2013 IEEE 37th Annual Computer Software and Applications Conference

Cloud Client Prediction Models Using Machine Learning Techniques

Samuel A. Ajila Akindele A. Bankole
Department of Systems and Computer Engineering, Carleton University
1125 Colonel By Drive, Ottawa K1S 5B6, ON Canada
{aabankol,ajila}@sce.carleton.ca

Abstract—**One way to proactively provision resources and meet Service Level Agreements (SLA) is by predicting future resource demands a few minutes ahead because of Virtual Machine (VM) boot time. In this research, we have developed and evaluated cloud client prediction models for TPC-W benchmark web application using three machine learning techniques: Support Vector Machine (SVM), Neural Networks (NN) and Linear Regression (LR). We have included two SLA metrics – Response Time and Throughput with the aim of providing the client with a more robust scaling decision choice. As an improvement to our previous work, we implemented our model on a public cloud infrastructure: Amazon EC2. Furthermore, we extended the experimentation time by over 200%. Finally, we have employed random workload pattern to reflect a more realistic simulation. Our results show that Support Vector Machine provides the best prediction model.**

Keywords-component; Cloud computing, Resource provisioning, Resource prediction, Machine learning

I. INTRODUCTION

The advent of cloud computing has made contemporary business owners (with limited capital for example) to rent and use infrastructure resources or services needed to run their businesses in a pay-as-you-use manner. This usage has been made possible by ubiquitous network connectivity and virtualization [13]. Specifically, Armbrust, M. et al. [6] described cloud computing as both the applications delivered as services over the Internet and the hardware and systems software in the data centre that provides those services. Cloud computing is therefore a sharp departure from the traditional method of owning a data center that warehouses infrastructure (networks, servers and cooling systems). While we applaud the numerous opportunities the cloud offers to providers and users, data security and confidentiality, availability of service and effective resource management techniques are some obstacles and challenges to the growth of cloud computing. In trying to meet up with both client Service Level Agreement (SLA) for Quality of Service (QoS) and their own operating cost, cloud providers are faced with the challenges of under-provisioning (a starvation or saturation of VM resources that leads to service degradation) and over-provisioning (underutilization and subsequent waste of VM resources). Under-provisioning would often lead to SLA penalty resulting into business revenue loss on the part of the cloud providers [6] and also a poor Quality of Experience (QoE) for the cloud client's customers (unacceptable response time for time critical applications for example). On the other hand, over-provisioning can lead to excessive energy consumption by the providers, culminating in high operating cost and waste of resources [6]; though this has no negative impact on the client. Provisioning generally requires some VM boot up time which has been reported to span various time durations, from between 5 and 10 minutes [2, 15], and between 5 and 15 minutes [18]. We reckon that during this time of system and resource unavailability, requests cannot be serviced which can lead to penalty on the part of the cloud providers. Multiplying this lag time over several server instantiations in a data center can result in heavy cumulative penalties. Therefore, automatically scaling (up and down) resources in response to load in order to save money, but without violating SLA is one area of research opportunity [5].

In our previous work [1], we developed and evaluated cloud client prediction models for TPC-W benchmark web application using three machine learning (ML) techniques: Neural Network (NN), Linear Regression (LR) and Support Vector Machine (SVM). We included the SLA metrics for Response Time and Throughput to the prediction model with the aim of providing the client with a more robust scaling decision choice. But in this work, we investigated the prediction capability of the same three ML techniques in a random traffic workload (TPC-W) pattern; a departure from the almost linear traffic workload (TPC-W) pattern in our previous work. In addition we have significantly increased the experimentation time by over 200% to provide sufficient validation dataset. Finally, we have implemented our model in a public cloud environment: Amazon Elastic Compute Cloud (Amazon EC2) infrastructure.

The contributions of this paper include:

- Insights into the effects of the various TPC-W workloads used in the experiment on the system performance of the database tier.

0730-3157/13 $26.00 © 2013 IEEE
DOI 10.1109/COMPSAC.2013.21

- The evaluation of the prediction capability of Support Vector Machine, Neural Network and Linear Regression using three benchmark workloads from TPC-W in:
  - An extended experimentation time frame of 532 minutes as opposed to our previous work of only 170 minutes.
  - Traffic patterns wherein workloads spike up and down and then stabilize in a randomized manner; a pattern we reckon to be somewhat realistic.

We still maintain the inclusion of business level SLA metrics thus providing wider and better scaling decision options for clients.

The rest of this paper is organized as follows: Section II discusses related work, while section III presents our methodology. Section IV evaluates the methodology from an experimental setup. We discuss our results in section V, and finally, present our conclusions and possible future work in section VI.

II. RELATED WORK

Several authors have worked in the area of resource usage prediction, for example, [8] presented a resource usage prediction algorithm that used a set of historical data to identify similar usage patterns to predict future usage. Though they reported impressive prediction capabilities of 0.9 to 4.8% prediction error, there was no record of the metric used to arrive at this value. In addition, their prediction was made for only the future 100 seconds. Going by the 5-10 minutes boot up time reported by [17, 2, and 15], the possibility of reduced QoS and QoE is highly probable. Quiroz, A. et al. [17] identified VM provisioning as a problem of the end-to-end data center provisioning and hence explored a decentralized online clustering approach to detect patterns and trends and use same for virtual resources provisioning. The decentralization allowed analysis of incoming jobs from multiple distributed queues. In addition, they employed a Quadratic Response Surface Model (QRSM) to capture workload behavior and thus estimate the application service time (response time). The authors in [16] investigated and evaluated both static and dynamic resource provisioning in three different traffic patterns – Weekly or Standard Oscillations, Large Spike and Random by using a scoring algorithm based on availability and cost. The authors established that dynamic resource provisioning outperforms the expensive static provisioning [1] by about 93% in cost reduction. However, their prediction model was based only on the CPU usage. Furthermore, there was no indication of the web application architecture. Finally, the workload mix if any was not discussed. A typical web application would have CPU, disk and network intensive workload mix. Wood, T. et al [25] focused on estimating the resource requirements an application would require in a virtual environment from utilization traces collected in the application's native environment. They applied a trace-based approach of historical data to forecast future CPU usage as a means of capacity planning. The mapping of native to virtual environment was hinged on a high correlation and proportionality between the two environments. They included various workload mixes in their work, something a typical enterprise application would exhibit. Using CPU traces from TPC-W and RUBiS applications, they employed Linear Regression to forecast future CPU utilization. While the authors reported a prediction error of less than 5% in the 90th percentile, they also used only CPU as a metric and reported their inability to include response time in their prediction model. As stated by Kundu, S. et al [14], using CPU alone as a scaling decision may be misleading, as an increase in CPU utilization may be due to inadequate memory or paging I/O, thus, in their most recent work [15] the authors went ahead to address memory and I/O in addition to CPU usage in their prediction model. They refined and employed NN and SVM machine learning techniques in their prediction model. By refining, they used the concept of clustering to create sub models which divide the input parameters or instances into non-overlapping sub-regions. Models were then built for each sub-region and then for performance prediction, the sub-model corresponding to the sub-region containing the input parameters were used. The improved NN resulted to a lower 90th percentile prediction error of 29.17% from 101.68% when a single NN model was used. Sadeka, I. et al, [18] analyzed the problem of resource provisioning from the application provider's point of viewpoint so that the hosted applications can make scaling decisions based on future resource usage. They also employed a set of machine learning techniques – NN and LR with both sliding and non-sliding window option. Though they reported impressive prediction accuracy (PRED 25[2]) of about 85.7% using NN, they did not report about the testing of the trained prediction model. It is not uncommon to have impressive training model prediction accuracy and poor test prediction accuracy. In fact, the possibility of over-fitting or under-fitting is highly probable in the absence of a suitable and independent test data for model validation. On the other hand, if test data are assumed to have been used, the ratio of training to testing dataset [13] was not mentioned. Furthermore, scaling decisions was based on only CPU resource utilization i.e. using the CPU's future resource utilization to scale up VM instances. The accuracy of using a single metric has been faulted by [14].

[1] Static provisioning determines the minimum number of machines required for 100% availability at peak periods of a given trial and then run the machines for the duration of the trial

[2] Percentage of observation whose prediction accuracy falls within 25% of actual value

Our work aims at analyzing the problem of resource provisioning from the Cloud client's perspective with the ability of the hosted application to make scaling decisions by not only evaluating the future resource utilization but also considering business SLA metrics of response time and throughput; thus providing a tripartite auto-scaling decision matrix. The prediction model that would be used to achieve this in a multi-tier web application is a set of machine learning techniques – Neural Network, Linear Regression and Support Vector Machine. These techniques would be evaluated using Mean Absolute Percentage Error (MAPE), Mean Absolute Error (MAE) Root Mean Square Error (RMSE) and PRED (25).

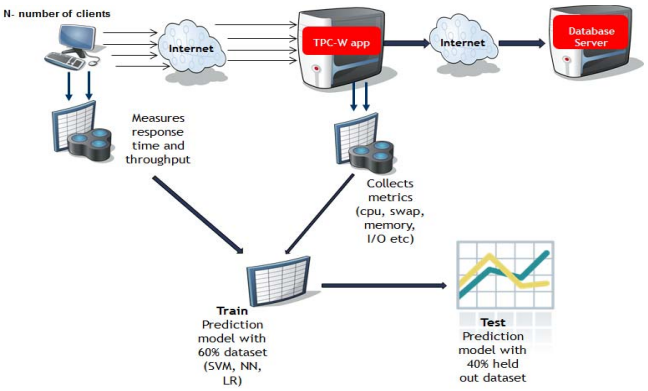


Fig. 1 Implementation methodology

III. METHODOLOGY

In this section, we describe our methodology for predictive resource provisioning for multi-tier web applications using machine learning to develop the performance prediction model (Fig. 1). NN and LR have been widely explored by several authors in building prediction models [17, 15, 24, and 13]. Recently SVM, a powerful classification technique [15] has been gaining significant popularity in time series and regression prediction [14, 12, and 18]. We introduce these learning techniques below.

A. Linear Regression

This is one of the staple methods in statistics and it finds application in numeric prediction especially where both the output or target class and the attributes or features are numeric [24]. Basically, the output or target class is expressed as a linear combination of the attributes, with preset weights:

$$y = w_0 + w_1a_1 + w_2a_2 + \ldots + w_ka_k \quad (1)$$

Where y is the target class and $a_1, a_2, \ldots, a_k$ are attribute values, and $w_o, w_1, \ldots, w_k$ are weights. The weights are calculated from the training data [24].

We can simplify (1) as: $y = \sum_{i=1}^{n} \sum_{j=1}^{k} w_{ij}a_{ij}$ (2)

where y, n and k are the target value, the number of instances and the number of features respectively. The goal of linear regression is to choose the coefficients w that will minimize the sum of squares of the differences between the actual and predicted values over all the training instances

B. Neural Network

A neural network is a two-stage regression or classification model, typically represented by a network diagram [23]. According to Trevor, H. et al [23], there is typically one output unit for regression problems though multiple quantitative responses can be handled in a seamless fashion. Derived features Z_m are created from linear combinations of the input, and then the target Y_k is modeled as a function of linear combinations of the Z_m,

$$\begin{aligned} Z_m &= \sigma(\alpha_{0m} + \alpha_m^T X), m = 1, \ldots, M, \\ T_k &= \beta_{0k} + \beta_k^T Z, k = 1, \ldots, K, \\ f_k(X) &= g_k(T), k = 1, \ldots, K, \end{aligned} \quad (3)$$

where $Z = (Z_1, Z_2, \ldots, Z_M)$, and $T = (T_1, T_2, \ldots, T_K)$

The activation function $\sigma(v)$ is usually chosen to be the sigmoid $\sigma(v) = \frac{1}{(1+e^v)}$. Sometimes, Gaussian radial basis functions are used for the $\sigma(v)$, producing what is known as a radial basis function network [23]

For regression, the sum-of-squares errors is used as the error function [23]

$$R(\theta) = \sum_{k=1}^{K} \sum_{i=1}^{N} (y_{ik} - f_k(x_i))^2. \quad (4)$$

The generic approach to minimizing $R(\theta)$ is by gradient decent, called back-propagation.

C. Support Vector Machines (SVM)

SVM has the advantage of reducing the problems of over-fitting or local minima. In addition, it is based on structural risk minimization as opposed to the empirical risk minimization of neural networks [13]. SVM now finds application in regression and is termed Support Vector Regression (SVR). The goal of SVR is to find a function that has at most ε (the precision by which the function is to be approximated [20]) deviation from the actual obtained target for all training data with as much flatness as possible [21]. Given training data (x_i, y_i) $(i = 1, \ldots l)$, where x is an n-dimensional input with $x \in R^n$ and the output is $y \in R$, the linear regression model can be written as [11] :

$$f(x) = < w, x > + b, \; w, x \in R^n, b \in R \quad (5)$$

where $f(x)$ is the target function and $<.,.>$ denotes the dot product in R^n. To achieve the flatness mentioned by [21], we minimize w i.e. $||w||^2 = < w, w >$. This can further be written as a convex optimization problem:

minimize $\frac{1}{2}||w||^2$ subject to the constraint

$$\begin{cases} y_i - < w, x_i > - b \leq \varepsilon \\ < w, x_i > + b - y_i \leq \varepsilon \end{cases} \quad (6)$$

Equation (6) assumes that there is always a function f that approximates all pairs of (x_i, y_i) with ε precision. However this may not be obtainable and thus [21] introduces slack variables γ_i , γ_i^* to handle infeasible constraints, with equation (6) leading to

Minimize $\frac{1}{2}||w||^2 + C\sum_{i=1}^{n}(\gamma_i + \gamma_i^*)$

$$\text{subject to } \begin{cases} yi - < w, x_i > -b \leq \varepsilon + \gamma_i \\ < w, x_i > +b - yi \leq \varepsilon + \gamma_i^* \\ \gamma_i, \gamma_i^* \geq 0 \end{cases} \quad (7)$$

The constant C>0 determines the tradeoff between the flatness of f and the amount up to which deviations larger than ε are tolerated. Equation (7) can be reformulated and solved to give the optimal Lagrange multipliers α *and* α^* with w and b given as

$$w = \sum_{i=1}^{n}(\alpha - \alpha^*)\, x_i \text{ and} \quad (8)$$

$$\text{b} = -\frac{1}{2} < w, (x_r + x_s) \quad (9)$$

x_r*and* x_s are the support vectors, thus inserting (8) and (9) into (5) yields

$$f(x) = \sum_{i=1}^{n}(\alpha - \alpha^*) < x_i, x > +b \quad (10)$$

This generic approach is usually extended for nonlinear functions. This is done by replacing x_i with $\varphi(x_i)$; a feature space that linearizes the relation between x_i and y_i [11].

Therefore, (10) can be re-written as:

$$f(x) = \sum_{i=1}^{n}(\alpha - \alpha^*)\; K < x_i, x > +b \quad (11)$$

where $K < x_i, x > = < \varphi(x_i)$, $\varphi(x)$ is the so called kernel function.

IV. THE SETUP

For our experiments, we used three different Amazon EC2 infrastructures running on Linux for the client, web server and database respectively.

- Client infrastructure: This was a High-CPU Instance with 1.7 GiB[3] of memory, 5 EC2 Compute Units[4] (2 virtual cores with 2.5 EC2 Compute Units each) and 350 GB of instance storage. We run the TPC-W emulator on this infrastructure
- Web server infrastructure: This has a 3.75 GiB of memory, 2 EC2 Compute Unit (1 virtual core with 2 EC2 Compute Unit) and 410 GB instance storage. The Java implementation of the TPC-W benchmark was deployed on a Tomcat web server environment
- Database infrastructure: This has 7.5 GiB of memory, 4 EC2 Compute Units (2 virtual cores with 2 EC2 Compute Units each) and 850 GB instance storage. MYSQL was the relational database management system used

Our experimental setup is divided into: Feature selection, Historical data collection, Feature reduction, Data preprocessing (scaling and normalizing), Training and testing of the historical dataset.

A. Feature selection

Usually, prediction models are based on a continuous observation of a number of specific features [19]. The following initial features were selected for the three target values (CPU utilization, response time and throughput) [4]:

i. DiskReadOps: This metric identifies the rate at which an application reads a disk
ii. DiskWriteOps: This metric identifies the rate at which an application writes to a hard disk
iii. DiskReadBytes: This metric is used to determine the volume of the data the application reads from the hard disk of the instance
iv. DiskWriteBytes: This metric is used to determine the volume of the data the application writes onto the hard disk of the instance
v. NetworkIn: This metric identifies the volume of incoming network traffic to an application on a single instance
vi. NetworkOut: This metric identifies the volume of outgoing network traffic to an application on a single instance
vii. Memory Utilized
viii. Memory Used
ix. Memory Available
x. Swap Utilized
xi. Swap Used

B. Data collection using TPC-W benchmark

TPC-W [22] is benchmark specification for e-commerce applications. The benchmark consists of a set of operations designed to exercise a web server/database system in a manner representative of a typical internet commerce application environment [7]. It has been used by several authors [24, 6] for resource provisioning and capacity planning [18]. Similar to [18], we have employed a Java implementation of TPC-W that emulates an online bookshop and deployed the application in a-two-tier architecture as depicted in Fig. 2. From the diagram, we can observe that the system resource metrics like CPU, memory etc were collected from the Web server while the response time and throughput were measured from the "client's" end. For this model, we have chosen to use the client as a reference point and the TPC-W has a remote browser emulator (RBE) that allows a single node to

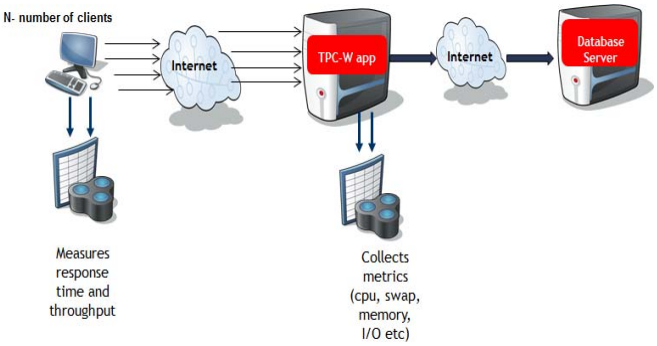
[3] 1 GiB = Gibibyte = 1.074GB

[4] One EC2 Compute Unit provides the equivalent CPU capacity of a 1.0-1.2 GHz 2007 Opteron or 2007 Xeon processor [4]

emulate several clients. The response time in this context is the time lag between when a page request is made to the reception of the last byte of the HTML response page. Similarly, the throughput is the total number of web interactions completed during an experimental run.

Fig.2 Model implementation architecture on Amazon EC2

The TPC-W offers three workload mixes – Shopping, Browsing and Ordering. Details of the various mix characteristics can be found in [7]. In our work we have used a combination of the three different workload mixes for every experimental run to create more realistic scenarios. Table 1 shows the client requests distribution at selected time intervals. By adjusting the number of emulated clients in a random pattern, we created a changing workload that sent request to the Web server in a continuous fashion throughout the duration of the experiment. Amazon EC2 has a web service that enables



one to monitor, manage and publish various metrics [4]. The traditional Top command in Linux was not used as the Top command shows metrics for the underlying host and not the actual instance [4]. We collected the data metrics (defined in Section A above) every 60 seconds. The duration for the entire experiment was 532 minutes. These data was used to build the prediction model from which forecast can be made for future resource requirement and SLA metrics of the Web server.

C. Feature reduction

The importance of selecting the right features for prediction modeling is very critical to reducing the potential source of error as the data mining principle of "junk in, junk out" means erroneous predictions can occur even if the prediction algorithm is optimal [19]. We employed Weka [12], a machine learning tool to determine the relevance of each feature in an instance to the target class (CPU, response time and throughput). Using attribute selection functionality, we eliminated the least correlated attributes: DiskReadOps, DiskWriteOps, DiskReadBytes, DiskWriteBytes, memory utilized, memory used, swap utilized and swap used.

D. Data preprocessing

During this phase, the 6 input features (including CPU utilization, Response Time and Throughput) are scaled to values between 0 and 1. Normalization or scaling is carried out by finding the highest value within each input in the 532 dataset, and dividing all the values within the same feature by the maximum value. The main advantage for normalizing is to avoid attributes in greater numeric ranges dominating those in smaller numeric range [10].

Table 1: Experimental workload mix

Workload Mix	**Time (minutes)**			
	1-7	175-182	490-497	504-511
Shopping users	84	292	300	160
Browsing users	52	228	192	160
Ordering users	52	212	268	160
Total user Requests	**188**	**732**	**760**	**480**

E. Training of Dataset

As discussed earlier, the goal of this work is to build prediction model that can forecast future resource requirement (using CPU utilization) and two business level SLA – response time and throughput. Towards this end, we used the normalized sampled dataset to train the prediction model. First, we trained the model with CPU utilization as the target class using the three machine learning techniques discussed above. Next, using the same dataset, we trained a new model for both response time and throughput. Thus, the same model was trained for both response time and throughput using the same parameters as we shall introduce shortly. We have used the metrics in Table 2 to evaluate both training and testing results of our models.

1) Model 1 - CPU utilization

- **Neural Network**: Using the Weka tool, we trained the model with the following parameters: learning rate ρ = 0.38, number of hidden layers = 1, number of hidden neurons = 4, momentum = 0.2 and epoch or training time =10000. These parameters gave the best results after several trials. Parameter selection is usually based on heuristics as there is no mathematical formula or theory that has been proposed to select the best parameters.
- **Linear Regression**: We also used the Weka tool to train the model. The only parameter set was the ridge parameter which was set to the default of 1.0E-8. Varying the value had no obvious impact on the target value
- **Support Vector Regression**: SVR has four kernels that can be used to train a model. They are: linear, polynomial, radial basis function (RBF) and sigmoid [13]. We tried the four different models and RBF showed promising result with the least MAPE value. This was expected as RBF can handle the case when the relationship between features and target value is nonlinear [10]. Before training, we used the Grid Parameter Search for Regression with cross validation

(v-fold cross validation) [9] to estimate the C and λ. The cross-validation is a technique used to avoid the over fitting problem [12, 9]. The search range for C was between 2^{-3} to 2^{5} and that of λ between 2^{-10} and 2^{2}. These values are purely heuristics with guidance from various author's work [12, 9]. The search returns the optimal C and λ by using the Mean Square Error to evaluate the accuracy of the various C and λ combinations. The best C and λ was 14 and 0.0092. Using these parameters, we trained the model with the Radial Basis Function (RBF) Kernel.

We used these parameters to train the historic dataset with the WEKA machine learning tools and Table 3 shows the values of the evaluation metrics.

2) Model 2 – Response time and Throughput

We approached the business SLA metrics in similar way as Model 1. For Throughput, SVR's C and λ were 8 and 0.009 respectively. NN values for ρ, hidden layer, hidden neurons and momentum were 0.4, 1, 3 and 0.2 respectively. Finally the ridge parameter for LR was 1.0E-8. Similarly, for Response time; SVR's C and λ were 1.05 and 0.009 respectively. NN values for ρ, hidden layer, hidden neurons and momentum were 0.5, 1, 3 and 0.2 respectively. The ridge parameter for LR was 1.0E-8.

Table 2 Performance metrics and their calculations

Metric	Calculation
MAPE[5]	$\frac{1}{n}\sum_{i=1}^{n}\frac{\lvert a_i-p_i\rvert}{a_i}$ where a_i and p_i are the actual and predicted values respectively
RMSE[6]	$\sqrt{\frac{\sum_{i=1}^{n}(a_i-p_i)^2}{n}}$
MAE[7]	$\frac{1}{n}\sum_{i=1}^{n}\lvert p_i-a_i\rvert$
PRED 25	No. of observations with relative error ≤ 25% / No. of observation

F. Testing of trained model

This step is very significant as it is possible to obtain impressive results for training data but dismal results when it comes to testing. Furthermore, prediction accuracy is based on the held out test dataset. We used a training-to-testing ratio of 60%:40% as this gave the optimal prediction output for our model. We have adopted a 12 minute prediction interval to test our prediction model. This is based on the reports from previous works [2, 15] regarding VM boot up time and motivation from the work of [18]. However, we have reported the prediction trend at the 9th, 10th, 11th and 12th minute. We added this to check for consistency and reliability in the prediction models of SVR, NN and LR.

V. RESULTS AND DISCUSSION

The objectives of this research work are to evaluate the accuracy of the selected machine techniques in forecasting future resource usage for random workload traffic patterns over an extended period of time. In addition, the inclusion of business level SLA metric to the prediction is considered. CPU utilization prediction model and SLA (response time and throughput) prediction model are used to achieve and meet the objectives.

A. Results

The training and test results for CPU utilization are displayed in tables 3 and 4 respectively. In addition, table 5 shows the 9 – 12 minute step prediction MAPE metric for the test dataset. For Response Time and Throughput, tables 6 and 8 shows the metric results for test dataset while tables 7 and 9 shows the 9-12 minutes step prediction.

Table 3: CPU utilization Training performance metric

Model	MAPE	RMSE	MAE	PRED(25)
SVR	12.66	5.00	2.48	0.64
NN	29.41	19.84	2.98	0.59
LR	16.42	4.83	2.63	0.51

Table 4: CPU utilization Test performance metric

Model	MAPE	RMSE	MAE	PRED(25)
SVR	22.84	11.84	8.74	0.64
NN	50.46	31.08	19.82	0.25
LR	36.19	16.18	15.98	0.68

Table 5: CPU utilization step prediction for MAPE

Model	9-min	10-min	11-min	12-min
SVR	22.31	22.69	22.78	22.84
NN	53.07	49.90	45.62	50.46
LR	34.43	35.14	35.92	36.19

Table 6: Response Time Test dataset performance metric

Model	MAPE	RMSE	MAE	PRED(25)
SVR	14.30	1.74	1.33	0.83
NN	35.15	3.80	2.94	0.48
LR	87.97	8.78	7.88	0.02

Table 7: Response Time step prediction for MAPE

Model	9-min	10-min	11-min	12-min
SVR	14.21	14.39	14.37	14.30
NN	39.52	34.50	32.44	35.15
LR	86.88	96.65	96.46	87.97

[5] Mean Absolute Percentage Error
[6] Root Mean Square Error
[7] Mean Absolute Error

Table 8: Throughput Test dataset performance metric

Model	MAPE	RMSE	MAE	PRED(25)
SVR	22.07	3.22	2.41	0.67
NN	38.90	6.12	4.46	0.47
LR	156.08	23.39	19.44	0.04

Table 9: Throughput Time step prediction for MAPE

Model	9-min	10-min	11-min	12-min
SVR	21.38	21.62	21.80	22.07
NN	38.84	36.87	37.39	38.90
LR	122.73	135.44	146.51	156.08

B. Discussion

The actual CPU utilization as observed in any of figures 3, 4 and 5 shows a more random utilization level with burstiness compared to our previous work [1]. As stated earlier, we have significantly increased the experimentation time to provide sufficient dataset for both training and validation (testing) of the model; a step towards objectivity. We summarize our results below.

Using the metrics defined in table 2, the training result from table 3 (Model 1 – CPU utilization) shows that SVR outperforms NN and LR in the MAPE and PRED (25) metrics. We have reported the metrics for the training dataset to emphasize the importance of optimizing the results of training and testing data. Increasing the accuracy of training data leads to over fitting and subsequent high and unacceptable MAPE metric for the test data. This approach was also applied to the SLA models. The test dataset results (table 4) for SVR gave the best overall prediction metric [MAPE, RMSE, MAE and PRED (25)]. Compared to the result of our previous work [1], we observe that the MAPE and PRED (25) values in this work is less accurate. This can be attributed to the random workload pattern mix used in this experiment. Table 1 for example shows a significant drop in users between the 497th and 511th minute. SVR has shown its strong forecasting ability over NN and LR even in a random traffic pattern. The NN model shows very poor metric values that are even below LR.

Figures 3, 4 and 5 respectively display the graph of the actual and predicted CPU utilization for LR, NN and SVR. SVR is most responsive to random or nonlinear user request pattern; thus displaying a strong generalization property. It can be observed that both NN and LR predicted below zero CPU utilization at some time interval; a serious anomaly. We attribute this to the weak generalization property especially in a random workload pattern. We can draw this conclusion because, compared to our previous work, the two major changes asides implementing our work on Amazon EC2 are; increased randomness in workload pattern and experimentation time. We also present the MAPE step prediction from the 9th to 12th minute. SVR shows consistency in prediction across the steps as shown in table 5.

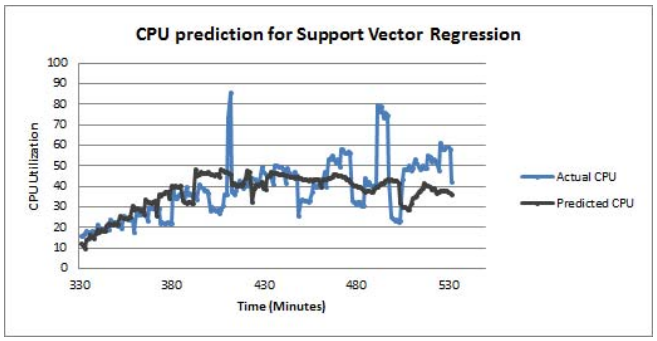


Fig. 3 Actual and Predicted CPU utilization for LR

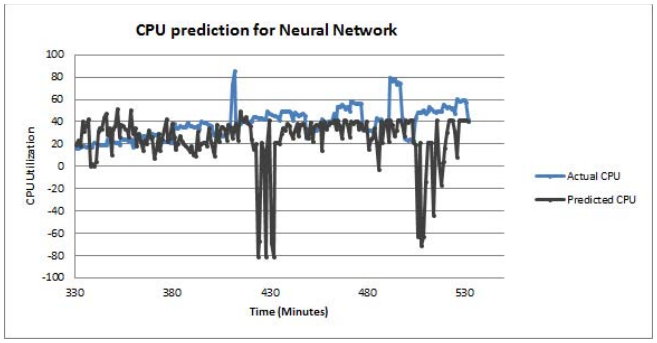


Fig. 4 Actual and Predicted CPU utilization for NN

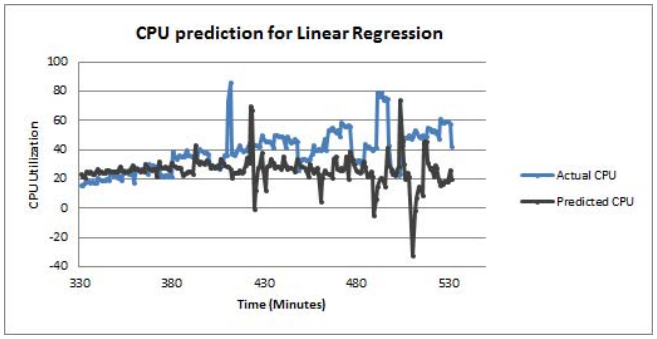


Fig. 5 Actual and Predicted CPU utilization for SVR

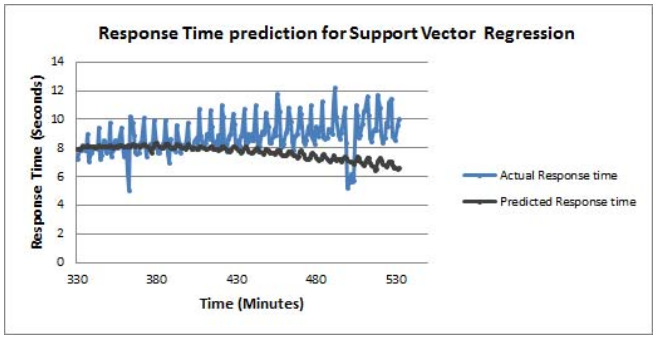


Fig. 6 Actual and Predicted Response Time – SVR

NN is slightly erratic over the step prediction. LR comes second to SVR in stability of prediction.

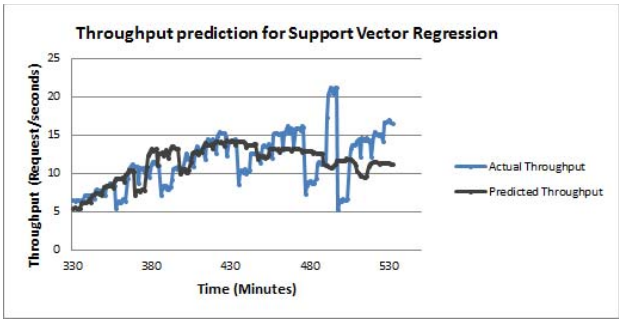


Fig. 7 Actual and Predicted Throughput – SVR

For the Response Time prediction model, SVR again comes on top of the other two; NN and LR (table 6). The consistency of SVR across the 9th to 12th minute step is also observed again (table 7). LR this time around has a very poor and unacceptable MAPE value. In fact, the PRED (25) metric is just 2%; thus inferring that the LR model cannot be relied upon for forecasting. Fig.6 shows the actual and predicted Response Time graph for SVR using the test data while Fig. 1A and 1B in Appendix B shows the NN and LR plots for these models respectively. From the figures we observe a lot of surge pattern in the response time.

Finally on the SLA metric, SVR had the best overall prediction metric for throughput (table 8) despite the burstiness experienced (Fig. 7). NN comes a distant second while LR again presents unacceptable metric outputs. Generally, preference is given to MAPE, RMSE and MAE as PRED (25) deals with a range of value as opposed to specific value. For example, a value that falls between 2 and 24.99 would meet the requirement of PRED (25) but further increase the RMSE, MAE and MAPE values. Again, the prediction result of SVR is persistent as seen in table 9. NN also showed more stable prediction accuracy. Fig. 2A and 2B show the throughput plot of NN and LR respectively.

From the two models (CPU and Response Time, Throughput SLA parameters), SVR is the most preferred model for forecasting. It has displayed a strong generalization property and the absence of over fitting. Furthermore, its prediction accuracy is consistent over time compared to the NN and LR.

VI. Conclusion and Future Work

In this paper, we have built three forecasting models using Linear Regression, Neural Network and Support Vector Regression for a two-tier TPC-W web application in a public cloud environment – Amazon EC2. We employed a more random workload pattern over an extended period of time compared to our previous work [1] and we have obtained impressive results. The Support Vector Regression model displayed superior prediction accuracy over both the Neural Network and Linear Regression in a 9 to 12 minutes window. Consequently, cloud customers can employ SVR to build their prediction models. Furthermore, the addition of business level SLA metrics (response time and throughput) into the prediction model paves the way for a three-fold combination decision matrix for adaptive resource allocation; specifically scaling up VM infrastructure. We find this very useful as response time and throughput may have degraded long before an application reaches its set CPU utilization threshold.

We have used this work to investigate the impact of the workload pattern on the Database server and we discovered that it was a bottleneck. We mitigated the bottleneck by using a High-Memory/CPU infrastructure as described in Section IV. Therefore, we plan extending the prediction model to include the Database server in our future work.

Acknowledgement: This research is sponsored in part by Canada NSERC discovery grant number 250271.

References

[1] Akindele A. Bankole and Samuel A. Ajila, *Cloud Client Prediction Models for Cloud Resource Provisioning in a Multitier Web Application Environment*, The 7th IEEE International Symposium on Service-Oriented System Engineering (IEEESOSE 2013), San Francisco Bay, USA March 25 – 28, 2013

[2] "Amazon CloudWatch Developer Guide API Version 2010-08-01", 2013. [Online]. Available: http://awsdocs.s3.amazonaws.com/AmazonCloudWatch/latest/acw-dg.pdf

[3] "Amazon elastic compute cloud (amazon ec2)", 2013. [Online]. Available: http://aws.amazon.com/ec2

[4] "Amazon Web services Discussion Forums", 2013. [Online]. Available: https://forums.aws.amazon.com/thread.jspa?threadID=67697

[5] Armbrust, M. et al. "A view of cloud computing". *Commun. ACM.* 53, 4 pp. 50–58, Apr. 2010.

[6] Armbrust, M. et al. 2009. Above the Clouds: A Berkeley View of Cloud Computing.

[7] Cain , H. W. et al., "An Architectural Evaluation of Java TPC-W" in *Proceedings of the Seventh International Symposium on High- Performance Computer Architecture*, 2001

[8] Caron, E. et al., "Forecasting for Grid and Cloud Computing On-Demand Resources Based on Pattern Matching" *2nd International Conference on Cloud Computing Technology and Science (CloudCom).* pp.456-463, November, 30 2010.

[9] Chih-Chung, C. and Chih-Jen , L., "LIBSVM : a library for support vector machines". ACM Transactions on Intelligent Systems and Technology, 2:27:1--27:27, 2011. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm

[10] Chih-Wei, H. et al, "A practical guide to support vector classification". Technical report, Department of Computer Science and Information Engineering, National Taiwan University, Taipei, 2003. [Online]. Available: http://www.csie.ntu.edu.tw/~cjlin/libsvm/

[11] Guosheng, H. et al., "Grid Resources Prediction with Support Vector Regression and Particle Swarm Optimization," *3rd International Joint Conference on Computational Science and Optimization (CSO),* vol.1, pp.417-422, May 2010.

[12] Hall, M. et al., "The WEKA Data Mining Software: An Update", SIGKDD Explorations, Volume 11, Issue 1. 2009

[13] Hilley, D. Cloud Computing: A Taxonomy of Platform and Infrastructure-level Offerings: 2009. https://smartech.gatech.edu/handle/1853/34402. Accessed: 2013-01-24.

[14] Khashman, A. and Nwulu, N.I, "Intelligent prediction of crude oil price using Support Vector Machines", *in IEEE 9th International Symposium on Applied Machine Intelligence and Informatics (SAMI),* pp.165-169, January, 2011.

[15] Kundu, S. et al. "Application performance modeling in a virtualized environment". *IEEE 16th International Symposium on High Performance Computer Architecture (HPCA)* pp. 1-10. January, 2010.

[16] Kundu, S. et al., "Modeling virtualized applications using machine learning techniques", *in Proc. Of 8th ACM SIGPLAN/Sigpos conference on Virtual Execution Environments*, pp3 – 14, London, UK 2012

[17] Kupferman, J. et al., "Scaling Into the Cloud". University of California, Santa Barbara, Tech. Rep. http://cs.ucsb.edu/~jkupferman/docs/ScalingIntoTheClouds.pdf. 2009.

[18] Quiroz, A et al., "Towards autonomic workload provisioning for enterprise Grids and clouds" in *Grid Computing, 2009 10th IEEE/ACM International Conference* pp 50-57, October, 2009

[19] Sadeka, I. et al., "Empirical prediction models for adaptive resource provisioning in the cloud", *Future Generation Computer Systems*, vol. 28, no. 1, pp 155 – 165, January, 2012

[20] Sakr, G.E et al., "Artificial intelligence for forest fire prediction" *IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM),* pp.1311-1316, July 2010.

[21] Sapankevych, N and Sankar, R., "Time Series Prediction Using Support Vector Machines: A Survey," Computational Intelligence Magazine, IEEE, vol.4, no.2, pp.24-38, May 2009.

[22] Smola, A.J and Scholkopf, B., "A Tutorial on Support Vector Regression" in *Statistics and Computing* vol 14, pp. 199 – 222, 2004.

[23] TPC, TPC-W Benchmark, Transaction Processing Performance Council (TPC), San Francisco, CA, USA, 2003.

[24] Trevor, H. et al. "The Elements of Statistical Learning: Data Mining, Inference, and Prediction", New York: Springer, 2009.

[25] Witten, I. H and Frank, E. "Data Mining Practical Machine Learning Tools and Techniques with Java Implementations", San Diego: Academic Press, 2000.

[26] Wood, T. et al., "Profiling and Modeling Resource Usage of Virtualized Applications" *Proceedings of the 9th ACM/IFIP/USENIX International Conference on Middleware*, New York, USA pp. 366-387, 2008.

Appendix A

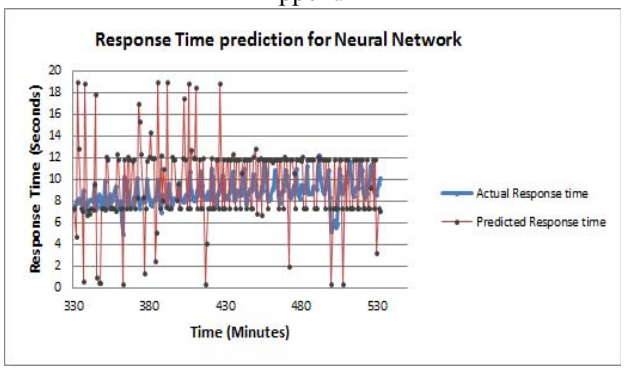


Fig. 1A Actual and Predicted Response Time – NN

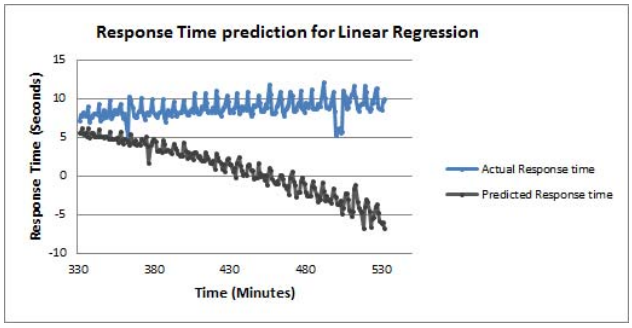


Fig. 2A Actual and Predicted Response Time –LR

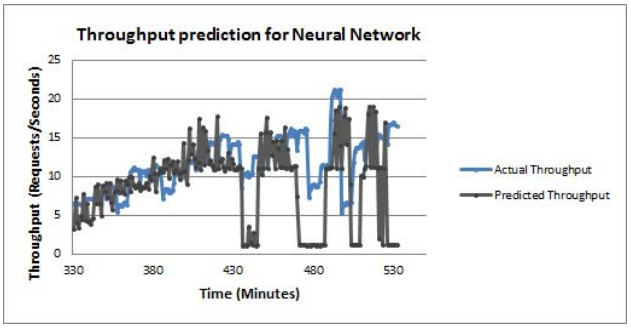


Fig. 3A Actual and Predicted Throughput – NN

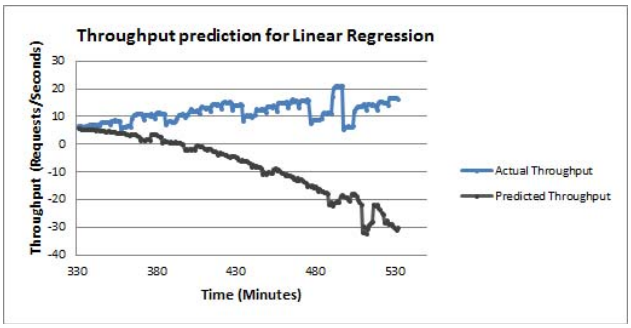


Fig. 4A Actual and Predicted Throughput – LR