecasting relies on a model to predict future values based on previously observed values. There are many existing research studies on this topic and researchers have addressed this problem by leveraging different approaches.

### 2.1. Statistical methods

One group of methods that is often used for predicting time series is statistical methods (Quiroz et al., 2009; Mentzer and Moon, 2004; Ganapathi et al., 2010), which cover the identification of similar past occurrences with the current short-term workload history (i.e. pattern matching) (Caron et al., 2010a,b, 2011; Gmach et al., 2007; Liu et al., 2011), autoregression (AR) model (Li et al., 2011; Li, 2005), Monte Carlo (Vercauteren and Aggarwal, 2007), Moving Average (MA) model (Ardagna et al., 2012), Exponential Smoothing (ES) (Kalekar, 2004), Autoregressive Integrated Moving Average (ARIMA) (Zhang et al., 2009; Roy et al., 2011; Doulamis et al., 2007; Kalantari and Akbari, 2009; Cortez et al., 2012), Linear Regression and Quadratic Regression (Sun et al., 2013; Yang et al., 2014) and Hidden Markov Model (HMM) (Khan and Anerousis, 2012; Li and Cheng, 2010; Gong et al., 2010). For short-term predictions and estimation of predicted values, filters such as Kalman's (Kalantari and Akbari, 2009; Cortez et al., 2012) are often used. Furthermore, some of the researchers focus on extracting the small number of trends from historical data that will be most useful to a resource management system (Bacigalupo et al., 2010, 2004, 2005; Bacigalupo, 2006). Moreover, Sarikaya et al. (2010) propose a Statistical Metric Model (SMM) that is system and metric independent for predicting workload behavior. A different solution to a workload prediction problem was presented by Wu et al. (2010), who proposed a model for grid performance prediction. They applied Savitzky–Golay filter to train a sequence of confidence windows and used Kalman filters to minimize prediction errors.

Thus, statistical methods have been successfully used for short-term predictions. Furthermore, HMMs are not a good fit for the time series predictions, since they are used mostly for predicting the labels (hidden states) of a fully observed sequence, not for completing a sequence. More reliable decisions about long-term future workloads are often made based on a complex prediction model, which uses machine-learning methods.

### 2.2. Learning methods

As previously mentioned, the problem with statistical methods is poor accuracy particularly with long-term forecasting. This means that erratic fluctuations, that are typical for time series, are practically impossible to predict. This problem can be resolved with the use of machine-learning methods such as $k$ Nearest Neighbors (kNN), Regression Tree, variations of neural networks (Frank et al., 2001; Chen et al., 2005; Donate et al., 2013; Eddahecha et al., 2013; Chang et al., 2014), Support Vector Machine (SVM) (Cao, 2003) and many others (Chen et al., 2005; Donate et al., 2013; Saadatfar et al., 2012). Advantages of these methods are that they learn from historical data (search connections among attributes) and build a model that is used for predicting future values. Variations of neural networks (NN) have been used widely for time series predictions. As mentioned, NNs can be accurate prediction models, but are time-consuming and complex. On the other hand, simple machine-learning methods such as Naïve Bayes and Linear Regression do not perform with sufficient accuracy for complex and non-linear problems such as time series predictions.

The machine-learning method kNN was used for time series prediction by various researchers (Troncoso Lora et al., 2004; Imandoust and Bolandraftar, 2013; Ban et al., 2013). The main idea of the kNN technique for pattern classification is based on the similarity of the individuals. It classifies objects based on closest training examples in the feature space. In a similar way, our approach searches for patterns in existing training datasets, but with our own pattern-matching technique TPM. Similar patterns are determined based on two phases – value and fluctuation. In contrast to kNN, our approach extends training dataset based on the results of TPM, which identifies the most similar time-points in the historical workload and produces additional attributes. Furthermore, our approach applies confidence factors to the Random Forest predictions, which improves confidence in the predicted values.

### 2.3. Hybrid methods

In order to achieve better workload prediction accuracy, the following researchers used a combination of statistical and machine-learning methods. Montes et al. (2011) propose an approach that combines the use of the machine-learning prediction techniques with a single entity vision of the grid in order to improve the management of the whole system. Furthermore, Vercauteren and Aggarwal (2007) propose a solution to the web server load prediction problem based on a hierarchical framework. Li et al. (2011) present an integrated approach that employs three-layered resource controllers using different analytic techniques, including statistical machine learning. Moreover, Li (2005) proposes a hierarchical framework for modeling workload. Zhang (2003) proposes a hybrid methodology that combines both ARIMA and ANN models. Moreover, Cortez et al. (2012) present three methods for traffic forecasting in TCP/IP based networks: a neural network ensemble method, ARIMA and Holt–Winters. Frank et al. (2001) use neural networks as time series predictors, which employ a sliding window over the input sequence. Imam and Miskhat (2011) present time delay neural networks and regression methods for predicting future workloads in the grid or cloud platform. In similar way, Islam et al. (2012) develop prediction-based resource measurements and provisioning strategies using neural network and linear regression to satisfy upcoming resource demands.



Please cite this article as: Cetinski K, Jurič MB. AME-WPC: Advanced model for efficient workload prediction in the cloud. Journal of Network and Computer Applications (2015), http://dx.doi.org/10.1016/j.jnca.2015.06.001

Machine-learning methods are more reliable for long-term workload predictions and can, in combination with statistical methods, provide more accurate workload predictions. Furthermore, it is important that we provide as much information about the addressed system as possible and extract additional features that have an influence on workload value, which can significantly improve the performance of both statistical and machine-learning methods.

Table 1 presents a comparison between our approach and existing hybrid solutions. The main drawback of existing solutions is the lack of information about the addressed system. This information would significantly improve the performance of either statistical or machine learning methods. Moreover, we strongly believe that combining statistical and learning methods is reasonable, as it provides the advantages of both sides. Training data, which presents the most important part of prediction with machine learning, should be extended with appropriate features. For this purpose we developed our own Two-phase Pattern Matching (TPM) method, which extracts significant features out of historical data. AME-WPC can be dynamically adapted to a specific system. After a certain period of time, learning datasets can be redefined based on most-recent historical workload data in order to further improve prediction accuracy of the prediction model. Furthermore, we apply confidence factors for achieving more reliable predictions. We address the challenges of poor prediction accuracy by building an Advanced Model for Efficient Workload Prediction in the Cloud (AME-WPC).

## 3. Problem definition and data collection

### 3.1. Problem definition

Our goal is to improve prediction accuracy of system workloads in order to improve automatic scaling of cloud resources. Automatic resource scaling enables the system to automatically increase or decrease the amount of infrastructure resources depending on the past, current and future needs. This is useful especially when we need to react quickly in the case of shortage of hardware resources. The time difference between sending the request and the actual acquisition is minimal but not instant and can consequently lead to serious performance and capacity bottlenecks.

The problem of workload prediction can be resolved in several ways. The most common solution is the use of historical workload data for planning the future workload of the system. Maximum or average loads can be observed for specified time intervals. However, such methods are very general and do not give accurate predictions. If we consider maximum workloads, resources will be unused most of the time. On the other hand, if an average workload is taken into account, there will be a lack of resources (and performance will decrease) when workload increases. These kinds of prediction methods are considered to be poor because they are not accurate enough and correspond to a very small number of cases (Roy et al., 2011).

The problem of workload prediction can be resolved more efficiently by using appropriate machine-learning methods, assuming that a historical record of workload is available for a specified period of time in the past. A major focus of research in the field of machine learning is to automatically learn to recognize complex patterns and make intelligent decisions based on existing data. A prediction model can be built by mining the data in the training window (i.e. historical workload data) and use it to predict the workload throughout a prediction window (i.e. testing data). Figure 1 shows training and prediction windows.

### 3.2. Data collection and preprocessing

The first thing to consider when trying to build a prediction model is data. We obtain our training data from the Grid Workloads Archive (http://gwa.ewi.tudelft.nl/), which offers several different workload traces and is widely adopted in the academic research field. Because we need a historical workload trace in order to build a prediction model, we choose the AuverGrid trace (http://gwa.ewi.tudelft.nl/datasets/gwa-t-4-auvergrid), since it provides the most complete data for the given attributes. These traces were provided by the AuverGrid team, the owners of the AuverGrid system.

The AuverGrid contains traces of job records collected during 12 months, from January 1 to December 31, 2006 (Fig. 2). Although the total number of 29 attributes is defined, some attributes are unavailable or partially available due to the limitations in the environment. The most useful attributes for us are *submission time*, *wait time*, *run time* and the *number of processors* used for each job. Based on these attributes we transformed the original data into a

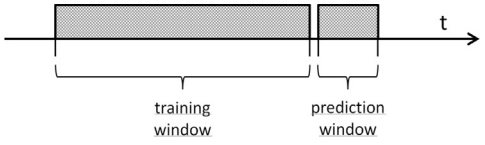


**Fig. 1.** Training and prediction windows.

**Table 1**
Comparison of existing approaches.

| | Statistical approach | Learning approach | Feature extraction | Confidence factors |
|---|---|---|---|---|
| **AME-WCA** | + | + | + | + |
| Caron et al. (2010a, b, 2011), Ardagna et al. (2012), Kalekar (2004), Zhang (2003), Roy et al. (2011), Bacigalupo et al. (2004, 2005, 2010, 2011); Bacigalupo (2006), Khan and Anerousis (2012), Li and Cheng (2010), Gong et al. (2010), Sarikaya et al. (2010), Sun et al. (2013), and Yang et al. (2014) | + | / | / | / |
| Gmach et al. (2007) and Doulamis et al. (2007) | + | / | + | / |
| Liu et al. (2011), Kalantari and Akbari (2009), and Wu et al. (2010) | + | / | / | + |
| Li et al. (2011), Li (2005), Zhang et al. (2009), Cortez et al. (2012), Frank et al. (2001), Imam and Miskhat (2011), and Islam et al. (2012) | + | + | / | / |
| Montes et al. (2011) | + | + | + | / |
| Vercauteren and Aggarwal (2007) | + | + | / | + |
| Chen et al. (2005), Donate et al. (2013), Eddahecha et al. (2013), Chang et al. (2014), Troncoso Lora et al. (2004), Imandoust and Bolandraftar (2013), and Ban et al. (2013) | / | + | / | / |
| Cao (2003) and Saadatfar et al. (2012) | / | + | + | / |

| JobID | SubmitTime | WaitTime | RunTime | NProc | UsedCPUTime | UsedMemory | ReqNProcs | ReqTime | ReqMe... | Status | UserID |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1136070690 | 0 | 11 | 1 | 4 | 35848 | 1 | 259200 | -1 | 1 | U1023S0 |
| 2 | 1136072890 | 1 | 46 | 1 | 3 | 49216 | 1 | 900 | -1 | 1 | U1018S0 |
| 3 | 1136074695 | 1 | 197831 | 1 | 0 | 0 | 1 | 259200 | -1 | 1 | U1033S0 |
| 4 | 1136074754 | 2 | 197682 | 1 | 0 | 0 | 1 | 259200 | -1 | 1 | U1033S0 |
| 5 | 1136074756 | 2 | 197552 | 1 | 0 | 0 | 1 | 259200 | -1 | 1 | U1033S0 |
| 6 | 1136074814 | 1 | 197495 | 1 | 0 | 0 | 1 | 259200 | -1 | 1 | U1033S0 |
| 7 | 1136077528 | 58 | -1 | 0 | -1 | -1 | 1 | -1 | -1 | 5 | -1 |
| 8 | 1136077775 | 58 | -1 | 0 | -1 | -1 | 1 | -1 | -1 | 5 | -1 |
| 9 | 1136077889 | 58 | -1 | 0 | -1 | -1 | 1 | -1 | -1 | 5 | -1 |
| 10 | 1136078195 | 58 | -1 | 0 | -1 | -1 | 1 | -1 | -1 | 5 | -1 |
| 11 | 1136078435 | 58 | -1 | 0 | -1 | -1 | 1 | -1 | -1 | 5 | -1 |
| 12 | 1136078608 | 58 | -1 | 0 | -1 | -1 | 1 | -1 | -1 | 5 | -1 |
| 13 | 1136078676 | 117 | -1 | 0 | -1 | -1 | 1 | -1 | -1 | 5 | -1 |
| 14 | 1136079148 | 58 | -1 | 0 | -1 | -1 | 1 | -1 | -1 | 5 | -1 |
| 15 | 1136272204 | 2 | 922 | 1 | 20 | 74696 | 1 | 172800 | -1 | 1 | U1030S0 |
| 16 | 1136274078 | 1 | 19143 | 1 | 18070 | 490916 | 1 | 259200 | -1 | 1 | U1033S0 |
| 17 | 1136274080 | 2 | 19432 | 1 | 18657 | 470052 | 1 | 259200 | -1 | 1 | U1033S0 |

**Fig. 2.** AuverGrid job traces.

time series that contain workload values recorded for each second in an entire time interval (12 months). We define workload as the amount of processors needed at a certain time. In order to create a time series, we developed a method in GNU Octave (Eaton, 2002), which transforms all logged jobs into a time series. GNU Octave is a high-level interpreted language primarily intended for numerical computations. The data is sub-sampled so that workload values are acquired only four times in an hour (every 15 min). The latter is required in order to make our model less computationally complex. The file is in a '.TAB' format, which is required when using machine-learning libraries such as Orange (Demšar et al., 2013). This presents our basic dataset, which we structure into training and testing sets. Figure 3 presents a part of our training dataset with basic attributes: *date* (*month and day*), *time* (*hour and minute*) and *workload*. Extended datasets, which contain additional attributes, will be defined later on(Fig. 4).

| | month | day | hour | minute | workload |
|---|---|---|---|---|---|
| 2 | d | d | d | d | c |
| 3 | | | | | class |
| 4 | 1 | 4 | 1 | 45 | 68 |
| 5 | 1 | 4 | 1 | 0 | 68 |
| 6 | 1 | 4 | 1 | 15 | 68 |
| 7 | 1 | 4 | 1 | 30 | 63 |
| 8 | 1 | 4 | 1 | 45 | 65 |
| 9 | 1 | 4 | 1 | 0 | 58 |
| 10 | 1 | 4 | 1 | 15 | 36 |
| 11 | 1 | 4 | 1 | 30 | 21 |
| 12 | 1 | 4 | 1 | 45 | 16 |
| 13 | 1 | 4 | 1 | 0 | 16 |
| 14 | 1 | 4 | 1 | 15 | 14 |
| 15 | 1 | 4 | 1 | 30 | 14 |
| 16 | 1 | 4 | 1 | 45 | 14 |
| 17 | 1 | 4 | 1 | 0 | 15 |
| 18 | 1 | 4 | 1 | 15 | 15 |
| 19 | 1 | 4 | 1 | 30 | 15 |
| 20 | 1 | 4 | 1 | 45 | 14 |
| 21 | 1 | 4 | 1 | 0 | 14 |
| 22 | 1 | 4 | 1 | 15 | 14 |
| 23 | 1 | 4 | 1 | 30 | 14 |
| 24 | 1 | 4 | 1 | 45 | 14 |
| 25 | 1 | 4 | 1 | 0 | 14 |
| 26 | 1 | 4 | 1 | 15 | 15 |
| 27 | 1 | 4 | 1 | 30 | 15 |
| 28 | 1 | 4 | 1 | 45 | 16 |
| 29 | 1 | 4 | 1 | 0 | 16 |
| 30 | 1 | 4 | 1 | 15 | 14 |

**Fig. 3.** Basic training dataset.

In machine learning, problems can be approached either with classification or regression. The difference is in output variable values: regression involves estimating or predicting a response and classification identifies group membership. We address workload prediction problems with both regression and classification in order to determine which method is more appropriate for our problem domain. Because the number of classes in our training data is too large, we transform the workload attribute to a different scale in order to use the classification approach. For the purpose of testing, data is transformed into a group of 11 classes and a group of 17 classes. We determine the number of classes based on the most appropriate division of the original data. In the next step, we divide our data into learning and testing sets. Learning data is divided into 3-, 5- and 7-month intervals and the testing data into 24-h intervals. We chose one-day interval predictions because in this case, it is neither realistic nor useful to predict further into the future. If the system is unpredictable and workload fluctuates regularly, the number of daily predictions should be properly adjusted.

## 4. The proposed model

In this section, we first present an overview of the AME-WPC. Our model consists of six steps (Fig. 5). In Step 1, we first analyze and obtain historical workload records (time series). Step 2 features extraction by leveraging our novel Two-phase Pattern Matching (TPM) method and attribute scoring. Additional features are extracted and scored from historical workload data (i.e. part of day, weekends) (Step 3). Then we divide our data into different training and testing sets (Step 4), which we use with the Random Forest method (Step 5). Finally, confidence factors are applied to workload predictions (Step 6) in order to achieve more reliable results. The following subsections present each part of our model in detail.

### 4.1. Analysis of time series

We can determine periodicity of the time series with the use of autocorrelation, which represents a cross-correlation of a signal with itself. Autocorrelation shows the similarity between observations as a



Please cite this article as: Cetinski K, Jurič MB. AME-WPC: Advanced model for efficient workload prediction in the cloud. Journal of Network and Computer Applications (2015), http://dx.doi.org/10.1016/j.jnca.2015.06.001

| class | from | to |
|---|---|---|
| 0 | 0 | 13 |
| 1 | 16 | 29 |
| 2 | 30 | 43 |
| 3 | 44 | 57 |
| 4 | 58 | 71 |
| 5 | 72 | 85 |
| 6 | 86 | 99 |
| 7 | 100 | 113 |
| 8 | 114 | 127 |
| 9 | 128 | 141 |
| 10 | 142 | 155 |
| 11 | 156 | 169 |

| class | from | to |
|---|---|---|
| 0 | 0 | 9 |
| 1 | 10 | 18 |
| 2 | 19 | 27 |
| 3 | 28 | 36 |
| 4 | 37 | 45 |
| 5 | 46 | 54 |
| 6 | 55 | 63 |
| 7 | 64 | 72 |
| 8 | 73 | 81 |
| 9 | 82 | 90 |
| 10 | 91 | 99 |
| 11 | 100 | 109 |
| 12 | 110 | 119 |
| 13 | 120 | 129 |
| 14 | 130 | 139 |
| 15 | 140 | 149 |
| 16 | 150 | 159 |
| 17 | 160 | 169 |

**Fig. 4.** Classification of workload in 11 and 17 classes.

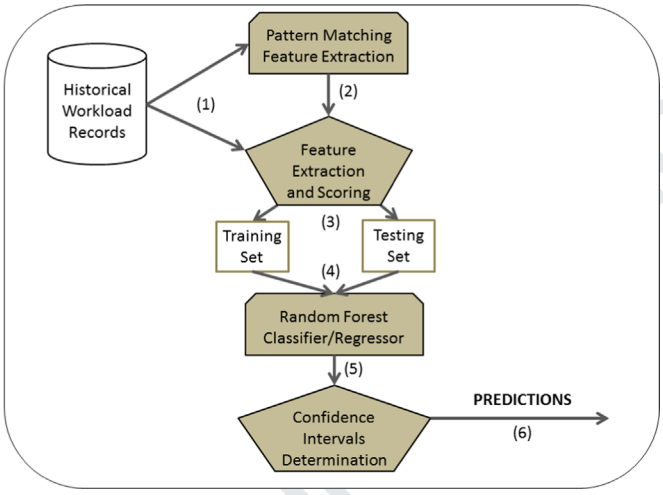


**Fig. 5.** Prediction model.

function of the time lag between them and is a mathematical tool for finding repeating patterns such as the presence of a periodic signal obscured by noise, or identifying the missing fundamental frequency in a signal implied by its harmonic frequencies.

Randomness of a time series can be verified with a lag plot, which checks whether a data set or time series is random. Lag plots can be generated for any arbitrary lag, although the most commonly used lag is 1. A plot of lag 1 is a plot of the values of $Y_i$ versus $Y_{i-1}$. This means that the vertical axis is $Y_i$ for all $i$ and the horizontal axis $Y_{i-1}$ contains for all $i$. Random data should not exhibit any identifiable structure on the lag plot. Non-random structures in the lag plot indicate that the underlying data are not random.

Before making a decision about which model to use and which features to include in the learning dataset, it is useful to check for periodicity, randomness and similar characteristics of the observed signal. Thus, according to the analysis of periodicity and randomness, which indicates whether a signal contains certain characteristics, we continue with feature extraction.

### 4.2. Feature extraction

We analyze our training dataset with a Two-phase Pattern Matching (TPM) method that we have developed with the use of Python programming language. The main objective is to find similar patterns to those in the historical data – we want to determine when in the past similar fluctuations and workload values have been recorded. Similarity is defined based on a value of the workload, as well as fluctuation of the pattern (whether the workload is rising, dropping or stable). First we find similar values for specific time interval and then select only those intervals that have high fluctuation similarities. We consider a 24-h time interval because our prediction will be for one day ahead. Pseudo-code of algorithms for value similarity and fluctuation similarity is shown below (Algorithms 1 and 2).

**Algorithm 1.** Method 1: Returns' patterns with corresponding values.

**Require:** *sptr* ← sample pattern
**Require:** *allptr* ← historical data
  **for all** 1-day patterns 1*dayptr* in *allptr* **do**
    compare same position element values ($x$ and $y$) of 1*dayptr* and *sptr*
    set assessment $s$ for 1*dayptr* to 0
    **if** $x == y$ **then**
      increase assessment for 1*dayptr* by 1
    **else**
      increase assessment for 1*dayptr* with a number proportional to the error
    **end if**
  **end for**
  **return** patterns with the largest assessments

**Algorithm 2.** Method 2: Returns' patterns with corresponding fluctuations.

**Require** *sptr* ← sample pattern
**Require** *alg*1*res* ← result of Algorithm 1
  **for all** 1-day patterns 1*dayptr* in *alg*1*res* **do**
    compare two sequential, same position elements ($x1$, $x2$) and ($y1$, $y2$) from 1*dayptr* and *sptr*
    set assessment $s$ for 1*dayptr* to 0
    set difference $d1 = x1 - x2$ and $d2 = y1 - y2$
    set values for error factors *err*1 and *err*2
    **if** $d1 < 0$ **then**
      set direction $dr1 = 1$
    **else if** $d1 == 0$ **then**
      set direction $dr1 = 0$
    **else**
      set direction $dr1 = -1$
    **end if**
    **if** $d2 < 0$ **then**
      set direction $dr2 = 1$
    **else if** $d2 == 0$ **then**
      set direction $dr2 = 0$
    **else**
      set direction $dr2 = -1$
    **end if**
    **if** $dr1 == dr2$ **then**
      increase assessment for 1*dayptr* by 1
    **else if** $(dr1 - dr2) == 1$ OR $(dr1 - dr2) == -1$ **then**
      increase assessment for 1*dayptr* by *err*1
    **else**
      decrease assessment for 1*dayptr*) by *err*2
    **end if**



Please cite this article as: Cetinski K, Jurič MB. AME-WPC: Advanced model for efficient workload prediction in the cloud. Journal of Network and Computer Applications (2015), http://dx.doi.org/10.1016/j.jnca.2015.06.001

**end for**
**return** patterns with the largest assessments

With analysis of historical data through TPM, we are able to obtain relevant information about connections among current and historical data. In such a manner, we collect additional attributes for our training dataset, which will be used as a learning set for the machine-learning method hereinafter.

### 4.3. Attribute scoring

We perform attribute scoring by leveraging Orange's (Demšar et al., 2013) built in scoring method, which is based on the feature selection algorithm Relief (Kononenko, 1994). The latter is considered one of the most successful algorithms for assessing the quality of features due to its simplicity and effectiveness. Feature scoring is an assessment of the usefulness of the feature for prediction of the dependent (class) variable. Orange, which is an open source data visualization and analysis tool that also provides data mining through Python scripting, provides various feature scoring methods for classification as well as regression. With the help of Orange's generalized scoring method, we identify different sets of attributes for classification and regression data. Orange's scoring method calculates scores for each individual attribute, where a larger value means that the attribute is more important for an accurate output of a machine-learning method and a lower value means that the attribute is less important (or unnecessary) for an accurate value of a machine-learning method. Orange's scoring method has an option to automatically determine the top $n$ candidates out of the provided attributes. In our case, $n$ was limited to 6 after performing tests with different values of $n$.

### 4.4. Training and testing datasets

Both training datasets (basic and extended) are divided into three different datasets according to the time intervals (3, 5 and 7 months) in order to verify whether larger datasets contribute to a more accurate prediction model. We perform workload predictions one day in advance, because long-term forecasts have no specific relevance within the domain of automatic resource scaling.

### 4.5. Random forest method

When a large amount of data should be analyzed in order to extract trends, similarities and patterns, machine learning can be a good choice. Some machine-learning methods (e.g. Support Vector Machines and Random Forest) can work directly with high-dimensional datasets such as time series. We use Random Forests, since it is one of the most successful ensemble methods that exhibits performance on the level of boosting and support-vector machines. The method is fast, robust to noise, does not overfit and offers possibilities for explanations and visualization of its output (Robnik-Šikonja, 2004). Random Forests are a combination of tree predictors such that each tree depends on the values of a random vector sampled independently and with the same distribution for all trees in the forest (Breiman, 2001). We develop and test the prediction model using the Random Forest method in the PythonWin environment using Python programming language with Orange (Demšar et al., 2013) and Scipy (Jones et al., 2001) libraries.

### 4.6. Confidence factors

To further improve our model, we add confidence factors to predictions in order to make them more reliable. Confidence factors are specified on the basis of an actual prediction value and Random Forest classifier probabilities of an individual class. Highest probabilities from Random Forest classifiers are summed with an additional value added based on the highest probability class. This factor is then multiplied with original values of Random Forest predictions.

Figure 5 presents all parts of our prediction model with their connections. For a successful learning process, quality training data is very important. To improve the accuracy of the prediction, we extend our basic training dataset (historical workload data), which contains the attributes of *date*, *time* and *workload*, with additional attributes that we identify through the analysis of the historical workload data. Our pattern-matching method identifies similar historical patterns which are included as additional attributes and significantly contribute to a better prediction accuracy which we prove through various tests in the next section.

## 5. Testing and results

In this section we analyze historical workload data according to our model presented in the previous section. We perform tests on various testing and training datasets as specified below. Finally, results are presented in the form of prediction errors and visual representations (graphs).

### 5.1. Historical data analysis

Before making the decision about which model to use and which features to include into learning dataset, it is useful to check for different characteristics of the observed signal (e.g. periodicity, randomness and others).

#### 5.1.1. Autocorrelation and randomness

Autocorrelation is a mathematical tool for finding repeating patterns such as the presence of a periodic signal obscured by noise. A correlogram shows the ACF (Autocorrelation Function) of a signal at different lags. The *X*-axis of the plot represents the lag in minutes, while the *Y*-axis represents the value of an autocorrelation. Figure 6 shows the correlogram of our workload signal. Note that with the exception of lag 0, which is always 1 by definition, almost all of the autocorrelations fall outside the 95 percent confidence limits (gray horizontal lines above and below zero autocorrelation). The lags slightly outside the 95 percent - confidence limit do not necessarily indicate non-randomness. For a 95 percent confidence interval, we may expect about one out of 20 lags to be statistically significant due to random fluctuations. Statistical significance implies non-randomness and strong periodicity of the signal.

We can see that autocorrelation coefficients for our signal are not only completely random, but also do not have high autocorrelation values, which means that the corresponding workload does not have strong periodicity – autocorrelation is weak. This is to be expected for time series signals, since fluctuations can be very difficult to interpret. We have to find other factors to help determine future workload values.

Randomness of a time series can be verified with a lag plot, which checks whether a data set or time series is random. Our lag plot exhibits a linear pattern (Fig. 7). This shows that the data is strongly non-random and further suggests that an autoregressive model may be appropriate. However, this would be useful only for short term predictions. This means if we know $Y_{i-1}$ we can make a strong guess about the value of $Y_i$. This prediction could be used for ongoing verification and adjustment of rough long-term predictions.

#### 5.1.2. Feature extraction

We have applied the Two-phase Pattern Matching method to our historical workload data. The results of TPM have shown that



Please cite this article as: Cetinski K, Jurič MB. AME-WPC: Advanced model for efficient workload prediction in the cloud. Journal of Network and Computer Applications (2015), http://dx.doi.org/10.1016/j.jnca.2015.06.001

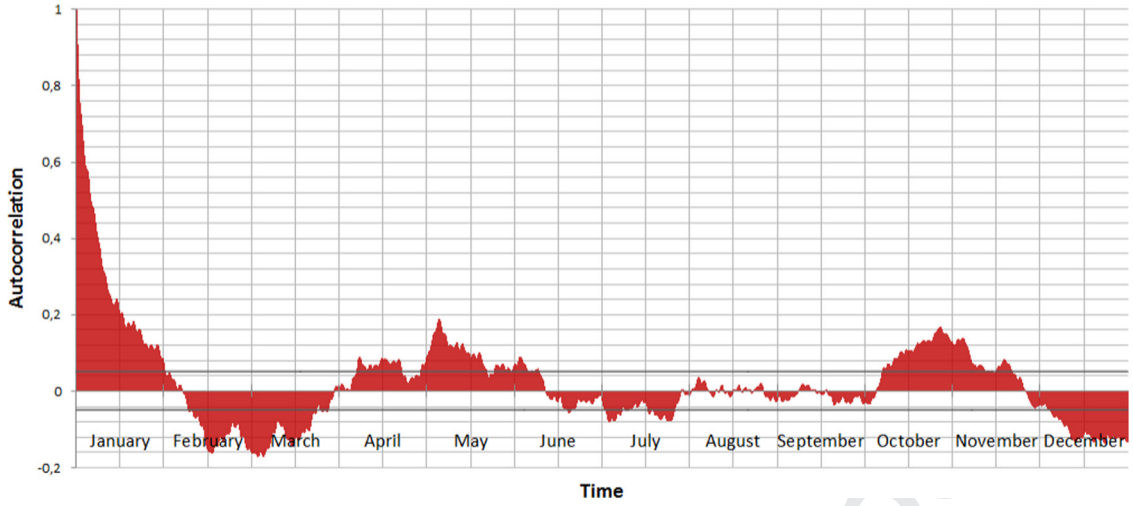


**Fig. 6.** Correlogram of our time series.

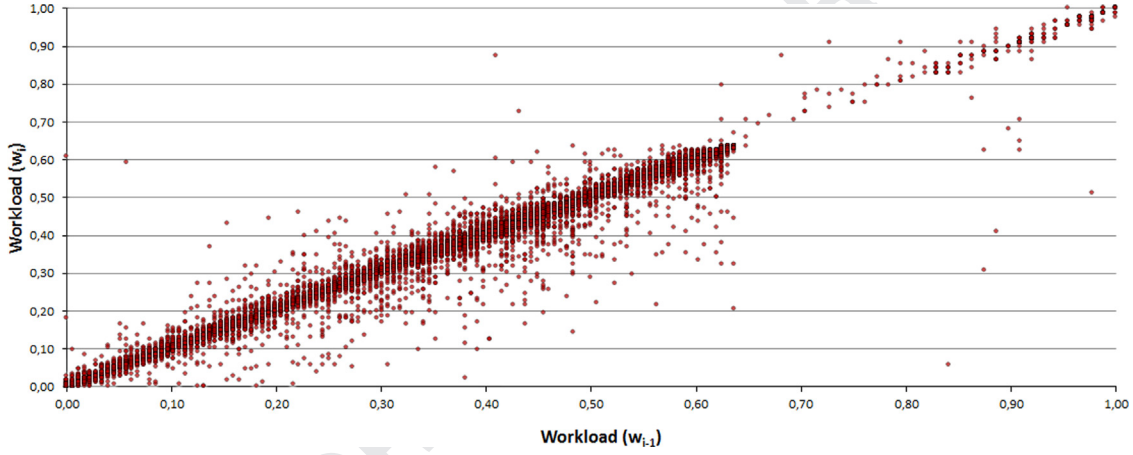


**Fig. 7.** Lag plot of our time series.

there is a strong correlation among current workload and historical workload for four specific time intervals (*att1*, *att2*, *att3*, *att4*). These attributes are continuous and represent workload values at a certain time in the past. *att1* presents workload 1 day ago, *att2* 9 days ago, *att3* 11 days ago and *att4* 18 days ago. More distant attributes (greater than one month) were not considered because that would result in a larger learning dataset or smaller learning dataset with missing values. Aside from that, we have identified two discrete attributes – *partofday* and *weekend*. The first specifies part of the day (1-night, 2-morning, 3-afternoon and 4-evening) and the second tells whether it is a weekend (1-weekend and 0-working day). Because the large number of attributes can negatively affect the behavior of machine-learning methods, the next step is to evaluate each identified attribute and choose only those that contribute the most (i.e. have the highest assessments).

#### 5.1.3. Attribute scoring

For classification, the scoring method eliminated the *weekend*, *part-of-day*, *att2* and *att4* attributes, due to their low classification scores. For regression, the scoring method eliminated *part-of-day*, *att2*, *att4* and *minute* because they had the lowest scores of all regression attributes. As a result we obtained updated training datasets with additional attributes for both methods (classification and regression), which will be used for testing the machine-learning method.

#### 5.1.4. Datasets

We label the learning datasets as *ti1*, *ti2* and *ti3*. The first dataset includes 8600 learning cases (approx. 3 months), the second database includes 14,500 learning cases (approx. 5 months) and the third one includes 20,000 cases (approx. 7 months). Workload prediction is performed one day ahead (24 h). Within each learning dataset (3, 5 and 7 months) we perform test on 3 different testing datasets – day-1, day-2 and day-3. Day-1 represents testing dataset for the day following the last day of the adequate training dataset. Day-2 represents testing data for the day after day-1 and day-3 for the day after day-2.

### 5.2. Evaluation of results

For the purpose of comparing prediction results of different datasets, we compute mean squared prediction errors.

In statistics, the mean squared error (MSE) of an estimator measures the average of the squares of the difference between the estimator and what is estimated (predicted):

$$MSE = \frac{1}{n}\sum(\hat{Y}_i - Y_i)^2 \quad (1)$$

In our case, the random variable $Y$ presents true values and the random variable $\hat{Y}$ presents predicted values. Figure 8 presents prediction errors, which were computed based on the aforementioned equation for MSE. Symbol $n$ presents the number of observed instances. Additionally, we also computed normalized MSE's, which

Please cite this article as: Cetinski K, Jurič MB. AME-WPC: Advanced model for efficient workload prediction in the cloud. Journal of Network and Computer Applications (2015), http://dx.doi.org/10.1016/j.jnca.2015.06.001

a

| Test data | Approach | MSE | NMSE |
|---|---|---|---|
| day 1 | Reg_A | 538 | 3,4 |
| | Reg_B | 735 | 4,6 |
| | kNN | 634 | 4,0 |
| day 2 | Reg_A | 1493 | 9,3 |
| | Reg_B | 1753 | 11,0 |
| | kNN | 2421 | 15,1 |
| day 3 | Reg_A | 882 | 5,5 |
| | Reg_B | 1090 | 6,8 |
| | kNN | 1649 | 10,0 |

| Test data | Approach | MSE | NMSE |
|---|---|---|---|
| day 1 | Clas1_A | 29 | 1,7 |
| | Clas1_B | 176 | 10,3 |
| | kNN | 55 | 3,2 |
| day 2 | Clas1_A | 50 | 2,9 |
| | Clas1_B | 64 | 3,8 |
| | kNN | 80 | 4,7 |
| day 3 | Clas1_A | 6 | 0,3 |
| | Clas1_B | 148 | 8,7 |
| | kNN | 57 | 3,3 |

| Test data | Approach | MSE | NMSE |
|---|---|---|---|
| day 1 | Clas2_A | 5 | 0,4 |
| | Clas2_B | 12 | 1,1 |
| | kNN | 22 | 2,0 |
| day 2 | Clas2_A | 8 | 0,7 |
| | Clas2_B | 13 | 1,2 |
| | kNN | 12 | 1,00 |
| day 3 | Clas2_A | 3 | 0,20 |
| | Clas2_B | 27 | 2,5 |
| | kNN | 12 | 1,1 |

b

| Test data | Approach | MSE | NMSE |
|---|---|---|---|
| day 1 | Reg_A | 1004 | 6,3 |
| | Reg_B | 2312 | 14,4 |
| | kNN | 2871 | 17,9 |
| day 2 | Reg_A | 6 | 0,04 |
| | Reg_B | 1790 | 11,2 |
| | kNN | 1864 | 11,6 |
| day 3 | Reg_A | 35 | 0,2 |
| | Reg_B | 626 | 3,9 |
| | kNN | 747 | 4,7 |

| Test data | Approach | MSE | NMSE |
|---|---|---|---|
| day 1 | Clas1_A | 0 | 0,0 |
| | Clas1_B | 1 | 0,1 |
| | kNN | 99 | 5,8 |
| day 2 | Clas1_A | 81 | 4,8 |
| | Clas1_B | 106 | 6,3 |
| | kNN | 87 | 5,1 |
| day 3 | Clas1_A | 78 | 4,6 |
| | Clas1_B | 113 | 6,7 |
| | kNN | 82 | 4,8 |

| Test data | Approach | MSE | NMSE |
|---|---|---|---|
| day 1 | Clas2_A | 4 | 0,4 |
| | Clas2_B | 9 | 0,8 |
| | kNN | 54 | 4,9 |
| day 2 | Clas2_A | 25 | 2,3 |
| | Clas2_B | 27 | 2,4 |
| | kNN | 32 | 2,9 |
| day 3 | Clas2_A | 9 | 0,8 |
| | Clas2_B | 40 | 3,6 |
| | kNN | 40 | 3,6 |

c

| Test data | Approach | MSE | NMSE |
|---|---|---|---|
| day 1 | Reg_A | 44 | 0,3 |
| | Reg_B | 51 | 0,3 |
| | kNN | 76 | 0,5 |
| day 2 | Reg_A | 9 | 0,1 |
| | Reg_B | 39 | 0,2 |
| | kNN | 122 | 0,8 |
| day 3 | Reg_A | 12 | 0,1 |
| | Reg_B | 43 | 0,3 |
| | kNN | 111 | 0,7 |

| Test data | Approach | MSE | NMSE |
|---|---|---|---|
| day 1 | Clas1_A | 42 | 2,5 |
| | Clas1_B | 175 | 10,3 |
| | kNN | 105 | 6,2 |
| day 2 | Clas1_A | 1 | 0,0 |
| | Clas1_B | 94 | 5,5 |
| | kNN | 39 | 2,3 |
| day 3 | Clas1_A | 1 | 0,0 |
| | Clas1_B | 46 | 2,7 |
| | kNN | 11 | 0,65 |

| Test data | Approach | MSE | NMSE |
|---|---|---|---|
| day 1 | Clas2_A | 1 | 0,1 |
| | Clas2_B | 0 | 0,00 |
| | kNN | 11 | 1,0 |
| day 2 | Clas2_A | 0 | 0,0 |
| | Clas2_B | 0 | 0,0 |
| | kNN | 0 | 0 |
| day 3 | Clas2_A | 1 | 0,1 |
| | Clas2_B | 4 | 0,4 |
| | kNN | 9 | 0,8 |

**Fig. 8.** Tables show MSE's and NMSE's of methods Clas1_A (Classification method with 17 classes on extended training data), Clas1_B (Classification method with 17 classes on basic training data), Clas2_A (Classification method with 11 classes on extended training data), Clas2_B (Classification method with 11 classes on basic training data), Reg_A (Regression with values 0–170 on extended training data) and Reg_B (Regression with values 0–170 on basic training data) mark on different training data: (a) 3 month time interval; (b) 5 month time interval; and (c) 7 month time interval.

were adjusted by MSE values measured on different scales due to different number of classes (regression and classification with different number of classes), to a notionally common scale. The following equation presents the formula for a normalized MSE:

$$NMSE = \frac{1}{n*m}\sum(\hat{Y}_i - Y_i)^2 \quad (2)$$

where $m$ stands for the number of classes, with either 11 or 17 in the case of classification or 170 in the case of regression.

We perform the following tests. For training data from 3, 5 and 7 months (*ti1*, *ti2* and *ti3*) we perform tests for the one-day-ahead prediction (24 h) for both basic and extended training datasets. If we look at the error numbers for 3 months of training data in Fig. 8 we see that regression (marked as *Reg* in the table) did not perform satisfactory enough for our model. Both classification methods outperformed our model, which is why we decided to rule out the regression method and try to further improve both classification methods.

To further improve our classification methods, we add confidence factors to predictions in order to make them more reliable. Confidence factors are specified on the basis of an actual prediction value and Random Forest classifier probabilities of individual classes. Highest probabilities from Random Forest classifiers are summed with an additional value added based on the highest probability class. These factors are then multiplied with original values of Random Forest predictions (Algorithm 3).

**Algorithm 3.** Computed highest probability class from the Random Forest class probabilities.

**Require:** *train* ← training dataset
**Require:** *test* ← testing dataset
set *forest* = Orange.ensemble.forest. RandomForestLearner (train)

**for all** *instance* in *test* **do**
set *iclass* = *instance.get_class().value*

Please cite this article as: Cetinski K, Jurič MB. AME-WPC: Advanced model for efficient workload prediction in the cloud. Journal of Network and Computer Applications (2015), http://dx.doi.org/10.1016/j.jnca.2015.06.001

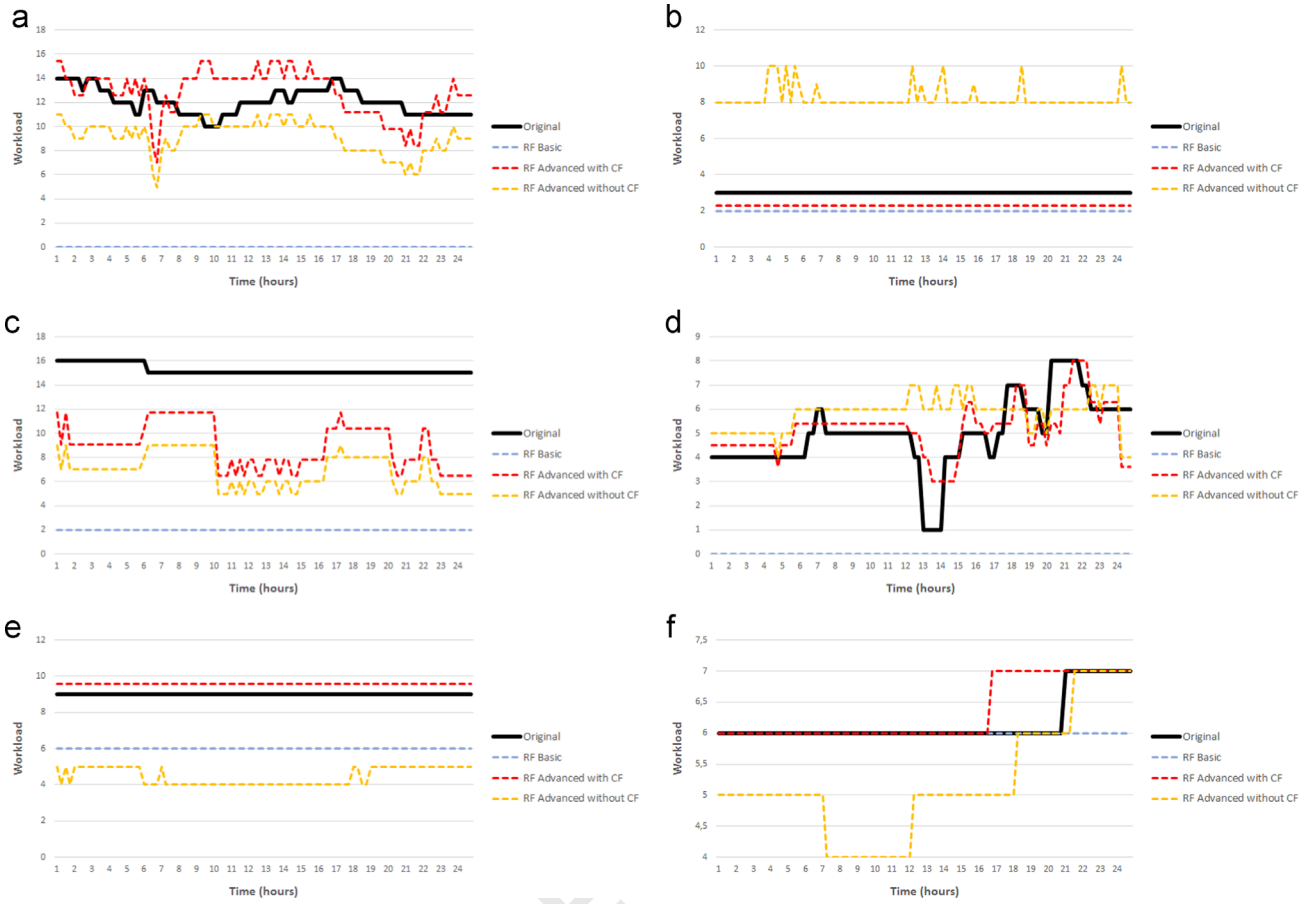


**Fig. 9.** Advanced Random Forest prediction with added confidence factors versus basic Random Forest prediction and advanced Random Forest prediction without confidence factors for the next 24 h on training data ti1 (day 3), ti2 (day 1) and ti3 (day 2). Clas1 represents classification with 17 classes and Clas2 classification with 11 classes: (a) 3 months, Clas1; (b) 5 months, Clas1; (c) 7 months, Clas1; (d) 3 months, Clas2; (e) 5 months, Clas2; and (f) 7 months, Clas2.

set *probabilities*=*forest* (*instance*, *Classifier.GetProbabilities*)
compute highest probability class for each *instance*
**end for**
**return** highest probability class for each *instance*

Experimental tests prove that confidence factors additionally improve our model, which clearly show visual presentations – actual workload values are more similar to the Random Forest prediction values with added confidence factors than to the Random Forest prediction values without confidence factors.

MSE values show that the larger size of the learning dataset does not considerably improve prediction accuracy, meaning that 3 months of training data is enough for making accurate predictions.

The graphs in Fig. 9 show prediction results for the basic prediction method (blue dotted line). This is a Random Forest that uses basic training data (only attributes from original dataset). Red dotted line presents the extended prediction method – this is Random Forest using extended training data with confidence factors. Graphs show Random Forest prediction values without confidence factors as well (orange dotted line). Predicting workload with a Random Forest tested on basic training dataset is significantly worse compared to our approach. The latter can be concluded from the error table as well as graphs.

Thus, some methods indicate that the first few hours can be predicted fairly accurate, which can be seen from visual presentations of predictions. This is to be expected because workload values from a short time period in the past are the most helpful information for predicting future values.

To pursue this further, the classification method with 11 classes (marked as *Clas2*) gives less errors. If we look at the visual presentation (Fig. 9) where an actual prediction curve is shown, the classification method with 11 classes also gives more useful results as far as critical peak workloads. In this case, fluctuations of the predicted workload are closer to those of the actual workload. We have to consider fluctuation as well because it is very important when building a model for preventing the lack of system resources. This means that the classification method with 11 classes would be more appropriate for our model.

Furthermore, we compare our prediction model with the machine-learning method kNN, which is already implemented in the Orange library (Demšar et al., 2013) for Python. We perform tests with different values of kNN's parameter $k$ in the range between 2 and 30. Due to the most accurate prediction results, we set parameter $k$ to 10. The graphs in Fig. 9 show prediction results for the extended prediction method – this is a Random Forest using extended training data (red dotted line) and the kNN prediction method using basic training data (green dotted line).

Tests were made on two different approaches (Clas1 and Clas2) and three different training datasets (3, 5 and 7 months of training data). Predictions made with kNN failed on all training and testing datasets. As is seen from the graphs in Fig. 9, kNN predictions are very unreliable and deviate significantly from the true values. We can then conclude that predicting workload with kNN is not suitable for our observed system.

With experimental results on different training and testing data, we have shown that the use of AME-WPC, which considers our own pattern matching method (TPM) and Random Forest classifier with extended training dataset and confidence factors result in better



Please cite this article as: Cetinski K, Jurič MB. AME-WPC: Advanced model for efficient workload prediction in the cloud. Journal of Network and Computer Applications (2015), http://dx.doi.org/10.1016/j.jnca.2015.06.001

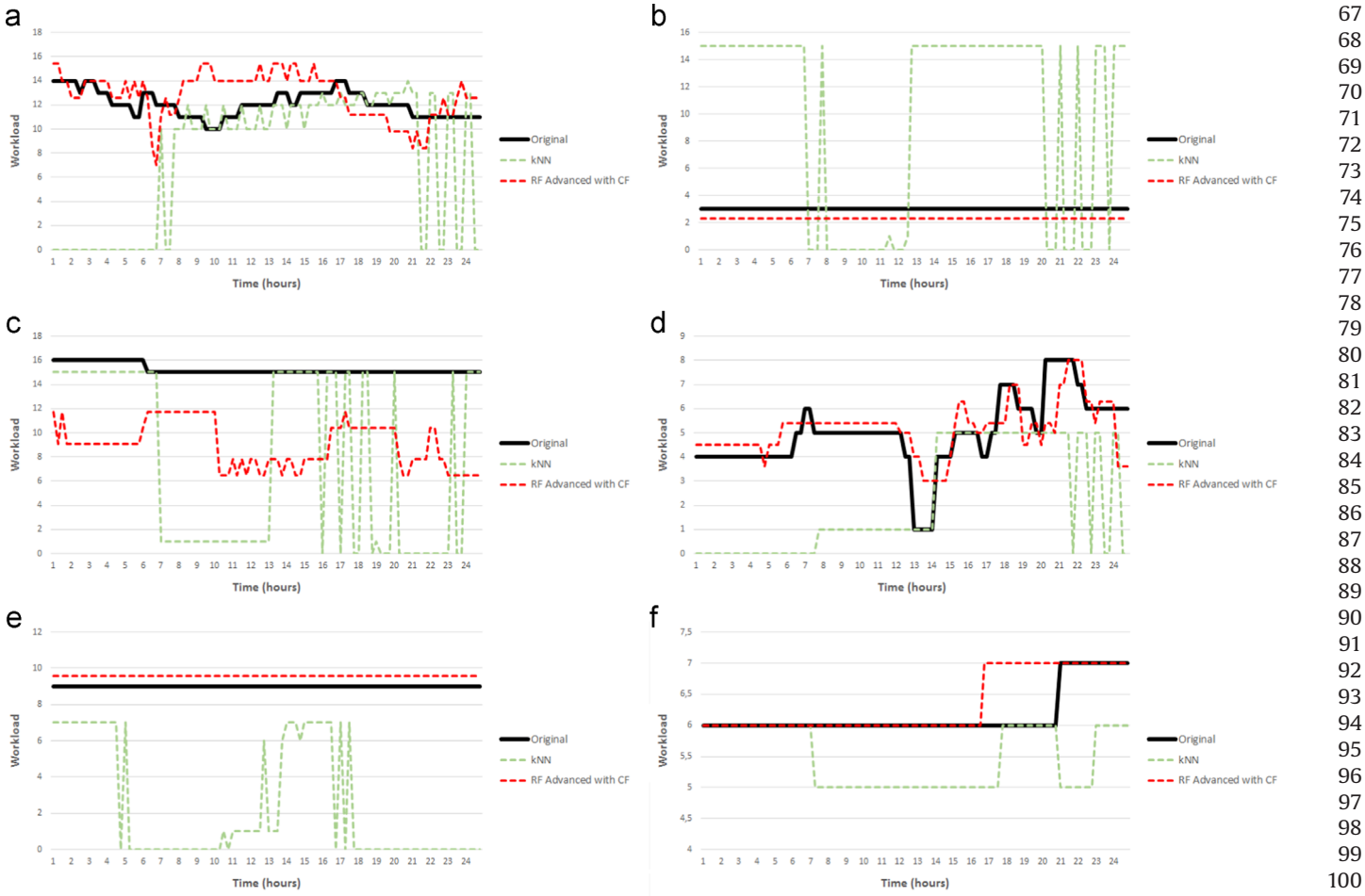


**Fig. 10.** Advanced Random Forest prediction versus kNN prediction for the next 24 h on training data ti1 (day 3), ti2 (day 1) and ti3 (day 2). Clas1 represents classification with 17 classes and Clas2 classification with 11 classes: (a) 3 months, Clas1; (b) 5 months, Clas1; (c) 7 months, Clas1; (d) 3 months, Clas2; (e) 5 months, Clas2; and (f) 7 months, Clas2.

prediction accuracy compared to the use of basic training datasets and the basic prediction methods. Results were presented in the form of MSE's, NMSE's (Fig. 8) and graphs (Figs. 9 and 10) show that prediction accuracy of our proposed model outperforms the basic Random Forest and kNN methods significantly.

## 6. Conclusion and future work

We believe that the more knowledge we have about the cloud system, the more precisely we can predict its future behavior. It is therefore necessary to consider historical workload, the type of data that describes system resources, connections among this data and the current state of the system (workload), upcoming events, projects and other factors that influence on the system workload.

In this research paper we presented an Advanced Model for Efficient Workload Prediction in the Cloud (AME-WPC), which combines statistical and learning methods. In order to improve the capabilities of the learning method, we proposed domain-specific database extensions, which we defined through analysis of the system factors that have a strong influence on the application workload (e.g. part of day, holidays and weekends). Additional database extensions were defined using a novel Two-phase Pattern Matching method (TPM). TPM covers two phases: it recognizes similar patterns based on workload value and similar patterns based on workload fluctuation. We addressed the workload prediction problem with classification as well as regression and tested our solution with the machine-learning method of Random Forest on both – basic and extended – training data. Finally, we evaluated our proposed model capabilities on the AuverGrid workload data series, which we obtained from the Grid Workloads Archive (http://gwa.ewi.tudelft.nl/). For evaluation purposes, we included the machine-learning method kNN. Experimental results demonstrated that the use of TPM and Random Forest with extended learning datasets significantly improve prediction accuracy of workload over time. Thereby, our approach can be considered efficient in terms of enabling better resource management and optimal service provisions which result in lower operational costs and a more stable environment.

As part of our future work, we intend to improve our prediction model while developing an advanced model for automatic resource scaling that will make scaling decisions based on a predicted workload. Additionally, we will include detection of events that are considered to influence workload fluctuations. In addition, AME-WPC can be dynamically adapted after a certain period of time through a redefined learning dataset based on most-recent historical workload data in order to further improve prediction accuracy of the model. As a result, we will implement a prototype system whose performance will be tested in real-world scenarios. We believe that the introduction of such systems can bring the automation of cloud management to a whole new level.

## References

Ardagna D, Casolari S, Colajanni M, Panicucci B. Dual time-scale distributed capacity allocation and load redirect algorithms for cloud systems. J Parallel Distrib Comput 2012;72(6):796–808.

Please cite this article as: Cetinski K, Jurič MB. AME-WPC: Advanced model for efficient workload prediction in the cloud. Journal of Network and Computer Applications (2015), http://dx.doi.org/10.1016/j.jnca.2015.06.001

Bacigalupo D, Jarvis S, Nudd G. An investigation into the application of different performance prediction techniques to e-commerce applications. In: Proceedings of the 18th international parallel and distributed processing symposium (IPDPS). IEEE; 2004. p. 248–55.

Bacigalupo DA, Jarvis SA, He L, Spooner DP, Dillenberger DN, Nudd GR. An investigation into the application of different performance prediction methods to distributed enterprise applications. J Supercomput 2005;34(2):93–111.

Bacigalupo DA, van Hemert J, Usmani A, Dillenberger DN, Wills GB, Jarvis SA. Resource management of enterprise cloud systems using layered queuing and historical performance models. In: IEEE international symposium on parallel & distributed processing, workshops and Phd forum (IPDPSW). IEEE; 2010. p. 1–8.

Bacigalupo DA, van Hemert J, Chen X, Usmani A, Chester AP, He L, et al. Managing dynamic enterprise and urgent workloads on clouds using layered queuing and historical performance models. Simul Model Pract Theory 2011;19(6):1479–95.

Bacigalupo D. Performance prediction-enhanced resource management of distributed enterprise systems [Ph.D. thesis]. UK: University of Warwick, Department of Computer Science; 2006.

Ban T, Zhang R, Pang S, Sarrafzadeh A, Inoue D. Referential kNN regression for financial time series forecasting. Neural Inf Process 2013;8226:601–8.

Breiman L. Random forests. Mach Learn 2001;45:5–32.

Buyya R, Yeo CS, Venugopal S, Broberg J, Brandic I. Cloud computing and emerging IT platforms: vision, hype, and reality for delivering computing as the 5th utility. Future Gener Comput Syst 2009;25(6):599–616.

Cao L. Support vector machines experts for time series forecasting. Neurocomputing 2003;51:321–39.

Caron E, Desprez F, Muresan A. Forecasting for grid and cloud computing on-demand resources based on pattern matching. In: Proceedings of the IEEE second international conference on cloud computing technology and science. IEEE; 2010a. p. 456–63.

Caron E, Desprez F, Muresan A. Forecasting for cloud computing on-demand resources based on pattern matching. Technical report; February, 2010b.

Caron E, Desprez F, Muresan A. Pattern matching based forecast of non-periodic repetitive behavior for cloud clients. J Grid Comput 2011;9(1):49–64.

Chang Y-C, Chang R-S, Chuang F-W. A predictive method for workload forecasting in the cloud environment. In: Advanced technologies, embedded and multimedia for human-centric computing; 2014. p. 577–85.

Chen Y, Yang B, Dong J, Abraham A. Time-series forecasting using flexible neural tree model. Inf Sci 2005;174(3–4):219–35.

Cortez P, Rio M, Rocha M, Sousa P. Multi-scale internet traffic forecasting using neural networks and time series methods. Expert Syst 2012;29(2):143–55.

Delft TU. Gwa-t-4 auvergrid. URL ⟨http://gwa.ewi.tudelft.nl/datasets/gwa-t-4-auvergrid⟩.

Delft TU. The grid workloads archive. URL ⟨http://gwa.ewi.tudelft.nl/⟩.

Demšar J, Curk T, Erjavec A, Črt GroupHočevar T, Milutinovič M, et al. Orange: data mining toolbox in Python. J Mach Learn Res 2013;14:2349–53.

Donate JP, Li X, Sánchez GG, de Miguel AS. Time series forecasting by evolving artificial neural networks with genetic algorithms, differential evolution and estimation of distribution algorithm. Neural Comput Appl 2013;22(1):11–20.

Doulamis N, Doulamis A, Litke A, Panagakis A, Varvarigou T, Varvarigos E. Adjusted fair scheduling and non-linear workload prediction for QoS guarantees in grid computing. Comput Commun 2007;30(3):499–515.

Eaton JW. GNU Octave manual. Network Theory Limited; 2002.

Eddahecha A, Chtouroub S, Chtouroua M. Hierarchical neural networks based prediction and control of dynamic reconfiguration for multilevel embedded systems. J Syst Archit 2013;59(1):48–59.

Frank R, Davey N, Hunt S. Time series prediction and neural networks. J Intell Robot Syst 2001;31(1–3):91–103.

Ganapathi A, Yanpei C, Fox A, Katz R, Patterson D. Statistics-driven workload modeling for the cloud. In: 2010 IEEE 26th international conference on data engineering workshops (ICDEW). IEEE; 2010. p. 87–92.

Gmach D, Rolia J, Cherkasova L, Kemper A. Workload analysis and demand prediction of enterprise data center applications. In: Proceedings of the 2007 IEEE 10th international symposium on workload characterization. IEEE; 2007. p. 171–80.

Gong Z, Gu X, Wilkes J. Press: predictive elastic resource scaling for cloud systems; 2010. p. 9–16.

Imam M, Miskhat S. Neural network and regression based processor load prediction for efficient scaling of grid and cloud resources. In: Proceedings of the 14th international conference on computer and information technology (ICCIT); 2011. p. 333–8.

Imandoust SB, Bolandraftar M. Application of K-nearest neighbor (KNN) approach for predicting economic events: theoretical background. Int J Eng Res Appl 2013;3(5):605–10.

Islam S, Keung J, Lee K, Liu A. Empirical prediction models for adaptive resource provisioning in the cloud. Future Gener Comput Syst 2012;28(1):155–62.

Jones E, Oliphant T, Peterson P, et al. SciPy: open source scientific tools for Python; 2001.

Kalantari M, Akbari M. Grid performance prediction using state-space model. Concurr Comput: Pract Exp 2009;21(9):1109–30.

Kalekar PS. Time series forecasting using Holt–Winters exponential smoothing. Technical Report; 2004.

Khan A, Anerousis N. Workload characterization and prediction in the cloud: a multiple time series approach. In: 2012 IEEE network operations and management symposium. IEEE; 2012. p. 1287–94.

Kononenko I. Estimating attributes: analysis and extensions of relief. In: Machine learning: ECML-94. Springer; 1994. p. 171–82.

Li S-T, Cheng Y-C. A stochastic HMM-based forecasting model for fuzzy time series. IEEE Trans Syst Man Cybern Part B. Cybern 2010;40(5):1255–66.

Li Q, Hao Q-F, Xiao L-M, Li Z-J. An integrated approach to automatic management of virtualized resources in cloud environments. Comput J 2011;54(6):905–19.

Li T. A hierarchical framework for modeling and forecasting web server workload. J Am Stat Assoc 2005;100(471):748–63.

Liu X, Ni Z, Yuan D, Jiang Y, Wu Z, Chen J, et al. A novel statistical time-series pattern based interval forecasting strategy for activity durations in workflow systems. J Syst Softw 2011;84(3):354–76.

Manvi SS, Shyam GK. Resource management for infrastructure as a service (IaaS) in cloud computing: a survey. J Netw Comput Appl 2014;41:424–40.

Mentzer J, Moon M. Sales forecasting management. Thousand Oaks; 2004. p. 73–112.

Montes J, Sánchez A, Pérez MS. Grid global behavior prediction. In: 2011 11th IEEE/ACM international symposium on cluster. Cloud and grid computing. IEEE; 2011. p. 124–33.

Quiroz A, Kim H, Parashar M, Gnanasambandam N, Sharma N. Towards autonomic workload provisioning for enterprise grids and clouds. In: Proceedings of the 2009 10th IEEE/ACM international conference on grid computing. IEEE; 2009. p. 50–7.

Robnik-Šikonja M. Improving random forests. Lecture Notes in Computer Science, vol. 3201. 2004. p. 359–70.

Roy N, Dubey A, Gokhale A. Efficient autoscaling in the cloud using predictive models for workload forecasting. In: Proceedings of the 2011 IEEE 4th international conference on cloud computing. IEEE; 2011. p. 500–7.

Saadatfar H, Fadishei H, Deldari H. Predicting job failures in auvergrid based on workload log analysis. New Gener Comput 2012;30(1):73–94.

Sarikaya R, Isci C, Buyuktosunoglu A. Runtime workload behavior prediction using statistical metric modeling with application to dynamic power management. In: 2010 IEEE International Symposium on Workload Characterization (IISWC); 2010. p. 1–10.

Sun YS, Chen Y-F, Chen MC. A workload analysis of live event broadcast service in cloud. Procedia Comput Sci 2013;19:1028–33. [In: The 4th International Conference on Ambient Systems, Networks and Technologies (ANT 2013), the 3rd International Conference on Sustainable Energy Information Technology (SEIT-2013)].

Troncoso Lora A, Riquelme Santos JM, Riquelme JC, Gómez Expósito A, Martínez Ramos JL. Time-series prediction: application to the short-term electric energy demand. Curr Top Artif Intell 2004;3040:577–86.

Vercauteren T, Aggarwal P. Hierarchical forecasting of Web server workload using sequential Monte Carlo training. IEEE Trans Signal Process 2007;55(4):1286–97.

Wu Y, Hwang K, Yuan Y. Adaptive workload prediction of grid performance in confidence windows. IEEE Trans Parallel Distrib Syst 2010;21(7):925–38.

Yang J, Liu C, Shang Y, Cheng B, Mao Z, Liu C, et al. A cost-aware auto-scaling approach using the workload prediction in service clouds. Inf Syst Front 2014;16(1):7–18.

Zhang H, Jiang G, Yoshihira K, Chen H, Saxena A. Intelligent workload factoring for a hybrid cloud computing model. In: Proceedings of the 2009 congress on services —I. IEEE; 2009. p. 701–8.

Zhang GP. Time series forecasting using a hybrid ARIMA and neural network model. Neurocomput 2003;50:159–75.



Please cite this article as: Cetinski K, Jurič MB. AME-WPC: Advanced model for efficient workload prediction in the cloud. Journal of Network and Computer Applications (2015), http://dx.doi.org/10.1016/j.jnca.2015.06.001