Forecasting Methods for Cloud Hosted Resources, a comparison

M van Greunen
MIH Media Lab
Stellenbosch University
Stellenbosch, South Africa
Email: manrich@ml.sun.ac.za

H.A. Engelbrecht
MIH Media Lab
Stellenbosch University
Stellenbosch, South Africa
Email: hebrecht@ml.sun.ac.za

***Abstract*—The emergence of cloud management systems, and the adoption of elastic cloud services enable dynamic adjustment of cloud hosted resources and provisioning. In order to effectively provision for dynamic workloads presented on cloud platforms, an accurate forecast of the load on the cloud resources is required. In this paper, we investigate various forecasting methods presented in recent research, identify and adapt evaluation metrics used in literature and compare forecasting methods on prediction performance. We investigate the performance gain of ensemble models when combining three of the best performing models into one model. We find that our 30th order Auto-regression model and Feed-Forward Neural Network method perform the best when evaluated on Google's Cluster dataset and using the provision specific metrics identified. We also show an improvement in forecasting accuracy when evaluating two ensemble models.**

I. Introduction

In recent years, the advent of Elastic Cloud solutions such as Google Compute Engine [1] and Amazon Web Services [2] provide dynamic resource availability and auto-scaling that enable reactive provisioning. Cloud Users (hosting applications or services in the cloud) are required to set auto-scaling policies such as target utilisation levels and overload thresholds according to their cloud application to minimise running costs and meet Service Level Agreements (SLAs) with their end-users. To achieve more effective provisioning (referred to as *proactive provisioning*) to handle fluctuations in workload presented to clouds, an accurate estimation of the resource requirement (or quantitative load) is required to ensure sufficient resources are available when needed [3].

The fields of statistics and machine learning have developed forecasting methods suitable for predicting on time series data, which have been applied to data from economics, finance and science. Much effort has been spent in improving the modelling and forecasting accuracy of methods, which include Auto-Regression and Exponential Smoothing together with work proposed in PRESS [4], Agile [5], iOverbook [6] (and many others). The problems that remain when using forecasting methods for provisioning cloud hosted resources are: (1) there does not exist a set of metrics to evaluate the performance accuracy of forecasting methods when applied to resource provisioning in clouds and (2) when comparing multiple types of forecasting methods, which prominent forecasting provides the most accurate prediction of the load presented on cloud hosted resources.

We make the following contributions:

1) Present a set of evaluation metrics collected and adapted from literature, used to evaluate and compare forecasting methods.
2) Implement and compare prominent forecasting methods presented in recent research from both statistical and machine learning fields.

The rest of the paper is summarised as follows: Section II covers research on proactive provisioning methods. Section III, covers preliminary theory on forecasting and prediction methods. Section IV describes the evaluation metrics and the dataset used in the Evaluation section of the paper and our experimental methodology. Section V discusses the evaluation results and we conclude in Section VI.

II. Literature Synopsis

In this section, we identify prominent approaches to proactive provisioning in the cloud environment, that was recently published. Specifically, we use the 2012 technical report by Lorido-Botrán et al. [7] as primary source to identify the set of advance prediction methods described here. The report lists five method categories, applicable to reactive scaling and proactive provisioning, but we focus here on prediction methods from machine learning and time-series analysis.

Gong, Gu and Wilkes [4] present *PRESS*, a time-series analysis forecasting method that uses Fast-Fourier-Transforms and dynamic time warping to extract signature-patterns from resource demand data. In cases where the data does not contain a significant repeating pattern, they use a machine learning approach of a discrete first-order Markov chain (with finite states). They focus on accurately predicting short-term load changes, evaluate PRESS on Google's 7-hour workload cluster dataset [8] and show that PRESS outperforms methods such as mean-max, auto-correlation and auto-regression.

Nguyen et al. [5] extend PRESS and propose a time-series analysis method, *Agile*, which uses Wavelet-transforms to perform medium-term resource demand predictions. Evaluating on the 29 day Google Cluster usage trace dataset [9], Agile consistently outperforms PRESS and Auto-regression when evaluated on CPU and Memory overload prediction rates.

Huang et al. [10] propose a time-series analysis prediction model that uses an exponential smoothing method, specifically a double exponential smoothing (Holt method), to improve accuracy of resource estimation in proactive provisioning.

978-3-901882-77-7

They are able to show improvements in accuracy on simulated workloads. Chandra et al. [11] employ an auto-regression, predictive time-series analysis method to dynamically provision resources in a shared datacenter. Caglar and Gokhale [6] present *iOverbook* a intelligent resource management tool that uses a machine learning method, namely an Artificial Neural Network to predict overbooking rates in datacenters. They identify ‘features’ associated with the resource data in Google’s cluster dataset and uses a Feed-Forward Neural Network (FFNN) to forecast the mean hourly CPU and Memory usage one-step-ahead. Prodan and Nae [12] propose a machine learning method that uses a Recurrent Neural Network (RNN) to predict load on a cloud hosted Massively Multiplayer Online Game (MMOG). They employ an Elman type RNN to estimate the load and dynamically provision and scale the resources used by the MMOG. They show that their estimator can accurately (using an average prediction error) predict various loads including flash-crowd behaviour.

From the synopsis above, we conclude that:

1) the squared error (or variants of it) is a popular accuracy metric used to evaluate the performance of a proposed method,
2) there is no agreement on training- or prediction window lengths when modelling or forecasting resource demand and
3) the proposed solutions are primarily compared with naive methods and not against other prominent approaches (with the exception of Agile being compared against PRESS).

III. Preliminaries

In Section II, we identified a set of prominent forecasting methods used in provisioning of cloud hosted resources. In this section, we cover the preliminary theory and formulation of forecasting equations for each method investigated. These include Exponential Smoothing, Auto-regression (AR), Markov Chains and two types of Neural Networks.

We define a **Time series** as a set of sequential data points or measurements collected at a regular time interval. A **model** is defined as a mathematical description of a process which generates a given time series [13, p.15]. **Forecasting** is defined as a procedure where historical data is input into a model and predictions are produced as the output of that model. According to Mao and Humphrey [14], the workloads presented to cloud hosted resources can be characterised into four categories, namely Stable, Trending, Seasonal/Cyclic and Bursty (or Stochastic). A Stable workload is characterised by constant load, whereby a Trending workload is observed when the load is constantly increasing, for example when a particular website becomes more popular. Seasonal/Cyclic workloads are characterised by having repeating highs and lows, for example load on an online retailer, in which higher workloads are observed by day compared to by night. A Bursty workload is characterised by a sudden increase in load, such as the increase in the number of views to a news site, reporting on a breaking story. For each method discussed here, we will highlight which of these characteristics the method is able to model. An appropriate resource model should be able to accurately model all four categories of workloads.

Exponential smoothing is a time-series smoothing technique, similar to the moving average, which assigns exponentially decreasing weights to past observations putting more emphasis on the most recent observations and thus is able to follow a time-series that contains a trend [15]. Improvements to Exponential smoothing was made by the authors in [16], presenting the Holt-Winters method which is capable of predicting on a time series that contain a trend and/or seasonal component. The forecasting expression for the Holt-Winters method are as follow, with weights being estimated using Least Squares Estimation (LSE):

$$y_{t+k} = (s_t + kb_t)I_{t-L+k}, \quad k = 1, 2, 3, ... \tag{1}$$

where s_t is the level-estimate, b_t the trend-estimate, and I_t the estimate of the seasonal component at time t. The number of future time steps to predict is given by k. Formal definitions for s_t, b_t and I_t can be found in [17, sec. 7.2].

Auto-regression (AR) takes a different approach than Exponential smoothing by using auto-correlation coefficients as weights when modelling time series. AR is able to model cyclic patterns and regress short-term trends. The forecasting expression for a AR model of order p is given by:

$$\begin{aligned} y_{t+1} &= \delta + \phi_1 y_t + \phi_2 y_{t-1} + ... + \phi_p y_{t+1-p} \\ \delta &= (1 - \sum_{i=1}^{p} \phi_i)\bar{y} \end{aligned} \tag{2}$$

where $\bar{y}$ is the mean of the time series up to time t and ϕ_i is the AR coefficients, calculated using Yule-Walker equations that are discussed in [18].

A **Markov Chain** is a machine learning method used to fit and predict stochastic sequential data. Mark Craven [19] defines a Markov chain model as one that has a set of *states* and associated *transitions probabilities*. These probabilities describe transitions from one given state to any possible next state. A discrete first-order Markov chain models a time series y_t, by defining m states (denoted as $\mathbf{S} = \{x_1, x_2, x_3, ..., x_m\}$) with each state corresponding to a discrete value. States are obtain by digitising the data into equally spaced bins. Each state x_i depends only on the previous state x_{i-1} and using this property a transition probability matrix $\mathbf{P}$ of size $m \times m$ is constructed. According to PRESS [4], a Markov chain is capable of modelling short-term trends and capture the bursty nature of cloud usage data. Given the state probability distribution π_t at time t, forecasting can be done by:

$$\hat{y}_{t+k} = \max(\pi_t \mathbf{P}^k) \quad k = 1, 2, 3, ... \tag{3}$$

where $\hat{y}_{t+k}$ is the predicted state, k steps into the future and $\mathbf{P}$ is the 1st-order transition matrix constructed using the method described in [19].

Neural Networks are machine learning techniques, inspired by the biology and cognitive functionality of living brains, capable of ‘learning’ complex functions that map inputs to corresponding outputs. In this paper we investigate two types of neural networks namely, a *Feed-Forward Neural Network* (FFNN) and an Elman *Recurrent Neural Networks* (RNN). The authors of *iOverbook* [6] proposed FFNNs as they are able to learn complex trends and seasonal characteristics of workloads present in a shared datacenter. Prodan and Nae [12] confirm this and use an Elman type RNN which is more

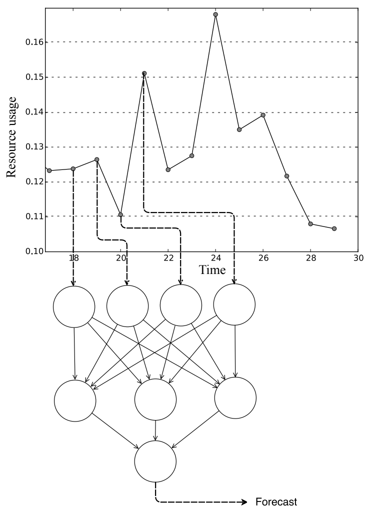


Fig. 1: An example of how a Feed-Forward Neural Network is used to predict a value into the future using past values as input.

resilient to the bursty nature of flash-clouds in MMOGs. Elman networks [20], are RNNs with 'context' neurons that provides memory. These context neurons allow the network to remember *state* which is beneficial when processing state-dependant tasks. Backpropagation, as presented in [21] is used to train both types of NNs described here. Figure 1 illustrates how a FFNN is used to forecast on time series data.

IV. Experimental Design

In this section, we list a set of evaluation metrics, discuss the dataset and parameters used in evaluation and describe our experimental methodology.

A. Evaluation metrics

When identifying evaluation metrics form literature, we want to be able to measure regression type accuracy as well as accuracy measures for resource provisioning. We use Root Mean Squared Error (RMSE) as a generic prediction accuracy metric for time series, and use provision specific metrics proposed by the authors of PRESS and Agile.

RMSE: gives an indication of how accurately the predictions follow the true values. Scikit-learn [22] defines RMSE as the error measure corresponding to the expected value of the quadratic error loss and is calculated as the square 'distance' between the predicted values and the true values.

Over- and under-estimation: RMSE calculates the error distance between the forecasted values $\hat{\mathbf{y}}$ and the true values $\mathbf{y}$, but fails to indicate if this error distance was because of values being over- or under-estimated. We will follow PRESS's [4] over- and under-estimation metric, classifying an over-estimation as a value more than 10% of the true value and under-estimation, a value less than 10%. The over-estimation rate (OER) is calculated by the ratio of over-predicted values to total forecasts, with the under-estimation rate (UER) calculated in a similar way.

An accurate model has a low over-estimation rate and a low under-estimation rate and using this we define an **Estimation score** (ES) used in this work, as follows:

$$\text{ES} = \frac{1}{2}(\text{OER}) + \frac{1}{2}(\text{UER}) \tag{4}$$

where OER and UER denotes the over-estimation and under-estimation rates. The factor of $\frac{1}{2}$ can be adjusted to weight each rate differently.

Correct estimation: A value within $\pm 10\%$ of the true value is classified as *correct*, with correct estimation rates calculated similar to PRESS.

Overload estimation: Similar to Agile [5], we define *overload* for a specific machine as any value above the 70th percentile of all values in the that machine's data trace. Liu and Cho [23] reported in their paper, that 93% of the machines monitored in the Google cluster dataset have a capacity set to 0.5, supporting Agile's argument for setting *overload* at a capacity of 0.7. Using the standard classification measures, namely *true positive* (T_p), *false positive* (F_p), *true negative* (T_n) and *false negative* (F_n); a predicted value is deemed true positive when it is forecasted as overloaded and agrees with its true value. Similar for true negative, a forecasted value and its corresponding true value are both less than the overload threshold. For false positive and false negative, the forecasted value is different to its true value. We calculate the True Positive Rate (TPR) and False Positive Rate (FPR) using the following:

$$\text{TPR} = \frac{T_p}{T_p + F_n}, \qquad \text{FPR} = \frac{F_p}{F_p + T_n} \tag{5}$$

where a higher TPR and lower FPR indicates higher accuracy.

Overloaded state estimation: Agile defines an *overload state* as Q consecutive overloaded values, seen Figure 2. Similar to Agile, we set $Q = 5$ samples or 25 minutes i.t.o the Google dataset. A *true positive* overloaded state prediction is one where the start of the predicted overloaded state is within a 'grace period' of 3 samples of the true overloaded state's start. The prediction is deem *true negative* if the end of a overloaded state is predicted within the 'grace period' of the true end of the overloaded state.

Trending and Seasonal/Cyclic workloads present longer overloaded states and a method with a high TPR and low FPR is more accurate.

B. Dataset

For the main analysis and evaluation we use the 2011 Google's Cluster usage trace dataset [9] which contains the resource requirements corresponding to tasks scheduled onto each machine in Google's production cluster cell, recorded over 29 days in May of 2011. The recorded resource measures include among other, CPU usage and Memory usage data captured at 5 minutes intervals, producing 8352 data points per machine, over the 29 days. According to Liu and Cho [23], the dataset has been sanitised, but still gives accurate information on cluster usage and load, important for the work presented in this paper.

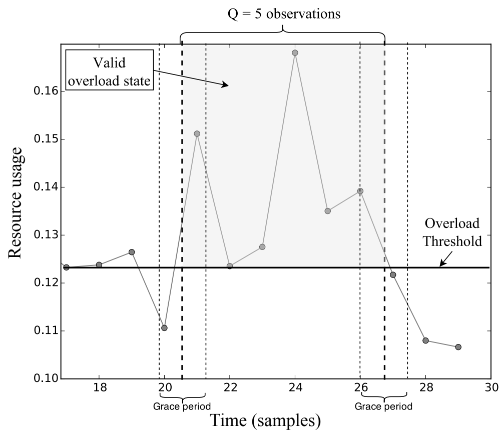


Fig. 2: An illustration of an overload state. A valid overload state is one where Q consecutive observations are above the overload threshold. When predicting an overload state, a positive prediction is classified as one where the start and end of the overload state is predicted within the grace period.

Parameter	**Value**
Training window	3000 samples
Forecasting window	30 samples
Overload threshold	70th percentile
Overload duration	5 samples
Grace period	3 samples
Minimum forecast value	5th percentile

TABLE I: Parameters used in experiments.

C. Experimental Methodology

To perform a fair comparison between methods we use the same training data, modelling and evaluation parameters across all experiments. Table I summarises these parameters, and the design decisions made are discussed here. We conducted our experiments following Agile's random sub-sampling validation approach, ensuring the models are data independent and results are statistically significant. All modelling and evaluations where performed using the **CPU** and **Memory** usage resource data.

Similar to Agile, we choose to use a Training window length of 3000 samples (250 hours), which is about a third of the 29 day data trace. The Forecasting window is set to 30 samples (2.5 hours) which is 1% of the training window. The Overload threshold was set to the 70th percentile of the total trace, the Overload duration and Grace period set to 5 and 3 samples respectively. The forecasting methods used may predict negative values and thus we followed PRESS's suggestion to replace negative values with the 5th percentile value of the total trace.

Next, we discuss method-specific design decisions. In our Holt-Winters implementation we speed-up refitting by limiting the model parameter to values in $[0, 1]$ and use the previous time step's model parameters as starting conditions for the Least Squared optimisation. For our auto-regression model, we set the order p, denoted by AR(p), equal to the Forecasting window of 30 samples. The order of an AR(p) model determines the number of predictions the model can accurately predict into the future. In our preliminary experiments, we found that higher values for p increases computation when fitting the model. For our Markov models, we followed PRESS's method which uses 40 states and digitises the data into equal-sized bins. We implemented a first-order and a second-order Markov model and did not consider higher order than second order, because the dimension of the transition matrix $\mathbf{P}$ increases exponentially with the order p.

We implemented both PRESS and Agile according to the descriptions in their respective papers. For our PRESS implementation we use a mean-ratio of 0.1 (instead of 0.05), because the 29 day Google dataset is more noisy compared to the 7-hour dataset used in PRESS. In our Agile wavelet model, we select a wavelet function and scale from PyWavelet's [24] wavelet library, that gives the best absolute distance between the fitted values and the true values.

A Neural Network's accuracy depends greatly on the choice of *hyper-parameters*, with grid search being a common approach to find suitable values for these parameters. We follow Bergstra and Bengio [25], which suggests rather using random search to find *good* parameters. Random search take less time and do not require a grid to be set up beforehand. The first 1000 samples of each machine is used to search for hyper-parameters using the best RMSE on a validation set (10% of the 1000 samples), we find a network for each machine.

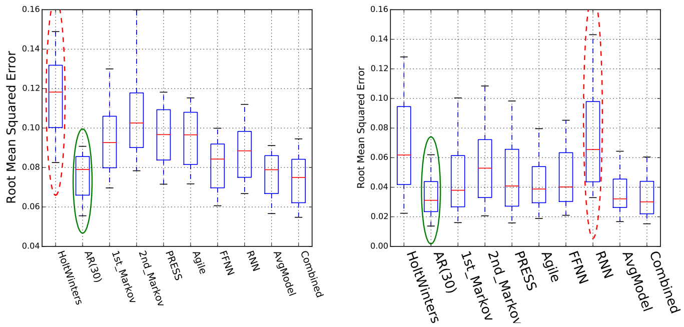
Ensemble models: After initial implementation and evaluation of individual methods is done, we identify three top performing models and combine these into ensemble models. We investigate two combination methods: an average model (denoted as *AvgModel*) and a Feed-Forward Neural Network (FFNN) combination model (denoted as *Combined*). The *AvgModel* weights forecasts produced by each model equally and combines to produce new forecasts per forecasting window. For our *Combined* ensemble model, we use a 3-layer FFNN (3 input neurons, 2 hidden neurons and 1 output neuron) and train it on the first 30 forecasts. Using this network we predict the next forecasting window and update the network on the previous window's true values.

V. Evaluation and Results

In this section. we discuss the evaluation results of all the methods investigated, evaluated using the generic and provisioning specific metrics discussed above. All results are whisker-box-plots with the top and the bottom of the box represents the 25th and 75th percentile of data across randomly selected 100 machines and the whisker ends represents the 90th and 10th percentile values respectively. The best and worst performing methods (excluding the two ensemble models) are highlighted in each plot, using a green solid-line and red dashed ellipses respectively. The three methods that performed the best and used in the ensemble models, are: AR(30), Agile and FFNN.

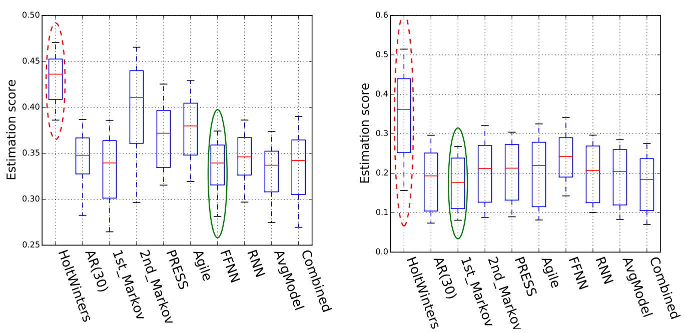
Figure 3 shows the **RMSE** results of every model. Holt-Winters method performs worst and our AR(30) model performs the best on both CPU and Memory usage. The Holt-Winters method and our RNN rely on seasonality in data, thus the large variance in their RMSE performance on Memory usage data (see Figure 3b) may suggest that the Memory data lacks seasonality. Both ensemble models outperform the other methods with the exception of AR(30), indicating that combining models does increase RMSE accuracy.

Figure 4 shows the **Estimation score** calculated in our work, as the average of the over- and under-estimation rates per



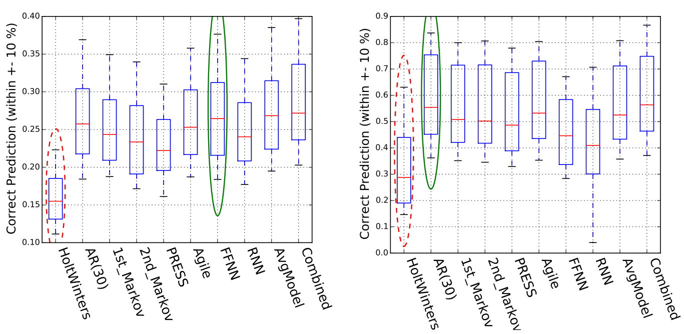
(a) RMSE on CPU usage. (b) RMSE on Memory usage.

Fig. 3: RMSE results for (a) CPU resource data and (b) Memory resource data. (Lower is more accurate).



(a) CPU usage data. (b) Memory usage data.

Fig. 4: Estimation score of all methods, calculated by combining the over- and under-estimation rates. (Lower estimation scores indicates more accurate predictions).

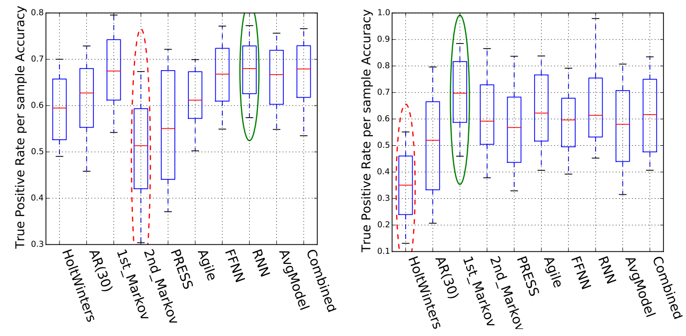


(a) CPU usage usage. (b) Memory usage data.

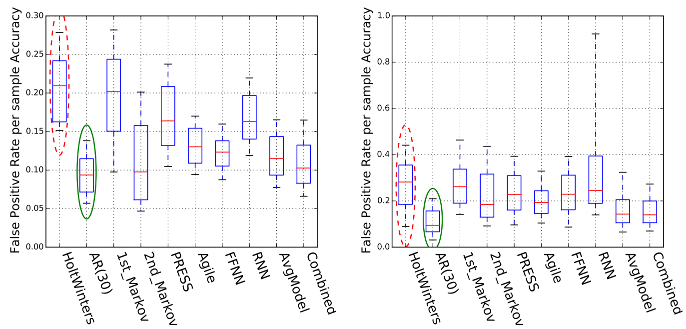
Fig. 5: Correct estimation rates for (a) CPU and (b) Memory usage.

method, with lower Estimation scores indicating more accurate predictions. Our FFNN performs the best on the CPU usage data (see Figure 4a) and the AR(30) method on Memory usage data (see Figure 4b). Holt-Winters' method under-performs on both CPU and Memory usage data. Again the ensemble models show an increase in accuracy with lower Estimation scores than the individual methods.

Figure 5 shows the **Correct estimation** rates for all methods evaluated. Higher correct predicted rates indicates better accuracy and using this we see that our FFNN model outperforms on CPU usage data and our AR(30) on Memory



(a) CPU usage data. (b) Memory usage data.



(c) CPU usage data. (d) Memory usage data.

Fig. 6: True Positive and False Positive Rates for correct overload predictions for CPU resource data in (a) and (c) and Memory resource data in (b) and (d).

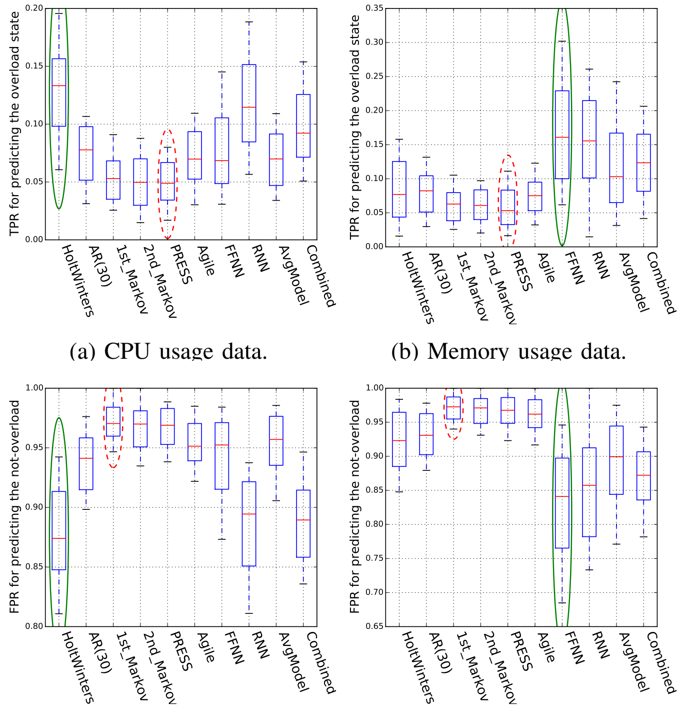
usage data. This confirms our Estimation score results and we see the increase in accuracy when combining models. Holt-Winters under-performs and again we see the large variance of the RNN model on Memory data (see Figure 5b), confirming the RMSE results.

The Estimation score and Correct prediction results indicate the FFNN and AR(30) are suitable methods for provisioning clouds that exhibit Stable and Treading workload, because these two methods will be able to accurately forecast on these types of workloads.

Overload estimation: High TPRs and low FPRs are the criteria for accurate forecasts as these forecasts are used in provisioning problems where it is important to have resources ready ahead of time. Figures 6a and 6c show that for CPU usage data, our RNN model has a high TPR and our AR(30) a low FPR. The first-order Markov chain performs the best on Memory usage with AR(30) having a low FPR. The large variance on RNN's performance, specifically on the Memory data (see Figure 6d) may suggest that the RNN model is biased to predicting overload on the top 75% of cases.

A Bursty workload will lead to lower TPR, as overloaded samples will be more difficult to predict, which may suggest that our RNN is able to handle Bursty workload better than other methods.

Overloaded state estimation: Figures 7a and 7c show a surprising result, the Holt-Winters method outperforms the other methods, having a high TPR and low FPR, with our neural networks implementations second best. Our RNN performs better on CPU data, which again suggests that the CPU data has a seasonal component and the Memory data not. Our



(a) CPU usage data.

(b) Memory usage data.

(c) CPU usage data.

(d) Memory usage data.

Fig. 7: Overload state results: True Positive and False Positive Rates for CPU resource data in (a) and (c) and Memory resource data in (b) and (d).

FFNN outperforms the RNN on forecasting accuracy on the Memory data, which is also surprising because a FFNN does not have a way to remember state, which we thought would be required for accurate overload state predictions. Future investigation on the root cause of the results, need to be performed.

Trending and Seasonal/Cyclic workloads present more overloaded states for longer periods compared Stable and Bursty workloads which indicates that both neural networks are able to accurately forecast on these types of workloads.

Investigating PRESS and Agile results: From the results above we find that PRESS and Agile are outperformed by other methods, but specifiably both these methods are outperformed by our AR(30) model. This does not agree with the results obtained in their respective papers. To investigate these findings, we identified the following differences in our implementations compared to the explanations given in PRESS and Agile's papers: For PRESS: (1) we performed our evaluations on Google's 29 day resources dataset compared to the 7 hour dataset used in [4]. (2) We performed medium-to-long term forecasts (using a forecasting window of 30 samples) compared to their short term forecasts. (3) We used a mean-ratio constraint of 0.1 instead of 0.05, for similar signature patterns. (4) In our first-order Markov chain, we used the frequency of state transitions to construct the transition matrix $\mathbf{P}$, as it was not clear which method PRESS used.

For Agile: (1) the specific wavelet functions employed in Agile's wavelet transforms are unknown and we used wavelet functions provided by the PyWavelet [24] library. (2) We set the order of our auto-regression model, equal to the forecasting window (30 samples), which may differ from the Auto-regression model Agile used to compare their wavelet-based approach.

To investigate the statistical significance of our AR(30) results compared to our PRESS and Agile results, we rerun all evaluations four more times using a new set of 100 random machines on each run. At a statistical significance level of $\alpha = 0.05$, we found that our AR(30) performed better than both PRESS and Agile, with most *p-values* practically zero. The only exceptions where for the Correct estimation rate (*p-value* $= 0.192$) and the True Positive Rate for overloaded state (*p-value* $= 0.156$), when comparing our AR(30) and our Agile implementation.

VI. Conclusion

In this paper, we stated that to improve provisioning of cloud hosted resources, accurate forecasts of the load ahead of time are required. The problem this work addresses is that there does not exist a set of metrics to evaluate the performance accuracy of forecasting methods and in order to compare prominent forecasting methods. We contributed by describing provision specific evaluation metrics collected and adapted from literature and used these to evaluate forecasting accuracy of six prominent provisioning methods from literature. Using the 2011 Google's Cluster usage trace dataset [9], we evaluated on CPU and Memory resource usage data and focussed on comparing the performance of the forecasting methods investigated.

Our experiments show that:
(1) Auto-regression performs better than other methods when compared across the majority of metrics on CPU and Memory usage data.
(2) Holt-Winters' method under-performed in all metrics except for predicting of overload state on CPU usage data.
(3) Our first-order Markov chain method under-performs on most of the metrics, which may be an indication why PRESS underperformed as well.
(4) Using the top three methods; AR(30), Agile and Feed-Forward Neural Network, the ensemble model increase the forecasting accuracy across the majority of metrics. Our simpler *AvgModel* performs better than the *Combined* model, with the *Combined* model only showing minor improvements in some cases.
(5) PRESS and Agile surprisingly under-performs and our results do not agree with the results obtained in their respective papers. We found that Auto-regression statistical significantly outperforms both methods across the majority of metrics.

In future, we want to investigate the poor performance of our version of PRESS and Agile, possibly comparing the specific implementations. Furthermore, evaluate the methods identified in a closed loop cloud environment and investigate the possible increase in accuracy when having knowledge of the application or workload. In conclusion, when choosing forecasting methods for proactive provisioning, we believe that it is more efficient to use less complex methods like Auto-regression and put greater focus on tuning the model to the specific cloud application and workload presented to the cloud resources.

Source code available at:
http://github.com/Manrich121/ForecastingCloud.git

References

[1] Google, "Compute Cloud Platform." [Online]. Available: https://cloud.google.com/

[2] Amazon, "Amazon Elastic Compute Cloud (EC2) - Scalable Cloud Hosting." [Online]. Available: http://aws.amazon.com/ec2/

[3] Q. Zhang, L. Cheng, and R. Boutaba, "Cloud computing: state-of-the-art and research challenges," *Journal of Internet Services and Applications*, vol. 1, no. 1, pp. 7–18, Apr. 2010. [Online]. Available: http://www.springerlink.com/index/10.1007/s13174-010-0007-6

[4] Z. Gong, X. Gu, and J. Wilkes, "PRESS: PRedictive Elastic reSource Scaling for cloud systems," in *Proceedings of the 2010 International Conference on Network and Service Management, CNSM 2010*. Ieee, Oct. 2010, pp. 9–16. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5691343

[5] H. Nguyen, Z. Shen, X. Gu, S. Subbiah, and J. Wilkes, "AGILE: elastic distributed resource scaling for Infrastructure-as-a-Service," *10th International Conference on Autonomic Computing (ICAC 13)*, p. 14, 2013. [Online]. Available: http://dance.csc.ncsu.edu/papers/icac2013.pdf http://cairo.csc.ncsu.edu/icac13/main_20130306120721_v2.pdf

[6] F. Caglar and A. Gokhale, "iOverbook: Intelligent Resource-Overbooking to Support Soft Real-time Applications in the Cloud," in *Cloud Computing (CLOUD), 2014 IEEE 7th International Conference on*, 2014. [Online]. Available: http://www.dre.vanderbilt.edu/ gokhale/WWW/papers/CLOUD-2014.pdf

[7] T. Lorido-Botrán, J. Miguel-Alonso, and J. A. Lozano, "Auto-scaling Techniques for Elastic Applications in Cloud Environments," *Technical Report: University of the Basque Country*, pp. 11 – 14, 2012.

[8] J. L. Hellerstein, "Google cluster data," Google research blog, Jan. 2010.

[9] J. Wilkes, "More Google cluster data," Google research blog, Nov. 2011. [Online]. Available: http://googleresearch.blogspot.com/2011/11/more-google-cluster-data.html

[10] J. Huang, C. Li, and J. Yu, "Resource prediction based on double exponential smoothing in cloud computing," *2012 2nd International Conference on Consumer Electronics, Communications and Networks (CECNet)*, pp. 2056–2060, 2012. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6201461

[11] A. Chandra, W. Gong, and P. Shenoy, "Dynamic resource allocation for shared data centers using online measurements," *Proceedings of the 2003 ACM SIGMETRICS international conference on Measurement and modeling of computer systems - SIGMETRICS '03*, p. 300, 2003. [Online]. Available: http://portal.acm.org/citation.cfm?doid=781027.781067

[12] R. Prodan and V. Nae, "Prediction-based real-time resource provisioning for massively multiplayer online games," *Future Generation Computer Systems*, vol. 25, no. 7, pp. 785–793, Jul. 2009. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/S0167739X08001933

[13] N. R. Farnum and L. W. Stanton, *Quantitative Forecasting Methods*, ser. Duxbury series in statistics and decision sciences. PWS-Kent Pub. Co., 1990, vol. 41. [Online]. Available: http://books.google.co.za/books?id=9AWYmbMLYmUC

[14] M. Mao and M. Humphrey, "Auto-scaling to minimize cost and meet application deadlines in cloud workflows," *2011 International Conference for High Performance Computing, Networking, Storage and Analysis (SC)*, pp. 1–12, 2011.

[15] R. G. Brown, *Smoothing, Forecasting and Prediction of Discrete Time Series*, ser. Dover Phoenix Editions. Dover Publications, 1963. [Online]. Available: https://books.google.co.za/books?id=XXFNW_QaJYgC

[16] P. R. Winters, "Forecasting Sales by Exponentially Weighted Moving Averages," *Management Science*, vol. 6, no. 3, pp. 324–342, 1960. [Online]. Available: http://dx.doi.org/10.1287/mnsc.6.3.324

[17] R. J. Hyndman and G. Athanasopoulos, *Forecasting: Principles and Practice*, 2013. [Online]. Available: http://otexts.com/fpp/

[18] P. Bourke, "AutoRegression (AR)," 1998. [Online]. Available: http://paulbourke.net/miscellaneous/ar/

[19] M. Craven, "Markov Chain Models (Part 1)," p. 8, 2011. [Online]. Available: https://www.biostat.wisc.edu/bmi576/lectures/markov-chains-1.pdf

[20] J. L. Elman, "Finidng structure in time," *Cognitive Science*, vol. 14, no. 2, pp. 179–211, 1990. [Online]. Available: http://onlinelibrary.wiley.com/doi/10.1207/s15516709cog1402_1/abstract http://doi.wiley.com/10.1207/s15516709cog1402_1

[21] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, pp. 533–536, 1986.

[22] Scikit-learn, "3.3. Model evaluation: quantifying the quality of predictions." [Online]. Available: http://scikit-learn.org/stable/modules/model_evaluation.html#regression-metrics

[23] Z. Liu and S. Cho, "Characterizing machines and workloads on a Google cluster," *Proceedings of the International Conference on Parallel Processing Workshops*, pp. 397–403, 2012.

[24] F. Wasilewski, "PyWavelets - Discrete Wavelet Transform in Python." [Online]. Available: http://www.pybytes.com/pywavelets/#

[25] J. Bergstra and Y. Bengio, "Random Search for Hyper-Parameter Optimization," *Journal of Machine Learning Research*, vol. 13, pp. 281–305, 2012.