Future Generation Computer Systems 28 (2012) 155–162

Contents lists available at SciVerse ScienceDirect

Future Generation Computer Systems

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

Empirical prediction models for adaptive resource provisioning in the cloud

Sadeka Islam[a,b,*], Jacky Keung[a,b,c], Kevin Lee[a,b], Anna Liu[a,b]

[a] National ICT Australia, Sydney, Australia
[b] School of Computer Science and Engineering, University of New South Wales, Sydney, Australia
[c] Department of Computing, The Hong Kong Polytechnic University, Hong Kong

ARTICLE INFO

Article history:
Received 9 March 2011
Received in revised form
11 May 2011
Accepted 28 May 2011
Available online 25 June 2011

Keywords:
Cloud computing
Resource provisioning
Resource prediction
Machine learning

ABSTRACT

Cloud computing allows dynamic resource scaling for enterprise online transaction systems, one of the key characteristics that differentiates the cloud from the traditional computing paradigm. However, initializing a new virtual instance in a cloud is not instantaneous; cloud hosting platforms introduce several minutes delay in the hardware resource allocation. In this paper, we develop prediction-based resource measurement and provisioning strategies using Neural Network and Linear Regression to satisfy upcoming resource demands.

Experimental results demonstrate that the proposed technique offers more adaptive resource management for applications hosted in the cloud environment, an important mechanism to achieve on-demand resource allocation in the cloud.

1. Introduction

Cloud computing is one of the most popular buzzwords in today's enterprise. An intrinsic feature of the cloud that differentiates it from traditional hosting services is its seemingly infinite amount of resource capacity (e.g. CPU, memory, Network I/O, disk etc.) offered at a competitive rate. It provides opportunities for start-up companies to host their applications in the cloud; thus, eliminating the overhead of procuring traditional infrastructure resources which typically takes several months.

However, a near-infinite resource pool for scale and flexibility is not the only potential of the cloud. Cloud hosting providers offer this near-infinite resource on demand using different pricing models, e.g. pay-per-use model for workloads with unforeseeable characteristics, reserved instance pricing model with long-term commitment of availability and spot instance model for workloads with flexible completion time. Therefore, application providers are able to choose an appropriate pricing model based on the anticipated workload characteristics and provision the resources accordingly in the cloud. The pay-as-you-go model and dynamic resource provisioning features of the cloud reduce the overhead associated with static provisioning, which is not considered as a cost effective option because of over-provisioning or under-provisioning of infrastructural resources at any particular moment. The time it takes to instantiate a new virtual machine (VM) instance on demand is relatively small, for instance, 5–15 minutes [1] as compared to a traditional month-long procurement process. A contemporary research challenge is to devise an intelligent way towards dynamic provision of resources in the cloud which is effective in terms of both cost and performance.

It is intuitive that if the dynamic resource scaling system is a reactive one, it might not be able to scale proportionally with the *Slashdot effect* [2] or sudden *Traffic surge* resulting from special offers or market campaigns; thus turning out to be catastrophic for application performance, leading to an unacceptable delay in response time and in the worst case, application unavailability. Therefore, proactive prediction-based resource scaling is required in order to cope up with the ever fluctuating resource usage pattern of e-commerce applications.

Predictive analysis of resource usage is the key to several crucial system design and deployment decisions such as, workload management, system sizing, capacity planning and dynamic rule generation in the cloud. Hence, our proposed prediction framework uses statistical models which are able to speculate the future surge in resource requirement; thus enabling proactive scaling to handle temporal bursty workload in a controllable way. For the prediction approach, we resort to machine learning algorithms (e.g. Neural Network and Linear Regression) and the sliding window technique, which have proved successful in the financial and health informatics area [3]. For training and testing the prediction model, we use data generated from running

[*] Corresponding author at: National ICT Australia, Sydney, Australia. Tel.: +61 293762194.
E-mail addresses: Sadeka.Islam@nicta.com.au (S. Islam), Jacky.Keung@comp.polyu.edu.hk (J. Keung), Kevin.Lee@nicta.com.au (K. Lee), Anna.Liu@nicta.com.au (A. Liu).

0167-739X/$ – see front matter © 2011 Elsevier B.V. All rights reserved.
doi:10.1016/j.future.2011.05.027

the TPC-W benchmark [4] in the Amazon EC2 cloud. Finally, the effectiveness of the prediction framework is validated by introducing evaluation metrics e.g. Mean Absolute Percentage Error (*MAPE*), *PRED*(25) etc. We show that our framework is not only able to make accurate projections but also skilled enough to forecast resource demand ahead of the VM instance setup time.

2. Related work

The unique business model of cloud computing provides subscription-based pay-per-use virtual services that allows IT to increase capacity or add capability in real time over the Internet, a way to extend IT's existing resources on demand without investing in new infrastructure and personnel associated costs.

The challenge here is that applications hosted in the cloud need to be elastic in order to achieve economy of scale while preserving the application-specific Service Level Agreements (SLAs) such as, response time, throughput etc. The usage prediction and dynamic provisioning of resources is one of the fundamental research challenges, because a balanced trade-off between the business-level SLAs and other constraints (e.g. VM setup overhead, cost effectiveness etc.) needs to be achieved. Recent research on dynamic provisioning in the cloud explores some interesting prediction techniques and heuristics as well as some measurement metrics to evaluate the approaches from different perspective; we strive to focus on the research problem of developing resource prediction models for facilitating proactive scaling in the cloud so that hosted applications are able to withstand the variation in workload with least drop in performance and availability. This section provides an overview of some of the related techniques.

2.1. Resource provisioning

The problem of resource provisioning in the cloud has been investigated from the platform hosting provider's perspective. Van et al. [5] presented solutions to automate the management of virtual machines for service hosting platforms while optimizing a global utility function that integrates the application-level SLAs and the operating cost of the platform provider. They resorted to the Constraint Programming approach to solve the optimization problem by defining business level SLAs of the application and the resource exploitation cost of the hosting provider as constraints. The proposed management system attempts to maximize the performance of the hosted applications with an optimal operating cost for the hosting provider. Throughout the study in [5], the important assumption on the performance model of the hosted application is omitted, whether it is a web application with stringent QoS requirements or a pure analytical one.

Quiroz et al. [6] introduced a Decentralized Online Clustering (DOC) mechanism for autonomic VM provisioning that introduces the unique challenges of enterprise grids and clouds. They also recognized the problem of inaccurate cloud client resource requests that lead to over-provisioning and therefore integrated a workload modeling technique, called Quadrature Response Surface Model (QRSM) with VM provisioning. This technique is used to model the application dynamics in the cloud environment so that a feedback on the appropriateness and the number of the requested resources with respect to the application-specific SLA and QoS requirements can be provided.

On the other hand, Silva et al. [7] proposed a heuristic for optimizing the number of machines for processing an analytical job with predefined number of independent tasks so that maximum speedup can be achieved within a limited budget. However, the traffic of a web application is dynamic and random in nature; hence predicting the optimal number of machines (instances) for the fulfillment of the application level SLAs in real time and within budget is not a trivial task.

Alternatively, Lim et al. [8] addressed the problem of building an effective external controller for automated adaptive scaling of applications deployed in the cloud. They recommended the Proportional Thresholding approach which dynamically adjusts the target range (i.e. high and low thresholds) based on the number of accumulated virtual machine instances. Thus the relative effect of allocating resources becomes finer as the number of accrued resources increases; eventually resulting in being adaptive and more resource efficient.

2.2. Resource prediction techniques

Although the works of Silva et al. [7] and Lim et al. [8] presented different new algorithms for adaptive scaling, their approach is not proactive and hence more susceptible to degradation of performance due to VM instance provisioning and booting delay in the cloud. In this regard, Caron et al. [9] initiated the groundwork for a new approach to workload prediction algorithms based on past usage patterns. Since dynamic allocation and de-allocation of virtual machine instances include some overheads such as, instance setup time, performance improvement and responsiveness could be achieved if the system can predict and scale in advance to adapt with the changing workload. Based on similar characteristics of web-traffic, they proposed a pattern matching algorithm that is used to identify closest resembling patterns similar to the present resource utilization pattern in a set of past usage traces of the cloud client. The resulting closest patterns are then interpolated by using a weighted interpolation to forecast approximate future values that are going to follow the present pattern. This prediction finally aids in making dynamic scaling decisions in real-time. However, their approach has several problems; firstly, searching for similar patterns each time over the entire set of historical data is inefficient. And secondly, it may lead to over-specialization, thus turning out to be ineffective.

Finally, Kupferman et al. [10] recommended a set of scoring metrics to measure the effectiveness and efficiency of dynamic scaling algorithms in terms of availability and cost. Their analysis revealed the fact that dynamic provisioning provides significant improvement over static allocation algorithms in terms of cost with least drop in availability. Again, prediction of resource provisioning based on past usage and intelligent de-allocation using ''smart kills'' further enhance the efficiency of the algorithm in the face of sharp and random spikes in traffic.

2.3. Resource prediction in the cloud

Our work aims at analyzing the problem of resource provisioning from the application provider's viewpoint so that the hosted application is capable of making autonomic scaling decisions by evaluating the future resource utilization (such as, CPU, memory, Network I/O etc.) in real time and thus request for additional virtual instances beforehand through intelligent prediction to maximize performance and availability. A majority of the existing works have been in the investigation of resource provisioning from the cloud service provider perspective.

The current scope of our work is bound to the development of a performance prediction model and its statistical validation only; in future work, we plan to implement and evaluate the model on top of the public cloud. Our proposed prediction technique is more prospective and robust in terms of resource prediction in the cloud for several reasons. First, we generated the historical data by running a standard client–server benchmark, such as TPC-W [4] on Amazon EC2 and used that data for training and testing of the forecasting models. Second, our prediction framework is developed using statistical learning algorithms and the sliding window mechanism which are simple yet effective. And finally, the accuracy of the prediction framework is assessed using our proposed evaluation metrics and cross-validation [11,12].

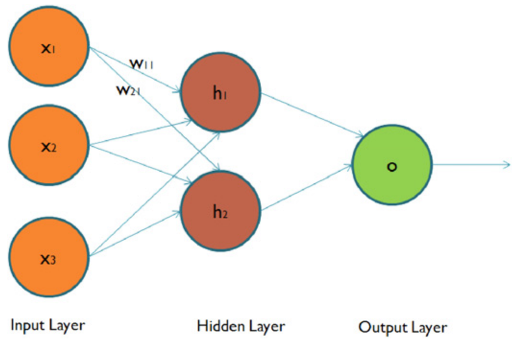

Fig. 1. A three layer Neural Network.

3. Methodology

We explored a number of existing machine learning techniques to make an acceptable prediction on future resource requirement in the cloud with respect to time. Hence our research problem actually relates to time-series analysis. The machine learning techniques, adopted here, are some of the most common and popular ones in time-series analysis. All of these methodologies are able to predict the most likely future outcome based on recent resource usage and historical data. The proposed learning algorithms are:

- Error Correction Neural Network (ECNN) [13].
- Linear Regression [14].

We selected these learning algorithms because these are naturally efficient and effective in the forecasting paradigm. However, it is worth noting that it is also possible to accommodate other learning methods, e.g. Support Vector Machines (SVM) for prediction of future resource usage.

In addition, we incorporate sliding window [15] and cross-validation [11,12] techniques in the training and prediction stages and consequently evaluate the validity of the learning algorithms for prediction accuracy using statistical metrics, as discussed in the following sections.

3.1. Learning algorithms

3.1.1. Error Correction Neural Network (ECNN)

In this approach, a Neural Network model [13] is trained, given the previous historical data. A typical three-layer Neural Network is shown in Fig. 1. The Neural Network consists of a number of layers: the input layer has a number of input neurons, $\boldsymbol{x} = [x_1, x_2, \dots, x_m]$; the output layer has one or more output neurons, $\boldsymbol{o} = [o_1, o_2, \dots, o_u]$ and several hidden layers with hidden neurons, $\boldsymbol{h} = [h_1, h_2, \dots, h_l]$ in between. In a fully-connected Neural Network, the neurons at each layer are connected to the neurons of the next layer; these connections are known as synapses. Each *synapse* is associated with a weight, which is to be determined during training.

During the training phase, the Neural Network is fed with input vectors and random weights are assigned to the synapses. After presentation of each input vector, the network generates a predicted output $\hat{y}$. The generated output is then compared with the actual output, y; the difference between the two is known as the *error term* which is then used as a feedback to correct the synaptic weights of the network. Since the error value guides the network towards convergence to the target output for a set of real-world inputs, therefore it is called a *feed forward* or *error correction* network. The rate at which the weight is updated is called the learning rate ρ, which is generally a value within the range [0, 1]. The training of the Neural Network continues until a specific criterion is met, e.g. the sum of squared errors falls below a certain threshold. The overall algorithm for training a three layer feed-forward Neural Network for forecasting can be expressed as the following [13]:

- Initialize the weight vectors $\boldsymbol{w_p}$,
  where $p = 1, 2, \dots, k$
  and $k = \sum(\textit{hidden neurons} + \textit{output neurons})$.
- Repeat
  - For $i = 1$ to N, where N is the number of input vectors in the training dataset.
  - Present input vector $\boldsymbol{x}(i)$ to the network.
  - For each neuron compute $\hat{y}_p(i)$, $e_p(i)$ and $\delta_p(i)$, given by the following formulae:
    $$\hat{y}_p(i) = f\left(\sum_{j=1}^{l} w_{pj}x_j(i)\right)$$
    $e_p(i) = \hat{y}_p(i) - y_p(i)$,
    where $\hat{y}_p(i)$ = predicted output and $y_p(i)$ = actual output for $\boldsymbol{x}(i)$
    $$\delta_p(i) = e_p(i)f'\left(\sum_{j=1}^{l} w_{pj}x_j(i)\right).$$
  - End {For}.
  - Update the weight vectors $\boldsymbol{w_p}$ using the formula: $\boldsymbol{w_p}(t+1) = \boldsymbol{w_p}(t) - \rho\sum_{i=1}^{N}\delta_p(i)\boldsymbol{x}(i)$, where ρ is the learning rate.
- Until a specific termination criterion has been met, i.e. the total prediction error is lower than a specific limit.

The above algorithm is also known as the back-propagation algorithm [13], which follows a gradient-descend technique [13] to reach the local optima. Thus the resultant Neural Network guarantees to produce an optimal prediction output after observing the input vector in real time.

3.1.2. Linear regression

An alternative approach to address the prediction problem is Linear Regression [14] which models the relationship between one or more input variables x and a dependent output variable y by using a linear equation to observed data. However, in the real world, the number of input variables is often more than one; in that case, it is called Multiple Linear Regression [14]. If the independent variable is $\boldsymbol{x} = [x_1, x_2, \dots, x_m]$, and the corresponding dependent variable is y, then the Linear Regression model takes this form:

$$y = \beta_0 + \sum_{i=1}^{m}\beta_i x_i.$$

The Linear Regression approach determines unknown parameter values, i.e., β_i during training phase that yields the best-fitted model with the given data points.

3.2. Sliding window technique

The sliding window technique [15] maps an input sequence $\boldsymbol{x}$ of size k to an individual output y by using a prediction model; this approach is very effective in the data mining field e.g. Protein Secondary Structure Prediction [16], Pronouncing of English Words [17], etc. Similarly, we map the challenge of resource usage prediction in the cloud as a time-series analysis problem where the input $\boldsymbol{x}$ is a vector of resource usage samples over k consecutive time intervals and the predicted output, y is r intervals ahead of this input window.

Fig. 2 demonstrates the idea of a sliding window where the input vector is of size k, i.e., $\boldsymbol{x} = [y(t), y(t+1), \dots, y(t+k-1)]$ over the time interval $[t, t+k-1]$ and the predicted output is $y(t+k+r-1)$ at time $(t+k+r-1)$ which is r time intervals ahead of the input window.

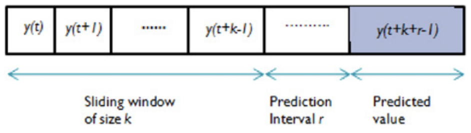

Fig. 2. Sliding window approach used in prediction of resource usage in the cloud.

3.3. *Evaluation criteria*

We evaluate the accuracy of the fitted models, generated from different learning algorithms based on a number of metrics: Mean Absolute Percentage Error (*MAPE*) [18], *PRED*(25) [18], Root Mean Squared Error (*RMSE*) [19] and R^2 Prediction Accuracy [20]. We describe the metrics in the following subsections.

3.3.1. *Mean Absolute Percentage Error (MAPE)*

The Mean Absolute Percentage Error [18] for a prediction model is given by the following formula:

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\frac{|\hat{y}_i - y_i|}{y_i} \tag{1}$$

where y_i is the actual output, $\hat{y}_i$ is the predicted output and n is the number of observations in the dataset for which the prediction is made. A lower value of *MAPE* implies a better fit of the prediction model; thus indicating superior prediction accuracy.

3.3.2. *PRED(25)*

The measure *PRED*(25) [18] is defined as the percentage of observations whose prediction accuracy falls within 25% of the actual value. A more formal definition of *PRED*(25) is as follows:

$$PRED(25) = \frac{No.\ of\ observations\ with\ relative\ error \leq 25\%}{No.\ of\ observations}. \tag{2}$$

From (2), it is intuitive that a *PRED*(25) value closer to 1.0 indicates a better fit of the prediction model.

3.3.3. *Root Mean Squared Error (RMSE)*

The metric Root Mean Square Error (*RMSE*) [19] is defined by the following formula:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(\hat{y}_i - y_i)^2}{n}} \tag{3}$$

where y_i is the actual output, $\hat{y}_i$ is the predicted output and n is the number of observations in the dataset. A smaller *RMSE* value indicates a more effective prediction scheme.

3.3.4. *R^2 Prediction Accuracy*

The R^2 Prediction Accuracy [20] is a measure of the goodness-of-fit of the prediction model. The formula of R^2 Prediction Accuracy is:

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(\hat{y}_i - y_i)^2}{\sum_{i=1}^{n}(\hat{y}_i - \bar{y})} \tag{4}$$

where $\bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i$ and y_i is the actual output, $\hat{y}_i$ is the predicted output and n is the number of observations in the dataset. Note that, the R^2 value falls within the range [0, 1]. This metric is commonly applied to Linear Regression models; in fact, R^2 Prediction Accuracy determines how the fitted model approximates the real data points. An R^2 prediction accuracy of 1.0 indicates that the forecasting model is a perfect fit.

3.4. *Cross validation*

We applied a cross-validation [11,12] technique for estimating the accuracy of the prediction models generated by the learning algorithms in Section 3.1. Cross-validation is one of the most common error estimation techniques where each observation $y_i \in y_1, y_2, \ldots, y_n$ in the sample dataset of size n is successively taken out and the remaining $n-1$ observations of the set are used to train the prediction model to estimate the projected resource usage. Later, the actual output y_i is used to validate the predicted output $\hat{y}_i$ inferred by the fitted model. This strategy is also known as Leave-One-Out Cross Validation (LOOCV) [11,12].

4. Experimental setup

We conducted our experiment of performance prediction in the cloud by implementing all of the approaches mentioned in Sections 3.1 and 3.2. Our experimental setup is divided into two steps:

- Historical Data Collection in the Cloud and
- Training of the Prediction System with the Historical Data.

4.1. *Data collection in the cloud using TPC-W benchmark*

TPC-W [4,21] is a specification for benchmarking e-commerce applications. It is commonly used for resource provisioning, scalability and capacity planning for traditional e-commerce websites. The TPC-W workload generator mimics a number of simultaneous user sessions for an e-commerce website, where the users can browse, order and perform transactions for products in an online retail store via emulated browsers. In our experiment, we employed a Java implementation of TPC-W that emulates an online bookshop and deployed the application into the Amazon EC2 cloud with the objective of making reasonable predictions for an e-commerce website where any user request needs to be served instantaneously. For this experiment, our focus is to build up prediction models which are able to forecast a sudden increase in resource requirement in advance; for this reason, we customized the TPC-W client implementation to generate web requests in a linear fashion as it closely resembles the ramp-up phase of flash crowd web traffic [22]. However, future work will cover other workload patterns, such as exponential surge etc.

We used Amazon's auto-scaling API to define static scaling rules for resource provisioning of the online bookshop in EC2. During the experiment, we collected the aggregated Percentage of CPU usage of all the EC2 instances every minute; for this reason, CPU utilization sometimes goes beyond 100% for the actual and predicted utilization graphs.

The sampled CPU usage dataset was then used to train prediction models in order to make future resource usage forecasts in the cloud. In our experiment, we also estimated the accuracy of the prediction models in terms of the sliding window perspective; i.e. whether the sliding window has any impact on the prediction accuracy of the fitted models. Therefore, we conducted two experiments: one forecasts the future resource usage using the current value only; whereas the other one implements the sliding window strategy to make predictions for a different size of the input window.

The architecture and characteristics of the dataset collection stage using the TPC-W benchmark in the cloud is depicted in Fig. 3 and Table 1 respectively. It should be noted that the data collection phase includes only a few number of EC2 instances; however, it does not imply that we trade-off operational cost to the completeness of our study. Since the efficacy of the prediction model depends on the workload pattern only, the result will be the same even if the scale of the EC2 instances varies.

Table 1
Data collection scenario.

Number of samples	135
Benchmark duration	135 min
Sampling interval	1 min
Sampled resource	Aggregated percentage of CPU usage across all VM instances
Maximum number of EC2 instances	3
Minimum number of EC2 instances	1

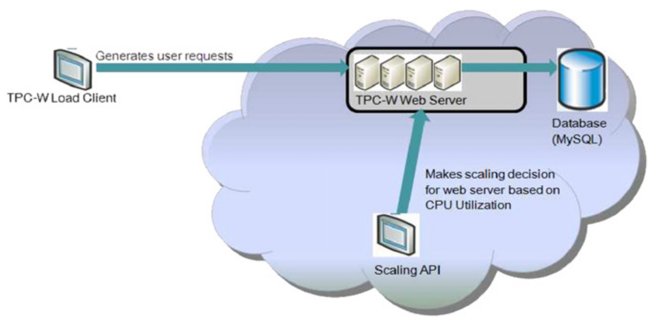

Fig. 3. Data collection architecture.

4.2. Training

We selected the prediction interval as 12 min because the setup time of virtual machine instances in the cloud is typically around 5–15 min; so the scaling system needs to request for new virtual instances 12–15 min ahead of time in order to meet up the surge in resource requirement in the cloud.

The following subsections provide a training description using machine learning algorithms, mentioned in Section 3.1 for constructing prediction models for future resource forecasting in the cloud.

4.2.1. Experiment 1 — training without sliding window

In this training approach, the prediction model observes the recent resource usage and then forecasts the future CPU usage on a 12 min interval depending on the currently seen value.

- **Error Correction Neural Network (ECNN).** For training the Neural Network using back-propagation, we use a third party package nnet [23] in the R-Project [24]. The parameters of our experiment are: learning rate $\rho = 0.7$, no of hidden layers $= 1$, number of hidden neurons $= 7$. The learning rate ρ should not be too small or too large; because if the rate is too small, then the prediction model will need more time to converge; again if it is too large, then it might oscillate around the local optima [13]. For this reason, we selected the learning rate ρ as 0.7 which is neither too small nor too large. The reason behind the selection of only one hidden layer is that most of the Neural Network based forecasting models [3] achieved effectiveness using one hidden layer. Again, the number of neurons in the hidden layer has been selected empirically.
- **Linear Regression.** In order to develop a Linear Regression based prediction model, we used another third party package dynlm [25] in the R-Project [24].

4.2.2. Experiment 2 — training with sliding window

In this training approach, the prediction model observes the recent resource usage over a sliding window of varying interval and then forecasts the future CPU usage requirement on a 12 min interval based on the currently monitored input window values. We changed the sliding window size and estimated the prediction

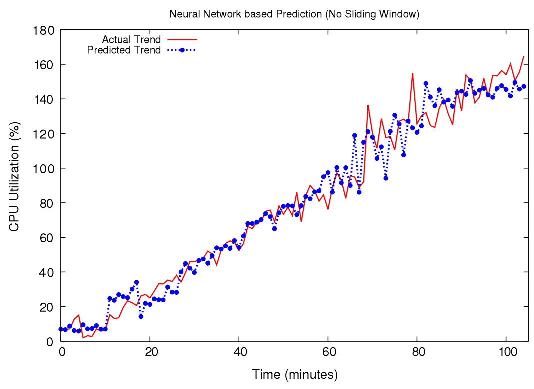

Fig. 4. CPU utilization vs. time in Neural Network approach with no sliding window.

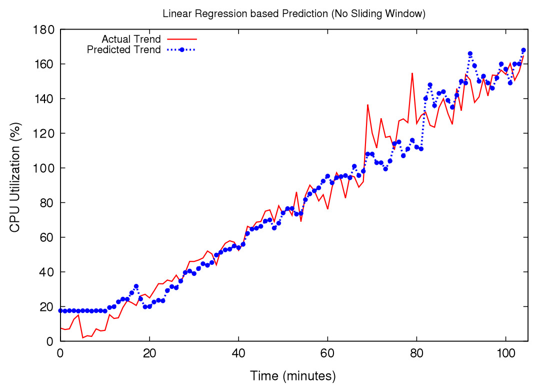

Fig. 5. CPU utilization vs. time in Linear Regression approach with no sliding window.

accuracy of the forecasting models produced by the learning algorithms specified in Section 3.1.

- **Error Correction Neural Network (ECNN).** The experimental setup for this sliding window based training of Neural Networks is almost similar to Section 4.2.1. The only difference here is the size of the input. Since we are using the sliding window technique in this training phase, therefore the input is not a single value but a vector of multiple values varying within a range [2, 9].
- **Multiple Linear Regression.** In the sliding window based Linear Regression approach, the training input is not a single value; however, an input vector of multiple values over an interval equal to the size of the sliding window. With the same experimental setup as in Section 4.2.1, we generated different linear forecasting models for different sliding window sizes for projecting future resource requirement in the cloud.

5. Results and discussion

We evaluated the prediction accuracy of Error Correction Neural Network and Linear Regression in terms of the metrics specified in Section 3.3, i.e. Mean Absolute Percentage Error (*MAPE*), Root Mean Squared Error (*RMSE*), R^2 prediction accuracy and *PRED*(25).

As specified in the previous section, we trained the prediction models without using the sliding window technique in Section 4.2.1; whereas we applied different sliding window size in Section 4.2.2. A comparison on the accuracy of the forecasting models with different learning algorithms and sliding window strategy is presented in Table 2.

Table 2
Comparison of predictive accuracy between Neural Network (NN) and Linear Regression (LR) models with and without the sliding window technique.

Metric	Without sliding window		With sliding window					
	NN	LR	NN			LR		
			Size = 3	Size = 6	Size = 9	Size = 3	Size = 6	Size = 9
MAPE	0.195	0.364	0.197	0.186	0.193	0.365	0.377	0.381
PRED(25)	0.819	0.810	0.800	0.857	0.857	0.838	0.848	0.848
RMSE	9.60	10.41	9.81	10.15	9.67	10.24	10.05	9.94
R^2	N/A	0.954	N/A	N/A	N/A	0.955	0.955	0.954

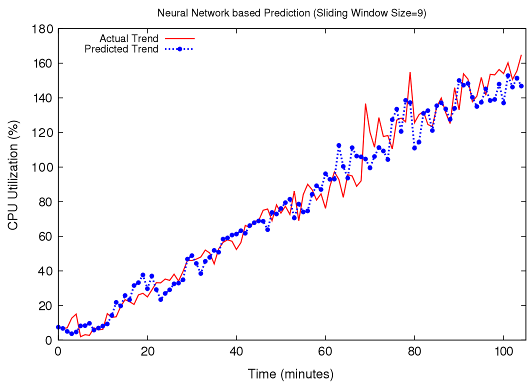

Fig. 6. CPU utilization vs. time in Neural Network approach for sliding window size = 9.

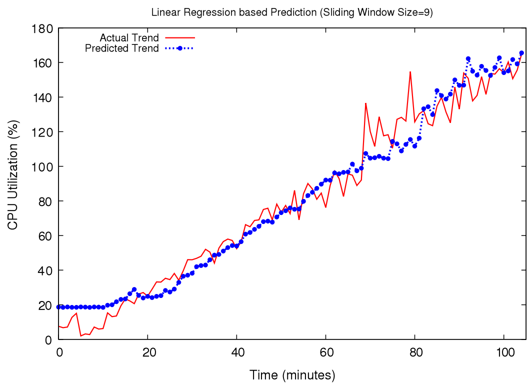

Fig. 7. CPU utilization vs. time in Linear Regression for sliding window size = 9.

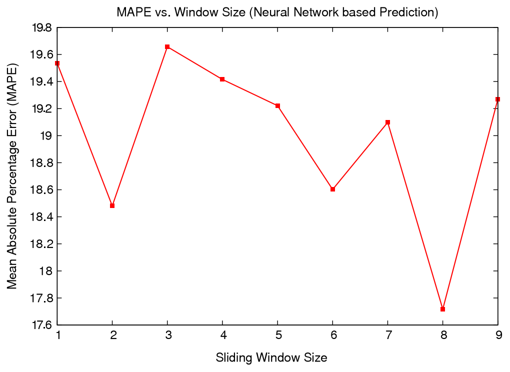

Fig. 8. Mean absolute percentage error vs. window size in Neural Network.

5.1. Comparison of prediction models

Figs. 4 and 5 correspond to the resource prediction models found from the experimental setup of Section 4.2.1 where prediction is made with respect to the most recent resource usage value in the cloud. It is apparent from the diagrams that the Neural Network prototype can closely exhibit the trend of the actual resource consumption in the cloud. From Table 2, it can be seen that the *MAPE* of Neural Network is 0.195 whereas it is 0.364 for Linear Regression; this implies that Neural Network performs better by yielding less percentage error than Linear Regression. Again, the *RMSE* of Neural Network and Linear Regression is 9.60 and 10.41 respectively; thus demonstrating the superiority of Neural Network over Linear Regression. According to *PRED*(25), Neural Network has more correct prediction rate (0.819) than that (0.810) of Linear Regression. Thus, all of these statistical metrics based comparisons go in favor of the Neural Network based prediction model for Experiment 1 with no sliding window method.

Alternatively, in Section 4.2.2, we applied the sliding window method which resulted in the projection models depicted in Figs. 6 and 7. In this case, the Neural Network based prototype is more accurate in predicting the actual pattern than that of Linear Regression. The assessment of the two models in terms of the evaluation metrics, listed in Table 2, also reveals the same notion. Similar to the no sliding window method, the *MAPE* of Neural Network is 50% smaller than that of Linear Regression for each of the different input window sizes, thus indicating better fit for Neural Network based prediction models. The *RMSE* of the Neural Network is slightly lower than Linear Regression which means that the Neural Network can predict the actual pattern more accurately by yielding less residuals. Finally, a higher value of *PRED*(25) ($>85\%$) for the Neural Network compared to Linear Regression demonstrates the enhanced efficacy of Neural Network models in terms of the rate of accurate predictions.

It should be noted, however, that the R^2 metric, typically used to explain the proportion of variation in a linear model, is not evaluated for the Neural Network model since there is no obvious way to compute it for a Neural Network prototype with one or more hidden layers. Hence, the Linear Regression model has not been compared to the Neural Network one based on the R^2 measure.

Next, we compare between different forecasting prototypes from the perspective of sliding window technique to determine whether the input window size has any influence on the prediction accuracy. From the prediction results of Table 2, it is noteworthy that the use of an input window has a positive effect on the accuracy of the prediction models. That is, the *MAPE* and *RMSE* becomes lower and the correct prediction rate goes higher with the use of the sliding window strategy. This is expected since the sliding window provides more specific knowledge about the underlying pattern within the neighborhood of the predicted data point; hence the projected value is more likely to comply with the sequence of the recently observed pattern.

5.2. Sensitivity analysis with varying sliding window size

Figs. 8–11 compare the responsiveness of the predictive accuracy to different size of the input window based on the performance metrics, *MAPE* and *PRED*(25). According to *PRED*(25), as the window size grows, the number of correct predictions goes up for both the Neural Network and Linear Regression models. Similarly, the *MAPE* measure declines with the expansion

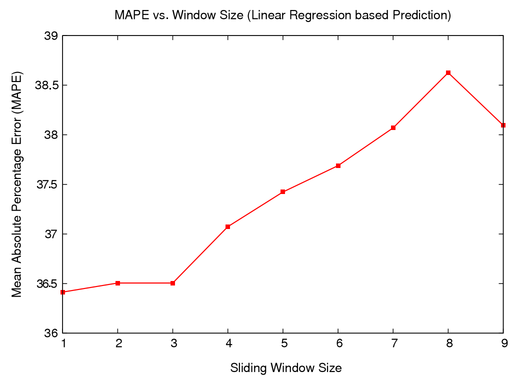

Fig. 9. Mean absolute percentage error vs. window size in Linear Regression.

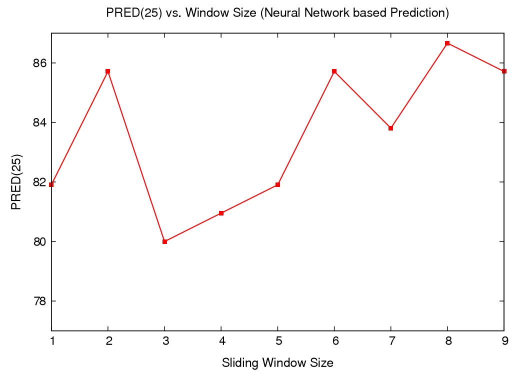

Fig. 10. *PRED*(25) vs. window size in Neural Network.

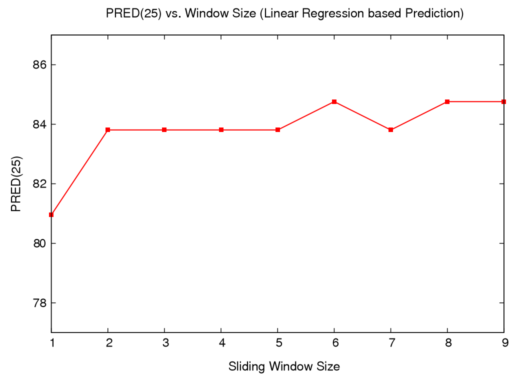

Fig. 11. *PRED*(25) vs. window size in Linear Regression.

of window size, reaching an optimal value of 17.72 at window size = 8 for the Neural Network model, implying better prediction accuracy with window size. In contrast, the *MAPE* performance of the Linear Regression model remains steady for a smaller window size (sliding window size of 1–3) and then gets worse as the window size increases; the underlying reason for this contradiction is the distribution of the actual resource utilization pattern which is linear in time.

To summarize, Neural Network based models with an optimal window size yield superior prediction accuracy than Linear Regression ones. To achieve more precision in projection, the fitted models can be trained with the production data in a regular interval; thus getting self-adaptive and more accurate, even in the presence random spikes. However, since the training of Neural Network models take significant time, the frequency of the training should be determined based on the underlying resource usage behavior of the application in the cloud.

6. Future work

In future, we plan to implement and evaluate our prediction models for a wide variety of workload generators (such as, Rain [26]). We also intend to integrate business-level SLAs (such as, performance, cost etc.) by converting these to utility functions which can be fed as additional input parameters into the prediction models. This strategy will facilitate the prediction framework to make business constraint based resource prediction which will eventually follow a dynamic rule generation part for adaptive and optimal resource provisioning in the cloud.

7. Conclusion

This paper provides an evolutionary approach to constructing an effective prediction model for adaptive resource provisioning in the cloud in order to facilitate dynamic and proactive resource management, scheduling and capacity planning for interactive e-commerce applications where immediacy and responsiveness are vitally important. Throughout the study, we have evaluated several major machine learning algorithms, in particular varying sliding window size with a view to providing accurate forecasting ahead of time. We exemplified our proposed prediction techniques in the context of the dataset obtained by using TPC-W, a benchmark which is well-established for e-commerce applications.

We also provided evaluation metrics for validating the accuracy of the proposed prediction techniques and compared the performance of the fitted models using these metrics. The prediction accuracy of our proposed learning models are encouraging and demonstrate superior effectiveness of Neural Network model with sliding window for forecasting resource utilization in the cloud. The integration of prediction strategies mentioned in this paper with auto-scaling process will certainly enhance the effectiveness of adaptive resource allocation procedure in the cloud in terms of both performance and cost.

Acknowledgment

NICTA is funded by the Australian Government as represented by the Department of Broadband, Communications and the Digital Economy and the Australian Research Council through the ICT Centre of Excellence program.

References

[1] A. Li, X. Yang, S. Kandula, M. Zhang, Cloudcmp: comparing public cloud providers, in: Internet Measurement Conference, pp. 1–14.
[2] A.M. Halavais, The Slashdot effect: analysis of a large-scale public conversation on the World Wide Web, Ph.D. Thesis, University of Washington, 2001.
[3] K. Nygren, Stock prediction — a Neural Network approach, Master's Thesis, Royal Institute of Technology, KTH, Stockholm, 2004.
[4] TPC, TPC-W Benchmark, Transaction Processing Performance Council (TPC), San Francisco, CA 94129-0920, USA, 2003.
[5] N. Van, H.D. Tran, F. Menaud, Jean-Marc, Autonomic virtual resource management for service hosting platforms, in: CLOUD'09 Proceedings of the 2009 ICSE Workshop on Software Engineering Challenges of Cloud Computing, pp. 1–8.
[6] A. Quiroz, H. Kim, M. Parashar, N. Gnanasambandam, N. Sharma, Towards autonomic workload provisioning for enterprise Grids and clouds, in: Grid Computing, 2009 10th IEEE/ACM International Conference, pp. 50–57.
[7] J.a.N. Silva, L. Veiga, P. Ferreira, Heuristic for resources allocation on utility computing infrastructures, in: MGC'08 Proceedings of the 6th International Workshop on Middleware for Grid Computing, ACM, New York, NY, USA, 2008, pp. 1–6.
[8] H.C. Lim, S. Babu, J.S. Chase, S.S. Parekh, Automated control in cloud computing: challenges and opportunities, in: ACDC'09: Proceedings of the 1st Workshop on Automated Control for Datacenters and Clouds, ACM, New York, NY, USA, 2009, pp. 13–18.

[9] E. Caron, F. Desprez, A. Muresan, Forecasting for cloud computing on-demand resources based on pattern matching, Technical Report, INRIA, 2010.
[10] J. Kupferman, J. Silverman, P. Jara, J. Browne, Scaling Into The Cloud.
[11] S. Arlot, A. Celisse, A survey of cross-validation procedures for model selection, Statistics Surveys 4 (2010) 40–79.
[12] B. Efron, G. Gong, A leisurely look at the bootstrap, the jackknife, and cross-validation, The American Statistician 37 (1983) 36–48.
[13] S. Theodoridis, K. Koutroumbas, Pattern Recognition, fourth edition, Academic Press, 2008.
[14] S. Weisberg, Applied Linear Regression, 3rd edition, in: Wiley Series in Probability and Statistics, Wiley, 2005.
[15] T. Dietterich, Machine learning for sequential data: a review, in: T. Caelli, A. Amin, R. Duin, i.c.k. de, Ridder, M. Kamel (Eds.), Structural, Syntactic, and Statistical Pattern Recognition, in: Lecture Notes in Computer Science, vol. 2396, Springer, Berlin, Heidelberg, 2002, pp. 227–246.
[16] N. Qian, T. Sejnowski, Predicting the secondary structure of globular proteins using Neural Network models, Journal of Molecular Biology 202 (1988) 865–884.
[17] T.J. Sejnowski, C.R. Rosenberg, Parallel networks that learn to pronounce English text, Journal of Complex Systems 1 (1987) 145–168.
[18] M. Jorgensen, Experience with the accuracy of software maintenance task effort prediction models, IEEE Transactions on Software Engineering 21 (1995) 674–681.
[19] N.J. Salkind, Encyclopedia of Research Design, Volume 10 of Issues, v. 47, Independence, 2002.
[20] S.B. Achelis, Technical Analysis from A to Z, McGraw Hill, 2001.
[21] R.C. Dodge JR, D.A. Menasce, D. Barbara, Testing e-commerce site scalability with Tpc-W, in: Proc. of 2001 Computer Measurement Group Conference.
[22] I. Ari, B. Hong, E.L. Miller, S.A. Brandt, D.D.E. Long, Managing flash crowds on the Internet, in: MASCOTS, pp. 246–249.
[23] W.N. Venables, B.D. Ripley, Modern Applied Statistics With S-PLUS, 3rd edition, in: Statistics and Computing, Springer, New York, 2001.
[24] R Development Core Team, R: A Language and Environment for Statistical Computing, R Foundation for Statistical Computing, Vienna, Austria, 2010.
[25] A. Zeileis, Dynlm: Dynamic Linear Regression, Cran, 2010.
[26] A. Beitch, B. Liu, T. Yung, R. Griffith, A. Fox, D.A. Patterson, Rain: A Workload Generation Toolkit for Cloud Computing Applications, Technical Report, University of California at Berkeley, 2010.

Sadeka Islam is pursuing her Ph.D. in University of New South Wales and National ICT Australia. Her research interests are in the areas of cloud computing, performance engineering, SLA based resource management and machine learning.

Jacky Keung is Assistant Professor in Software Engineering at the Hong Kong Polytechnic University. Prior to joining PolyU he was a Senior Research Scientist in the Software Engineering Research Group at National ICT Australia (NICTA), the top ICT research organization funded by the Australian Government. He also holds an academic position in the School of Computer Science and Engineering at the University of New South Wales, Sydney. Australia. He completed his B.S (Hons) in Computer Science from the University of Sydney, and received his Ph.D. from the University of New South Wales. His current research interests are in software engineering for cloud computing, machine learning, data-intensive pattern analysis, statistical software measurement and its application to project management, cost estimation, quality control and risk management, as well as software process improvement. His research achievements have been published in top quality journals and conferences, including IEEE Transactions on Software Engineering and have received numerous international recognitions and awards. He is a member of the Australian Computer Society, and a member of the IEEE Computer Society.

Kevin Lee is a researcher in the Managing Complexity Group at the National ICT Australia. His research interest is primarily in semantic technology and cloud computing, including formal ontologies, description logics, automated reasoning and belief change. His current project investigates the problem of automated resource provisioning in the cloud. Specifically, he makes use of machine learning techniques to predict resource usage. Based on this prediction, he devises algorithms to optimize between resource availability and costs. Furthermore, he has been the main investigator in a number of collaborative projects with industry as well as government and research institutions. He has also published in prestigious international conferences, including AAAI and TABLEAUX.

Anna Liu is currently a Research Leader at NICTA, heading up a team of 30 software engineering and software architecture researchers and Ph.D. Students. She also holds a Conjoint Associate Professorship at the School of Computer Science and Engineering, University of New South Wales. Her currently research interest is in cloud computing application design and development techniques for performance and scale. Prior to returning to academia, she was a Principal Software Architect at Microsoft Australia, leading various service-oriented architecture and enterprise application integration projects for financial services organizations and government agencies. Previously, she was a research leader at the CSIRO, in the area of component based software engineering and middleware technology evaluation. She has held visiting scientist appointment at the Software Engineering Institute SEI/CMU, and has presented numerous tutorials at international conferences including OOPSLA 2003, OOPSLA2005, ICSE2001 and WICSA2000.