# Adaptive Workload Prediction for Proactive Auto Scaling in PaaS Systems

R.S. Shariffdeen, D.T.S.P. Munasinghe, H.S. Bhathiya, U.K.J.U. Bandara, and H.M.N. Dilum Bandara
*Department of Computer Science & Engineering*
*University of Moratuwa*
*Moratuwa, Sri Lanka*
{*ridwan.11, tharindu.11, bhathiya.11, bandaraukju.11, dilumb*}*@cse.mrt.ac.lk*

***Abstract*—Elasticity is a key feature of cloud computing where resources are allocated and released according to user demands. Reactive auto scaling, in which the scaling actions take place just after meeting the triggering thresholds, suffers from several issues like risk of under provisioning at peak loads and over provisioning during other times. Proactive scaling solutions, where future resource demand can be forecast and necessary scaling actions enacted beforehand, can overcome these issues. Nevertheless, the effectiveness of such proactive scaling solutions depends on the accuracy of the prediction method(s) adopted. We propose a forecasting technique to enhance the accuracy of workload forecasting in cloud auto-scalers. An ensemble workload prediction mechanism based on time series and machine learning techniques is proposed to make more accurate predictions on drastically different workload patterns. In this work, we initially evaluated several forecasting models for their applicability in forecasting different workload patterns. The proposed ensemble technique is then implemented using three well-known forecasting models and tested for three real-world workloads. Simulation results show that our ensemble method produces significantly lower forecast errors compared to the use of individual models and the prediction technique employed in Apache Stratos, an open source PaaS platform.**

***Keywords*–auto scaling; cloud computing; time series analysis; workload prediction**

## I. Introduction

Cloud computing has already gained ground with advantages like scalability, on-demand resource provisioning, and high availability. According to the NIST definition [1], cloud computing refers to the delivery of computing resources, such as networks, servers, storage, applications, and platforms, over the network based on user demand. Based on this definition, it is evident that the elasticity is a major requirement of any cloud platform, which is often achieved via an auto scaling process. Auto scaling refers to the dynamic allocation and release of resources for a cloud application, in order to optimize its resource utilization while minimizing the cost, as well as achieving the desired Quality of Service (QoS) and availability goals [2], [3].

Unlike in the case of IaaS, if the application itself runs on a PaaS, the burden of auto scaling is delegated from the developer to the PaaS provider. Operating at the PaaS layer enables the auto-scaling services to use high level, application-specific metrics such as the number of requests in flight, as well as low level, cloud-specific metrics such as CPU and memory utilization. The same PaaS platform may run vastly different applications with different workload patterns, complicating the auto-scaling process.

Auto scaling can be achieved via either reactive or proactive approaches. In reactive, threshold-based auto scaling, users have to specify thresholds for workload metrics, and scaling would occur only after such thresholds are exceeded [4]. These thresholds must be high enough for efficient resource utilization, as well as low enough to compensate for delays in formulating and executing scaling actions. Although this seems to be the most customizable approach from the user's perspective, it has several weaknesses [5] such as the inability to adapt to workload patterns (e.g., thrashing during load fluctuations), low resource utilization and higher cost under smaller thresholds, and the risk of service/QoS degradation under larger thresholds and rapidly increasing workloads. Coming up with an optimum threshold in a reactive auto-scaling system is nontrivial and requires application-specific experience and expertise.

In the proactive approach, resource requirements for the future time horizon are forecasted based on the demand history. With accurate predictions, an application can take early scaling decisions so that when the workload reaches a particular level, the required resources would have already been allocated. Consequently, resource utilization can be safely increased to a maximum level. However, the reliability of such an approach depends mainly on the accuracy of the predicted values. Because we forecast the future workload requirement from historical data of the respective measures, time series forecasting is applicable for this scenario.

We have identified the following challenges specific to workload prediction for auto scaling:

- A PaaS cloud system may be used to build different applications with vastly different workload patterns. Hence, the workload prediction model should not tend to get overfitted to a specific workload pattern.
- As the workload dataset grows with time, the predictive model should evolve and continuously learn the latest workload characteristics.
- The workload predictor should be able to produce results within a bounded time.
- Given that the time horizon for the prediction should be chosen based on the physical constraints like up-time and graceful shutdown time of Virtual Machines

(VMs), the predictor should be able to produce sufficiently accurate results over a sufficiently large time horizon.

The objective of this work is to come up with a prediction method that can be trained in real-time to capture the latest trends and provide sufficiently accurate results for drastically different workload patterns. First, we evaluate the ability of existing models to produce accurate real-time predictions against evolving datasets. In this evaluation we tested time series forecasting methods, including statistical methods like ARIMA and exponential model, machine learning models like neural networks, and a prediction method used in Apache Stratos, an open source PaaS framework. Through this, we demonstrate the limitations in existing solutions in accurately predicting real-world workload traces, and highlight the need for more accurate predictions. Next, we propose an ensemble technique which combines results from a neural network, ARIMA and exponential models. Based on simulations with three publicly available workload traces (two real-world cloud workload datasets and the Google cluster dataset), we demonstrate that the proposed ensemble model outperforms each of the tested individual models.

The rest of the paper is structured as follows: Section 2 outlines related work on cloud workload prediction and their limitations. Section 3 describes and evaluates several existing time-series forecasting techniques while exploring their ability to provide accurate predictions on publicly available datasets with predictable patterns under real-time training scenarios. Section 4 introduces the proposed ensemble technique and prediction algorithm. Section 5 presents the performance analysis, and concluding remarks are presented in Section 6.

## II. Related Work

There is a significant number of already established research work in the workload prediction domain. Kupferman et al. [6] applied single order auto-regression to predict the request rate and found that its accuracy depends on several parameters such as the size of the input window and the horizon window. Exponential smoothing is popularly used for prediction. Mi et al. [7] also used quadratic exponential smoothing against real workload traces such as World Cup 98 [8], and showed good results. Auto-Regressive Moving Average (ARMA) method is one of the dominant time series analysis techniques for workload and resource usage prediction. Roy et al. [3] used a second order ARMA filter for workload prediction on the World Cup 98 traces and showed accurate results. Bunch et al. [9] used an exponential smoothing algorithm to forecast how many requests to expect and how many requests will be enqueued for the next $t$ seconds for an auto scaler in PaaS cloud.

Machine learning techniques like neural networks, regression, and Hidden Markov Model (HMM) have been applied by several authors for workload prediction. Yang et al. [10] have used a sliding window based Linear Regression Model (LRM) for workload prediction and showed a lower prediction deviation. Khan et al. [11] used Hidden Markov Model to explore the temporal correlations in workload pattern changes. Several work also focus on using history window values as the input for a neural network [12]. However, the accuracy of such method depends on the input window size.

The related work outlined above primarily focus on a specific dataset. Hence, the resulting predictive model might not be a good fit for scenarios involving multiple applications with drastically different workload characteristics. In some of the solutions the predictive model has been derived using offline training on datasets, so that dynamic adaptation to current workload changes may not be possible. However, these may not work well on PasS platforms as the same platform may be used to build different applications with drastically different workload characteristics.

From a mathematical perspective, certain preconditions have to be fulfilled by a given dataset for it to be fitted satisfactorily using time-series prediction techniques like exponential moving average and ARIMA. It is quite unlikely that datasets from all applications will meet these preconditions. Therefore, a given prediction technique may require its parameters to be adjusted to fit to specific datasets. Hence, identifying optimum parameters for the model should happen dynamically via an online training process, rather than using predefined constants defined during the offline training.

## III. Evaluation of Existing Models

In this analysis, we analyze the ability of several prediction techniques to learn from the current workload history, and predict future workload while the workload itself is evolving. We simulated an online training scenario by streaming data points of the workload time series one at a time to each prediction technique. We then plotted the one-step lookahead predictions from the resulting prediction technique at each stage, against the real data points. For evaluation purpose, we used the forecast package in R [13] which contains time-series datasets in different domains with typical patterns like trends, cycles and seasonality factors.

### A. Datasets

The objective of this evaluation is to identify how well an individual prediction method can make predictions based only on past data. Here we used several public datasets with predictable patterns from the R forecast package, some of which are not directly related to cloud workloads. Selection of multiple datasets enables us to evaluate the time-series prediction techniques under several probable workload patterns. Nevertheless, the experimental results in Section 5, where we compare the performance of the proposed ensemble technique vs. existing methods, are based only on two real-world cloud workloads and the Google cluster dataset, in addition to those found in the forecast package.

Following datasets from the R forecast package is selected for the evaluation:

- euretail – Quarterly retail trade index in the Euro area (17 countries) from 1996 - 2011, covering wholesale and retail trade, and repair of motor vehicles and motorcycles (Index: 2005 = 100).
- sunspotarea – Annual averages of the daily sunspot areas (in units of millionths of a hemisphere) for the full sun.
- oil – Annual oil production (millions of tonnes) from Saudi Arabia between 1965 - 2010.

In addition, as a representative of real-world workloads, we have used a CPU usage trace obtained from a dedicated standalone server from a private cloud. The server has a Linux-based operating system and is equipped with four Quad-Core AMD Opteron CPUs and 16 GB of RAM. We have also obtained a trace of queued HTTP requests over time by replaying an access log of a content-based website against a hosted copy of itself.

## B. Time Series Forecasting Techniques

We chose three widely used prediction methods from literature, namely ARIMA, neural network, and exponential models, as well as the existing workload prediction technique in Apache Stratos.

*1) ARIMA:* Autoregressive Integrated Moving Average (ARIMA) model combines two models called autoregression (of order $p$) and moving average (of order $q$).

*Autoregression Model AR(p)* – In an autoregression model, the variable of interest is forecast using a linear combination of past values of the variable. The term autoregression indicates that it is a regression of the variable against itself. Thus, an autoregressive model of order $p$ can be written as:

$$x_t = c + \phi_1 x_{t-1} + \phi_2 x_{t-2} + ... + \phi_p x_{t-p} + e_t \tag{1}$$

where $c$ is a constant and $e_t$ is white noise. Autoregressive models are typically restricted to stationary data, and some constraints are applied on the values of the parameters [14].

*Moving Average Model MA(q)* – This model uses past forecast errors in a regression-like model.

$$x_t = c + \theta_1 e_{t-1} + \theta_2 e_{t-2} + ... + \theta_q e_{t-q} \tag{2}$$

$x_t$ can be thought of as a weighted moving average of the last few forecast errors. By combining differencing with autoregression and a moving average model, non-seasonal ARIMA model can be obtained. The full model can be written as:

$$\begin{aligned} x'_t = c + \phi_1 x'_{t-1} + \phi_2 x'_{t-2} + ... + \phi_p x'_{t-p} \\ + e_t + \theta_1 e_{t-1} + \theta_2 e_{t-2} + ... + \theta_q e_{t-q} \end{aligned} \tag{3}$$

The *predictors* on the right hand side include both the lagged values of $X_t$ and lagged errors. This is called an ARIMA($p$, $d$, $q$) model, where $p$ and $q$ are the orders of the autoregressive and moving average parts respectively, and $d$ is the degree of first differencing involved.

*2) Neural Networks:* Artificial Neural Networks (ANNs) are a class of nonlinear, and data-driven models. In the domain of time series forecasting, neural network is an excellent alternative for existing statistical models which require some assumptions to be satisfied in the time series. Ability of modeling nonlinear relationships is the major advantage in neural networks. Auto regressive feed forward neural networks [14] are the specialized neural networks in the time series forecasting domain. They consist of a feed forward structure of three types of layers, an input layer, one or more hidden layer, and an output layer. Each layer consists with nodes, to those in the immediate next layer by acyclic links which have associated weights. In auto-regressive neural networks, lagged values of the time series (input window) are injected as the inputs and train the weights of the model to forecast for a given time horizon by using the past history of the dataset.

*3) Exponential Model:* Several selected versions of exponential models are presented next.

*Exponential Weighted Moving Average (EWMA)* – In EWMA methods, forecasts are calculated using weighted averages where the weights decrease exponentially as the observations come from further in the past. The smallest weights are associated with the oldest observations [14], [15], [16]. Exponential smoothing can be indicated by the following equation:

$$\hat{x}_{t+1|t} = \alpha x_t + \alpha(1-\alpha)\hat{x}_{t|t-1} \tag{4}$$

$$\hat{x}_{t+1|t} = \alpha x_t + \alpha(1-\alpha)x_{t-1} + \alpha(1-\alpha)^2 x_{t-2} + ... \tag{5}$$

where $0 \leq \alpha \leq 1$ is the smoothing parameter. $\hat{x}_{t+1|t}$, the one-step-ahead forecast for time $(t+1)$, is a weighted average of all the observations in the series $x_1$, ..., $x_t$. The rate at which the weights decrease is controlled by the parameter $\alpha$. If $\alpha$ is small (i.e., close to 0), more weight is given to observations from the distant past. If $\alpha$ is large (i.e., close to 1), more weight is given to the more recent observations. This method is suitable for forecasting data with no trend or seasonal patterns.

Following variants of exponential smoothing are suitable for forecasting time series with different characteristics [14]:

- *Double Exponential Smoothing* – Applicable to a time series with a linear trend.
- *Triple Exponential Smoothing* – Applicable to a time series with a trend and seasonality.
- *Holt's Linear Trend Model* – Extended simple exponential smoothing to allow forecasting of data with a trend.
- *Exponential Trend Model* – Applicable when the trend in the forecast function is can be exponential rather than linear.
- *Damped Trend Method* – Dampens the trend to a flat line some time in the future to model the general nature of practical time series.
- *Holt-Winters Seasonal Method* – Applicable to a time series with adaptive or multiplicative trends and seasonality.

*4) Prediction method in Apache Stratos:* Apache Stratos [17] is an open source PaaS framework where a user can build a PaaS system on top of an existing IaaS cloud. The autoscaling solution in Stratos uses a workload demand prediction method which can be explained using the motion equation in physics. The method calculates one lookahead prediction using the workload demand in the last period as follows [17]:

$$S_{t+h|t} = u_t h + \frac{1}{2} a_t h^2 \tag{6}$$

$$\hat{x}_{t+h} = avg(x)_t + S_{t+h|t} \tag{7}$$

where $S_{t+h|t}$ is the predicted change of workload for time $(t+h)$, $x$ is the workload metric considered, $\hat{x}_{t+h}$ is the predicted workload metric for time $(t+h)$, $u_t$ and $a_t$ are the first and second derivatives of the metric within the $t$-th interval, and $h$ is the time horizon considered.

## C. Time Series Forecasting Library

We used the following time series forecasting models in R forecast package [13] to implement each of the selected forecasting solution:

$auto.arima()$ – Finds the best fitted ARIMA model and approximates the model coefficients by minimizing the past forecast error [14].

$ets()$ – Finds out the best fitted exponential model among the family of exponential models (simple exponential smoothing, Holt's linear trend, exponential trend, and Holt-Winters seasonal method) automatically and calculates the model coefficients by reducing the past forecast error [14].

$nnetar()$ – Feed-forward neural networks with a single hidden layer and lagged inputs for forecasting univariate time series.

## D. Error Measures

For quantitative analysis, we used the following two error measures:

- *Root Mean Squared Errors (RMSEs)* are of the same scale as the data. As it is scale dependent, it cannot be used to make comparisons between series that are of different scales [14]. RMSE can be represented as follows:

$$RMSE = \sqrt{\frac{\sum_{t=1}^{n}(\hat{x}_t - x_t)^2}{n}}$$

where $x_t$ is the real value at time $t$, $\hat{x}_t$ is the forecast at time $t$ and $n$ is the number of data points in dataset.

- *Percentage errors* have the advantage of being scale-independent, but they overemphasize the errors for values that are actually small. Percentage errors can be represented as follows:

$$MAPE = \frac{100}{n}\sum_{t=1}^{n}\left|\frac{\hat{x}_t - x_t}{x_t}\right|$$

where $x_t$ is the real value at time $t$, $\hat{x}_t$ is the forecast at time $t$ and $n$ is the number of data points in dataset.

## E. Observations

Fig. 1 to 5 show the comparison of each of the time-series prediction models under five datasets. Solid lines represent the actual time series whereas dashed lines represent the predicted values from each method. The respective error measures are presented in Tables 1 and 2. The bold cells represent the minimum RMSE and MAPE values, while the underlined cells represent the maximum RMSE and MAPE values.

According to the results obtained, there is no single model which performs well in all online training scenarios. Each model will fit the datasets which satisfy its assumptions. Where they are not satisfied, performance of some models would be below the average. For example, while neural networks perform well on the *oil* dataset, they perform poorly on *euretail* dataset. The current prediction method from Stratos has the highest mean errors almost all the cases considered.

The analysis primarily depicts that there is no single method which guarantees best performance in all cases. A method which performs well in some dataset might perform worse in another dataset. As an autonomous PaaS auto-scaler should be able to work with any type of workload pattern, the prediction method should be able to give good enough estimates in an average case without producing large errors on any specific workload pattern.

Table I. COMPARISON OF ERRORS FOR STANDARD DATASETS USING RMSE AND MAPE

| Model | sunspotarea | | Euretail | | oil | |
|---|---|---|---|---|---|---|
| | RMSE | MAPE | RMSE | MAPE | RMSE | MAPE |
| ARIMA | **382.360** | 1.359 | **0.524** | **0.004** | 55.313 | 0.251 |
| Exponential | 505.750 | 1.161 | 0.576 | 0.011 | 54.989 | 0.250 |
| Neural net. | 473.924 | **0.465** | 1.882 | 0.006 | **51.616** | **0.160** |
| Current | 546.938 | 0.965 | 0.650 | **0.004** | 61.807 | 0.585 |

Table II. COMPARISON OF ERRORS FOR CLOUD DATASETS USING RMSE AND MAPE

| Model | Memory | | CPU | |
|---|---|---|---|---|
| | RMSE | MAPE | RMSE | MAPE |
| ARIMA | 7.238 | 0.136 | 2.976 | 0.036 |
| Exponential | **7.005** | 0.160 | 3.150 | 0.048 |
| Neural net. | 8.169 | **0.135** | **2.792** | 0.031 |
| Stratos | 9.928 | 0.172 | 5.692 | **0.024** |

The idea of ensemble learning is quite frequently used in situations where there is no dominant technique which can provide the best results, but it is possible to obtain good enough results by combining results from several weak learners. In the general forecasting domain we can see several models combining techniques proposed by several researchers [18], [19], [20]. There are several views regarding model selection and combining multiple models. While some researchers claim that combined methods improve accuracy, others claim that they suppress the effects of large errors from individual models rather than improving the overall accuracy. According to the literature, there are multiple ways of combining individual results [21]:

- Simple average – Obtaining the average of the forecast from each model as the forecast of ensemble.
- Median-based – Obtaining the median of the forecast from each model as the forecast of ensemble.
- Trimmed average – Obtaining the average of forecasts while excluding the worst performing models.
- Error-based combining – Assigning the weight of each model to be the inverse of the past forecast error (e.g., MAE, MAPE, or RMSE) of each model.
- Variance-based method – Determining the optimal weights through minimization of the total sum of squared error.

## IV. PROPOSED ENSEMBLE PREDICTION METHOD

We propose an error-based ensemble technique for workload prediction. Our approach tries to address situations where offline training is not possible due to workload history data not being available at the beginning of the prediction process. While the auto scaler is in operation, workload history gets accumulated based on the user's workload requirement (e.g., CPU, memory, and request count). However, the prediction method should still be able to predict the future time horizon based on the initially available dataset. After the initial

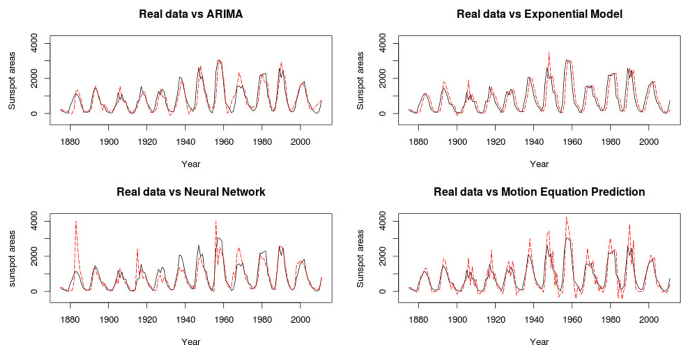

Figure 1. Comparison of predictions for sunspotarea dataset.

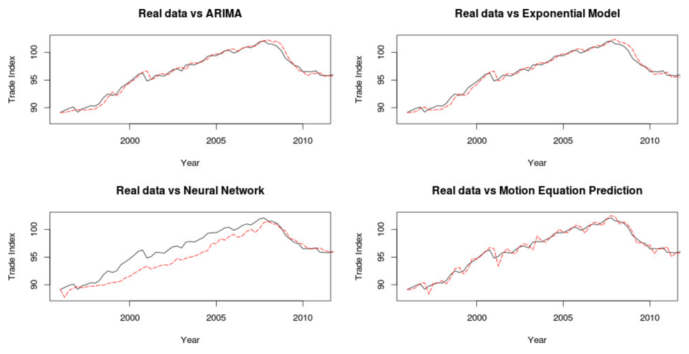

Figure 2. Comparison of predictions for euretail dataset.

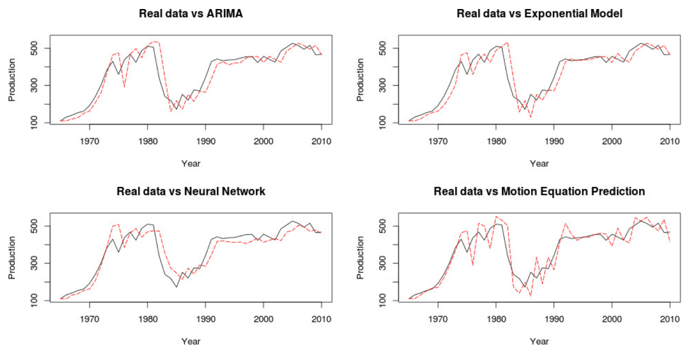

Figure 3. Comparison of predictions for oil dataset.

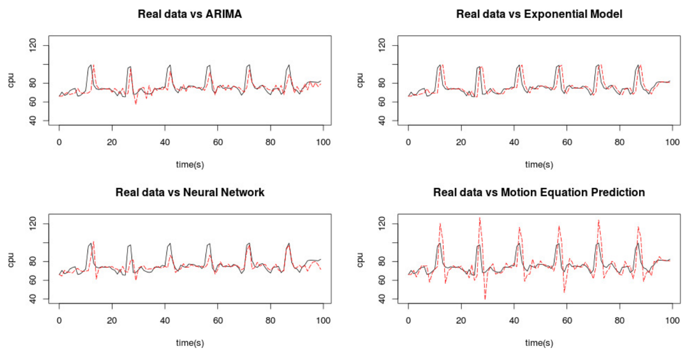

Figure 4. Comparison of predictions for CPU utilization

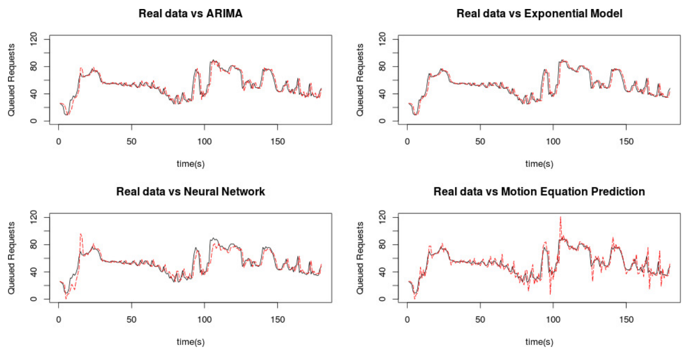

Figure 5. Comparison of predictions for queued HTTP request count.

predictions, actual data will become available for the next time periods, so that we can accumulate the latest actual data into the workload history and use them for the next forecast horizon.

In existing error-based combining techniques, mean values of error metrics (e.g., absolute error, absolute percentage error, squared error, etc.) are taken into account while calculating the contributing factors for individual methods.

Considering the currently available dataset in the time series as $X = [x_1, x_2, ..x_t]$ we want to calculate predictions for $x_{t+h}$. Let the final predicted value be $\hat{x}_{t+h}$ and the predicted value from the $i$-th model for $x_{t+h}$ be $\hat{x}^{(i)}_{t+h}$. We define $\hat{x}_{t+h}$ as a weighted sum of predictions from each model:

$$\hat{x}_{t+h} = \sum_{i=1}^{k} w_i \hat{x}^{(i)}_{t+h} \quad \forall k \in \{1, 2, 3, ..., n\} \tag{8}$$

where $w_i$ is the weight assigned to the $i$-th forecasting method. To ensure unbiasedness, it is often assumed that the weights add up to unity [21]. Hence, we define the contribution from the $i$-th model to the final result as $c_i$, so that the same equation can be redefined as:

$$\hat{x}_{t+h} = \frac{\sum_{i=1}^{k} c_i \hat{x}^{(i)}_{t+h}}{\sum_{i=1}^{k} c_i} \tag{9}$$

where $w_i = \frac{c_i}{\sum_{j=1}^{k} c_j}$. As $\sum_{j=1}^{k} w_j = 1$, this weight assignment is unbiased and would result in the weighted average of the predictions.

## A. *Determination of Contribution Coefficients*

To determine the optimal contribution coefficients in our proposed ensemble technique, we use past forecast errors for each model. In our prediction problem, accuracy of the prediction of the next time horizon is more significant. Hence, the contributions are calculated using inverses of the past forecasting error measures. There are various choices for error measurement, and next we discuss several selected measures.

*1) Error of the Most Recent Prediction:* This model only captures how accurately the last prediction has been calculated by the model. Hence, it does not capture the overall accuracy. If the model has repeatedly made large errors in past predictions, except the most recent one, contribution coefficients are biased and may lead to error-prone decisions.

*Absolute Error of the Last Prediction* can be defined as follows:

$$AE_t = |\hat{x}_t - x_t|$$

*Squared error of the Last Prediction*, which penalizes errors more efficiently than the absolute error, can be defined as follows:

$$SE_t = (\hat{x}_t - x_t)^2$$

Whereas *Absolute Percentage/Relative Error* is sensitive to errors related to values of small magnitudes and is defined as:

$$APE_t = 100 \left| \frac{\hat{x}_t - x_t}{x_t} \right|$$

*2) Average Error Over Prediction History:* This measure captures the overall error in past predictions, where all past errors have the same level of significance. If a method has produced larger errors in past predictions, the contribution may be reduced significantly even though it may be producing the best predictions for more recent values.

In this case *Mean Absolute Error/Root Mean Squared Error* can be defined as:

$$RMSE = \sqrt{\frac{\sum_{t=1}^{n}(\hat{x}_t - x_t)^2}{n}}$$

Whereas *Mean Absolute Percentage Error* is defined as:

$$MAPE = \frac{100}{n} \sum_{t=1}^{n} \left| \frac{\hat{x}_t - x_t}{x_t} \right|$$

While calculating the contributions based on errors, models whose errors are based on the last observation overlook overall accuracy and assign high importance to the last prediction error. On the other extreme, the average errors assume that all the last prediction errors have the same level of significance. However, what we really need is an error measure which has a larger contribution from the errors in more recent predictions and smaller contributions from early predictions. Exponential smoothing provides the best fit for our requirement, so that we can calculate the contribution as the errors are fitted under the exponential model. Here, the contribution coefficients $c_{i,t}$ are calculated from the inverses of the fitted past forecast errors ($e_t$). If $b_{i,t}$ is the fitted value of the past forecasting error from the $i$-th model at the $t$-th time interval, $c_{i,t} = \frac{1}{b_{i,t}}$. $e_t$ can be the absolute error, squared error, or absolute relative error at $t$-th prediction. $b_{i,t}$ can be defined as follows:

$$b_{i,t} = \alpha e_{(i,t)} + (1-\alpha) b_{(i,t-1)} \tag{10}$$

where $0 \leq \alpha \leq 1$

$$b_{i,t} = \alpha e_{(i,t)} + \alpha(1-\alpha) e_{(i,t-1)} + \alpha(1-\alpha)^2 e_{(i,t-2)} + ...$$

$$c_{i,t} = \frac{1}{b_{i,t}} \tag{11}$$

## *B. Selection of Models*

The above mentioned error-based weighting mechanism can be used with different combinations of forecasting models. To preserve the ability to cope up with drastically different workload patterns, models used in ensemble techniques should conceptually be different and capture the different characteristics of datasets.

ARIMA assumes a linear form of the model, i.e., a linear correlation structure is assumed among the time-series values. Therefore, it cannot capture non-linear patterns [19]. Alternatively, seasonal ARIMA models can fit the seasonal factors of the time series accurately.

A neural network can capture complex nonlinear relationships within a time series. It has a data driven approach which can extract patterns within a time series without predefined knowledge on the relationships inside data. But neural networks require sufficiently large amounts of data points to accurately identify patterns. Observations in Section 3 also show that neural networks make larger errors in initial stages due to lack of data, but perform well after they correctly identify the relationships. Time to train a neural networks is also significant factor. As we propose real-time training, the data point history should not be allowed to grow arbitrarily large, which would slow down the prediction process.

Contrary to the general belief that ARIMA is more general than exponential models, there are no equivalent ARIMA models for some of the nonlinear exponential models (e.g., the exponential trend model) [14]. Therefore, to preserve generality, we also choose the exponential model as part of our ensemble solution.

There is a possibility that the above models may produce out-of-range forecasts, especially in the case of early-stage neural networks. To compensate for this, we use a naive forecast model which forecasts the last known data point for the next interval as well.

## *C. Proposed Prediction Algorithm*

Proposed workload prediction technique can be explained using the following algorithm:

1) Consider the time-series history window at time $t$, $X = [x_1, x_2, ..x_t]$.
2) Calculate forecast value from the $i$-th time-series forecasting method over the time horizon $h$, $\hat{x}_{t+h}^{(i)} \forall i \in \{1, 2, 3, ..., k\}$, where $k$ is the number of forecasting methods used.
3) Fit a history window for the last $t$ actual data points using the $i$-th method.
4) Use an exponential smoothing model to fit the errors resulting from Step 3, and use them to calculate the contribution factor for the $i$-th model at $t$, $c_{(i,t)}$.
5) Calculate the point forecast for time $(t+h)$ using $\hat{x}_{t+h} = \frac{\sum_{i=1}^{k} c_{(i,t)} \hat{x}_{t+h}^{(i)}}{\sum_{i=1}^{k} c_{(i,t)}}$.
6) At time $(t+1)$, actual value for time $(t+1)$ will be available. Add this value to the history window $X = X \cup \{x_{t+1}\}$ and repeat from Step 2.

Table III. Comparison of performance on standard datasets. The ARMA-based model has been taken from [3].

| Model | sunspotarea | | euretail | | oil | |
|---|---|---|---|---|---|---|
| | RMSE | MAPE | RMSE | MAPE | RMSE | MAPE |
| ARIMA | 382.360 | 1.359 | **0.524** | 0.004 | 55.313 | 0.251 |
| Exponential | 505.750 | 1.161 | 0.576 | 0.011 | 54.989 | 0.251 |
| Neural net. | 473.924 | 0.465 | 1.882 | 0.006 | 51.616 | **0.160** |
| Stratos | 546.938 | 0.965 | 0.650 | 0.004 | 61.807 | 0.585 |
| ARMA-based model | 530.160 | 1.181 | 0.600 | 0.004 | 51.986 | 0.250 |
| Mean ensemble | 369.525 | 0.519 | 0.697 | **0.003** | **49.922** | 0.227 |
| Median ensemble | 397.439 | 0.777 | 0.531 | 0.004 | 50.046 | 0.250 |
| Proposed ensemble | **356.315** | **0.393** | 0.537 | **0.003** | 50.147 | 0.256 |

Table IV. Comparison of performance on cloud datasets.

| Model | Google Cluster | | Memory | | CPU | |
|---|---|---|---|---|---|---|
| | RMSE | MAPE | RMSE | MAPE | RMSE | MAPE |
| ARIMA | 12.963 | 0.051 | 7.238 | 0.136 | 2.976 | 0.036 |
| Exponential | 12.886 | 0.041 | 7.005 | 0.160 | 3.150 | 0.048 |
| Neural net. | 12.530 | 0.036 | 8.169 | 0.135 | **2.792** | 0.031 |
| Stratos | 19.757 | 0.116 | 9.928 | 0.172 | 5.692 | 0.024 |
| ARMA-based model | 12.549 | 0.069 | 7.185 | 0.180 | 3.477 | **0.023** |
| Mean Ensemble | 12.099 | 0.051 | 7.036 | 0.130 | 2.900 | 0.029 |
| Median Ensemble | 12.059 | 0.055 | 7.010 | 0.141 | 2.944 | 0.028 |
| Proposed Ensemble | **11.934** | **0.027** | **6.972** | **0.129** | 2.873 | 0.027 |

## V. Experimental Results

We implemented the proposed ensemble-based prediction algorithm in R using time series and machine learning model implementations in the forecast package. For our ensemble solution we selected ARIMA, Neural Network, Exponential Model and Naive Prediction as the base models. After emperical evaluation of various error measurements for contribution coefficient calculation, we used square root of the exponentially fitted squared forecasting error as the error measure for calculating the contribution coefficients from each model (where $e_{i,t}$ is defined as $e_{(i,t)} = (x_t - \hat{x}_t^{(i)})^2$ in Equation 10 and $c_{(i,t)} = \frac{1}{\sqrt{b_{(i,t)}}}$ in Equation 11).

We tested proposed technique against the publicly available datasets mentioned in Section 4.2, as well as several real-world cloud workloads [22] collected from server applications. Our proposed solution is compared with each individual models in ensemble solution, two other popular ensemble techniques (mean ensemble, median ensemble), existing prediction technique in Apache Stratos and prediction model describes in [3].

In addition to the cloud workload dataset used in Section 3, we also used a portion of the Google cluster dataset [23] in evaluating our ensemble model. The dataset includes a series of task execution details against their starting and ending times. Due to the large size of the dataset, we have summarized it to a suitably concise set of values by summing up numbers of started tasks in each time unit over the transformed time scale.

Boldfaced cells in Tables 3 and 4 contain the smallest error values for each dataset under evaluation, whereas underlined cells contain the worst performance observed for each dataset among all methods. As shown visually in Fig. 6, although the ensemble model shows smoothened predictions at the start of the Google cluster dataset, it eventually manages to capture finer details of the series as the available dataset grows during real-time training.

According to the results obtained, the existing model from Stratos suffers from errors of the largest magnitude in several datasets, while each individual model shows largest errors in at least one dataset. While the mean and median ensemble methods generally perform better than individual models, they suffer when one or more of their base models perform significant errors, due to the lack of consideration of such forecasting errors resulting from individual models. Our proposed prediction technique provides the best prediction results for several datasets while never performing worse than the individual prediction methods, because the contribution from each of its base models is explicitly governed by its accuracy.

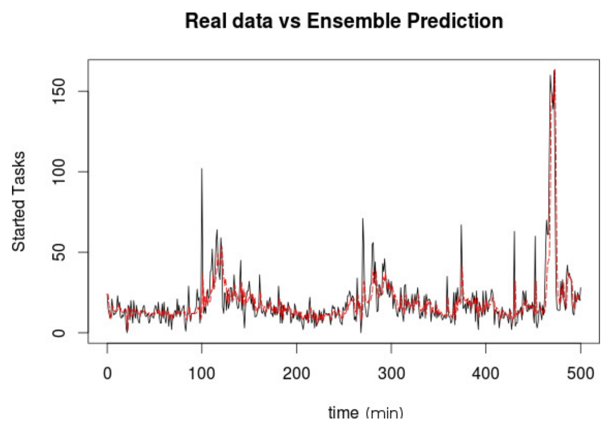

Figure 6. Ensemble prediction applied to Google cluster dataset.

## VI. Summary

Providing better QoS while maintaining lower resource lease costs in a PaaS system is extremely challenging for a reactive, threshold-based auto scaling solution. Even though proactive solutions can overcome these challenges, they are dependent on highly accurate prediction mechanisms. Existing workload prediction techniques which mainly focus on single, offline-trained models do not work well with heterogeneous PaaS applications with drastically different and varying workload patterns. To address this problem, we proposed an ensemble prediction technique that combines predicted values from different time-series prediction techniques. We train each model in real time (as in a functional auto scaler) and combine the forecasts based on weights calculated using inverse errors of fitted values for the training data. Rather than taking only the last forecasting error or the mean over all the past errors, we used exponential smoothing to fit the past forecasting error of each model. We implemented the proposed ensemble technique using four popular forecasting models and tested it on three publicly available datasets, two cloud workload dataset, and the Google cluster dataset. Experimental results show that the proposed ensemble prediction technique produces the least errors for many of the drastically different datasets considered, and it always performs better on any dataset where an individual model performs worse. As the proposed forecasting method should generate the forecast within a bounded time, we limit the size of the input training window. Further work is needed to identify the optimum input window size that would maximize accuracy while not violating the temporal restrictions on calculating the forecasts in real-time.

## References

[1] P. M. Mell and T. Grance, "The NIST definition of cloud computing," Tech. Rep., 2011. [Online]. Available: http://dx.doi.org/10.6028/nist.sp.800-145

[2] 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," *Communications of the ACM*, vol. 53, no. 4, p. 50, Apr. 2010. [Online]. Available: http://dx.doi.org/10.1145/1721654.1721672

[3] N. Roy, A. Dubey, and A. Gokhale, "Efficient Autoscaling in the Cloud Using Predictive Models for Workload Forecasting," in *2011 IEEE 4th International Conference on Cloud Computing*. Institute of Electrical & Electronics Engineers (IEEE), jul 2011. [Online]. Available: http://dx.doi.org/10.1109/cloud.2011.42

[4] T. Lorido-Botran, J. Miguel-Alonso, and J. A. Lozano, "A review of auto-scaling techniques for elastic applications in cloud environments," *J Grid Computing*, vol. 12, no. 4, pp. 559–592, Oct. 2014. [Online]. Available: http://dx.doi.org/10.1007/s10723-014-9314-7

[5] H. Alipour, Y. Liu, and A. Hamou-Lhadj, "Analyzing auto-scaling issues in cloud environments," in *24th Annual Intl. Conf. on Computer Science and Software Engineering*, ser. CASCON '14. Riverton, NJ, USA: IBM Corp., 2014, pp. 75–89. [Online]. Available: http://dl.acm.org/citation.cfm?id=2735522.2735532

[6] J. Kupferman, J. Silverman, P. Jara, and J. Browne, "Scaling into the cloud," Available at http://cs.ucsb.edu/~jkupferman/docs/ScalingIntoTheClouds.pdf (2015/07/20), 2009.

[7] H. Mi, H. Wang, G. Yin, Y. Zhou, D. Shi, and L. Yuan, "Online self-reconfiguration with performance guarantee for energy-efficient large-scale cloud computing data centers," in *IEEE Intl. Conf. on Services Computing*, July 2010. [Online]. Available: http://dx.doi.org/10.1109/scc.2010.69

[8] M. Arlitt and T. Jin, "1998 World Cup web site access logs," 1998. [Online]. Available: http://www.acm.org/sigcomm/ITA/

[9] C. Bunch, V. Arora, N. Chohan, C. Krintz, S. Hegde, and A. Srivastava, "A pluggable autoscaling service for open cloud PaaS systems," in *5th IEEE Intl. Conf. on Utility and Cloud Computing*, Nov. 2012. [Online]. Available: http://dx.doi.org/10.1109/ucc.2012.12

[10] J. Yang, C. Liu, Y. Shang, Z. Mao, and J. Chen, "Workload predicting-based automatic scaling in service clouds," in *6th IEEE Intl. Conf. on Cloud Computing*, June 2013. [Online]. Available: http://dx.doi.org/10.1109/cloud.2013.146

[11] A. Khan, X. Yan, S. Tao, and N. Anerousis, "Workload characterization and prediction in the cloud: A multiple time series approach," in *IEEE Network Operations and Management Symposium*, Apr. 2012. [Online]. Available: http://dx.doi.org/10.1109/noms.2012.6212065

[12] S. Islam, J. Keung, K. Lee, and A. Liu, "Empirical prediction models for adaptive resource provisioning in the cloud," *Future Generation Computer Systems*, vol. 28, no. 1, pp. 155–162, Jan. 2012. [Online]. Available: http://dx.doi.org/10.1016/j.future.2011.05.027

[13] R. J. Hyndman, *forecast: Forecasting functions for time series and linear models*, 2015, r package version 6.2. [Online]. Available: http://github.com/robjhyndman/forecast

[14] R. J. Hyndman and G. Athanasopoulos, *Forecasting: Principles and practice*. OTexts, 2013. [Online]. Available: https://www.otexts.org/fpp/

[15] "Welcome to STAT 510 – Applied time series analysis," Available at https://onlinecourses.science.psu.edu/stat510/ (2015/07/20).

[16] "How to identify patterns in time series data: Time series analysis," Available at https://www.statsoft.com/Textbook/Time-Series-Analysis (2015/07/20).

[17] "Why Apache Stratos is the preferred choice in the PaaS space," Available at http://stratos.apache.org/about/why-apache-stratos.html (2015/07/20).

[18] N. Wagner, Z. Michalewicz, S. Schellenberg, C. Chiriac, and A. Mohais, "Intelligent techniques for forecasting multiple time series in real-world systems," *Intl. Journal of Intelligent Computing and Cybernetics*, vol. 4, no. 3, pp. 284–310, Aug. 2011. [Online]. Available: http://dx.doi.org/10.1108/17563781111159996

[19] G. Zhang, "Time series forecasting using a hybrid ARIMA and neural network model," *Neurocomputing*, vol. 50, pp. 159–175, Jan. 2003. [Online]. Available: http://dx.doi.org/10.1016/s0925-2312(01)00702-0

[20] H. Zou and Y. Yang, "Combining time series models for forecasting," *Intl. Journal of Forecasting*, vol. 20, no. 1, pp. 69–84, Jan. 2004. [Online]. Available: http://dx.doi.org/10.1016/s0169-2070(03)00004-9

[21] R. Adhikari and R. K. Agrawal, "Combining multiple time series models through a robust weighted mechanism," in *1st Intl. Conf. on Recent Advances in Information Technology (RAIT)*, Mar. 2012. [Online]. Available: http://dx.doi.org/10.1109/rait.2012.6194621

[22] "AutoscaleAnalyser," https://github.com/hsbhathiya/AutoscaleAnalyser/tree/master/datasets/cloud_traces, 2015.

[23] J. L. Hellerstein, "Google cluster data," Jan. 2010. [Online]. Available: http://googleresearch.blogspot.com/2010/01/google-cluster-data.html