
Scalable Computing: Practice and Experience, ISSN 1895-1767, http://www.scpe.org
© 2020 SCPE. Volume 21, Issues 4, pp. 661–672, DOI 10.12694:/scpe.v21i4.1805

# A DYNAMIC PREDICTION FOR ELASTIC RESOURCE ALLOCATION IN HYBRID CLOUD ENVIRONMENT

VIPUL CHUDASAMA, MADHURI BHAVSAR*

**Abstract.** Cloud applications heavily use resources and generate more traffic specifically during specific events. In order to achieve quality in service provisioning, the elasticity of resources is a major requirement. With the use of a hybrid cloud model, organizations combine the private and public cloud services to deploy applications for the elasticity of resources. For elasticity, a traditional adaptive policy implements threshold-based auto-scaling approaches that are adaptive and simple to follow. However, during a high dynamic and unpredictable workload, such a static threshold policy may not be effective. An efficient auto-scaling technique that predicts the system load is highly necessary. Balancing a dynamism of load through the best auto-scale policy is still a challenging issue. In this paper, we suggest an algorithm using Deep learning and queuing theory concepts that proactively indicate an appropriate number of future computing resources for short term resource demand. Experiment results show that the proposed model predicts SLA violation with higher accuracy 5% than the baseline model. The suggested model enhances the elasticity of resources with performance metrics.

**Key words:** Cloud Computing, Autoscaling, Elasticity, SLA violation, Hybrid cloud

**AMS subject classifications.** 68M14

**1. Introduction.** Cloud computing technology enables users to access data through a virtual environment. Data centers are nowadays offering high-performance cloud services to the users as per demand. Accessibility of data is quick and cost-effective due to the elasticity and pay per use model [1] . The majority of the utility services, including logistics to the health-care support, are using resources provided by the cloud. In Smart Cities, the Internet of Things (IoT) enabled applications deployed in the cloud to use computing processes and resources such as CPU, software, and hardware devices [3]. One of the features of the cloud is on-demand services in which resources are provided as per demand, which increases customer satisfaction. In the purest form, cloud computing provides solutions by storing and accessing data or programs as service for users.

In many organizations, data resides in local infrastructure like small clusters. Organizations develop policies to access data for their employees. Due to the expansion and other factors, demand for resource virtualization is increasing day by day. A hybrid cloud is an affordable solution to deal with a burst in requirements for specific events [2]. In the cloud bursting model, an application that is running in local infrastructure and bursts to a public cloud for more resources. This type of Hybrid model has the advantages of cost reduction and scalability with data sensitivity [4].

In such scenarios, the elasticity of services to satisfy the need for resources increases user satisfaction. In order to achieve elasticity of the services and promised QoS, there is a need for resource management in Cloud Data centers. As per average demand, CPU utilization is 15 % to 20 % in normal state and follows a linear relationship in peak demand [5]. In such dynamics, requirements for resources are market-driven. So the main goal is to provide Elastic services with a dynamic policy which adds or removes storage and computing resources to enhance the application performance. Major Cloud providers (AWS,Google) provide elasticity features based on some metrics (CPU utilization, Memory). The use of elasticity through such auto scale mechanisms can satisfy the peak demand for application and guarantees QoS requirement.

Dynamic provision of resources is one of the complex tasks in distributed systems. Distributed systems such web servers, big-data cluster and grid require efficient resource management to provide the elastic services. Fig. 1.1 represents a scenario of web servers in data centers which provide cloud service. The data center

*Department of Computer Science and Engineering, Nirma University, Ahmedabad, India (vipul.chudasama@nirmauni.ac.in, madhuri.bhavsar@nirmauni.ac.in)

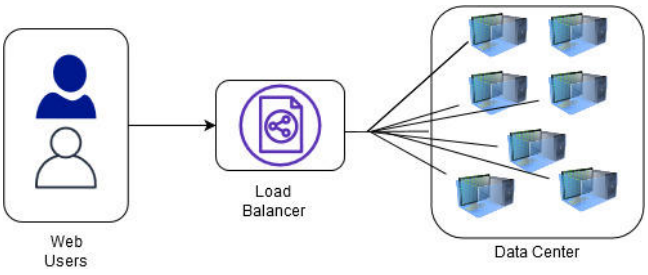


FIG. 1.1. *Elastic Cloud service*

manages the allocation and deallocation of servers through auto-scale to optimize resources. Flexible allocation and deallocation of resources are achieved with an auto-scale mechanism. An auto-scale mechanism is deployed on every workload in a cloud to maintain the resources in a balanced state. The balanced state of resources will help providers frame the policies to scale down idle resources and save energy consumption. The role of cloud providers with such optimizations in their operational environment helps in Green computing while maintaining Service Level Agreement (SLA) with end users [6]. Such an auto-scale mechanism is further divided in two classes: (i) Reactive approach, where static rules trigger to match the requirement of resources; (ii) Proactive approach, which is used to forecast the workload to meet the resource demand. In both classes, highly variable workload patterns are used to get future resource utilization demands, which is a challenging task. Such foreseen future demand helps allocate applications to the system that will improve the utilization with the help of time series analysis [15].

In recent works, autoregressive models such as ARIMA, AR, ARMA are proposed to predict metrics (system load) from monitoring service [8], which follows linear patterns. In Linear models, a dependent variable that is regressed on a number of independent variables while some non-linear features cannot be interpreted. However, non-linear patterns of resource usage are also addressed in some proposals to forecast future demand[9][10]. The neural network-based regression models are used to capture the non-linearity of resource usage. Our contribution in this work is to propose a hybrid model with Deep learning (LSTM) and Queuing theory to estimate resource requirements. We have considered the user's feedback to optimize the mechanism, which was not considered in previous studies. The main contributions of the paper are as follows:

- Design of an auto-scaling method based on deep learning is proposed to enhance the service through resource management.
- An LSTM based regression model to predict host load is presented.
- A queuing model to forecast the number of resources under the provision in the hybrid cloud.
- An assessment using real-server load information is given.

The rest of the paper is organized as follows: Section 2 discusses related work on autoscale strategies using Machine learning(ML) for workload prediction. Section 3 defines the proposed predictive approaches Section 4 discusses proposed predictive model and algorithm , Section 5 discuss experiment and analysis of proposed work, Section 6 and 7 discuss results and conclusions.

**2. Related Work.** The majority of research focuses on auto scaling mechanisms to manage the resources of Cloud. Web application workload is highly dynamic in nature. Recent studies in [11-13] have provided extensive review of resource management approaches.

As discussed in [14], the workload can be classified into five different classes such as once-in-a-lifetime, static, continuously changing, periodic, and unpredictable. Bayesian classifiers consist of probabilistic gives good results on CPU intensive tasks and the Markov method provides reliability for memory-intensive tasks. Auto-scaling of web applications with time series forecasting methods are discussed in [15-17]. A workload factoring technique was discussed in [18] for a hybrid cloud where a threshold-based technique was used to classify into two classes. The classes capture the base workload and flash workload. Threshold-based techniques are also applied by public cloud providers(Amazon EC2,Microsoft). The policies are framed based on the metrics provided, to scale up the resources or scale down resources of the environment. Improvement in the threshold-

based policy was suggested in [19] at fine-grain level by adding more levels in order to apply the decision of scaling the resources.

A Neural Network based workload prediction which implemented differential evolution and particle swarm optimization to estimate the workload was discussed in [20]. A Reinforcement Learning (RL) is a technique in which decision-making agents learn and decide best action without prior knowledge of the system. Actions are in the form of adding or removing the resources to obtain the highest rewards (e.g minimum response time, maximum throughput). For the cloud environment, authors [21] applied a Q-learning algorithm with an optimal scaling policy to reduce the execution time. Machine learning-based techniques like Support Vector Machine (SVM), Artificial Neural Network (ANN) model, and deep learning have found to provide excellent performance with time series data based on the minimization principle [27-28]. A number of authors have estimated resource scaling decisions with time series techniques like auto-regression, moving average and exponential smoothing. Machine learning models explore techniques like K-Nearest Neighbour (KNN), Support Vector Machine (SVM), Neural Network (NN), and Linear Regression which involve analysis, training, and prediction of metrics to optimize resource management in cloud [29].

The Deep Learning techniques are also preferred to explore features of Web traffic. Resource allocation and power management framework is discussed in [23] to predict the workload using long short-term memory (LSTM) recurrent neural network. The workload prediction mechanism with four LSTM units to improve the accuracy of the predictor has been presented in [24]. Another proposal has been presented to predict the workload of VM using DL approach [25]. Deep learning explores representational learning to generate the model which estimates the future values of workload in the cloud [30]. Queue Network Model (QN) can be used to analyse the performance of a distributed system. In a Distributed system, multiple servers handle the client requests with the optimization of parameters such as average queue time and average response time. These methods can be an open queue or closed queue. In [26], the authors proposed a proactive framework to improve the QoS of web application users' workload behaviour to allocate the virtual machines (VM) using the queuing model M/G/m.

Overall, the goal of proactive approaches suggested by Machine learning is to capture patterns or trends in the historical data. Prediction metrics obtained by these proactive approaches improve resource allocation in the cloud. In order to achieve elasticity, we explored the proactive approaches which provide future demand for resources and extended it with a queuing approach to propose a performance model in cloud.

**3. The proposed predictive approaches.** In Predictive techniques, time series data must be organized in a training set by splitting it with time series variable (T). So time series training set is defined as input and output set X, Y, respectively. Here we have applied a sliding window approach to process log data. Also, data is normalized before the process.

$$X = \begin{bmatrix} x_1 & x_2 & \dots & x_k \\ x_2 & x_3 & \dots & x_{k+1} \\ \vdots & & & \\ x_{n-1} & x_{n-2} & & x_{n-k} \end{bmatrix}, Y = \begin{bmatrix} x_{k+1} \\ x_{k+2} \\ \vdots \\ x_n \end{bmatrix}$$

SVM (Support Vector Machine) and Deep Learning (DL) are state-of-the-art machine learning methods to solve problems having time series data. SVMs learn from non-linear training data and map high dimensional feature space using kernel functions. An SVM regression model explores the number of requests application on a host to infer new feature space with the following function:

$$\hat{y}_t = b + \sum_{m=1}^{M} w_m \times K(x_t, x_m) \tag{3.1}$$

where $w_m$ are the weight vector ($W$) and $x_t$ is the time series data window at time t and b is a constant (bias term); and K is the kernel function. The optimal weight vector, $W$, is output of SVM to minimizes the regularized risk, $R_{reg}$, defined in equation 3.2. The Vapnik loss measured as error between predicted and actual

data is defined in equation 3.3. Here constant $C$ and $\epsilon$ are chosen by the user and are data dependent [34].

$$R_{reg} = \frac{1}{2}\sum_{m=1}^{M} w_m^2 + C\sum_{m=1}^{M} L_\epsilon(y_m, \hat{y}_m) \tag{3.2}$$

$$L_\epsilon\left(y_m, \hat{y}_m\right) = \begin{cases} |y_m - \hat{y}_m| - \epsilon & \text{if} |y_m - \hat{y}_m| > \epsilon \\ 0, & \text{otherwise} \end{cases} \tag{3.3}$$

The main objective of kernel function is to transform input data into high-dimensional feature space. Linear, Polynomial and Gaussian are the most popular kernels. The selection of the kernel ($K$) is based on the underlying problem. We have used linear and Polynomial kernels for time series data in this work.

$$K\left(x, y\right) = \left(x^T \cdot y + c\right) \tag{3.4}$$

$$K\left(x, y\right) = \left(x \cdot y + 1\right)^p \tag{3.5}$$

Deep learning (DL), one of the classes of ANN (Artificial Neural Network) which exploit learning ability using multilayer deep networks and improve the influence of NNs (Neural Network). A standard NN mimics the working of the brain to process information using units called neurons. The neurons are triggered by a peripheral vision sensor and some neurons are triggered by the weighting of the previously active neurons. The neural learning network will have a collection of values for weights between neurons utilizing information flowing through them. Communication between neurons is done using forward direction to process information and generate output for the next layer. The non linear function $f$ which perform this task is given in equation 3.6, where $b$ is the bias and weights of connections defined by $w_i$.

$$f\left(W^t x\right) = f\left(\sum_{i=1}^{n} W_i x_i + b\right) \tag{3.6}$$

The most common activation functions are Sigmoid function, hyperbolic tangent function (*tanh*), and rectified linear function (*ReLU*). Their formulas are as follows:

$$f\left(W^t x\right) = Sigmoid\left(W^t x\right) = \frac{1}{1 + exp\left(-W^t x\right)} \tag{3.7}$$

$$f\left(W^t x\right) = tanh\left(W^t x\right) = \frac{e^{W^t x} - e^{-W^t x}}{e^{W^t x} + e^{-W^t x}} \tag{3.8}$$

$$f\left(W^t x\right) = Relu\left(W^t x\right) = max\left(0, W^t x\right) \tag{3.9}$$

A recurrent neural network (RNN) is considered as a sub class of artificial neural networks where temporal sequence of data are modeled with nodes which form a directed graph.

One of the improvement proposed in RNN to handle problem of vanishing gradient by in incorporating LSTM(Long short-term memory) network architecture Fig. 3.1. The traditional LSTM contains recurrent connections with input gates, forget gates,output gates and connected to output layer. In LSTM computation is performed on input/output training set with activations,using following formulas:

$$i_t = \sigma(W_{ix} x_t + W_{im} m_{t-1} + W_{ic} c_{t-1} + b_i \tag{3.10}$$

$$f_t = \sigma(W_{fx} x_t + W_{mf} m_{t-1} + W_{cf} c_{t-1} + b_f) \tag{3.11}$$

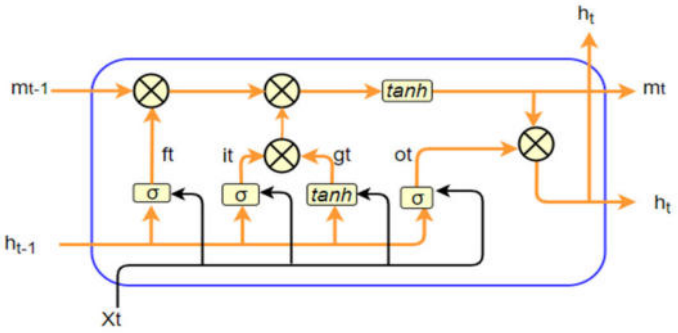


FIG. 3.1. *Architecture of an LSTM unit*

$$c_t = f_t \odot c_{t-1} + i_t \odot g(W_{cx}x_t + W_{cm}m_{t-1}) \tag{3.12}$$

$$o_t = \sigma(W_{ox}x_t + W_{om}m_{t-1} + W_{oc}c_t + b_o) \tag{3.13}$$

$$m_t = o_t \odot h(c_t) \tag{3.14}$$

$$y_t = W_{ym}m_t + b_y \tag{3.15}$$

For the proposed work, we have explored Bidirectional LSTM which is an extension of traditional LSTMs and Bidirectional RNN [32]. With Bidirectional LSTM, the model takes input sequences in a bidirectional way and concatenates interpretations which boost prediction efficiency [36]. This can provide additional context to the network and results in faster and enhanced learning on current problems. The dataset is split in 80 and 20 for training and testing respectively.

Different error measures can be used to evaluate the accuracy of the Predictive models. Some of the most common error measures are the following:

– Mean Absolute Error (MAE):

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{3.16}$$

– Mean Squared Error (MSE):

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{3.17}$$

– Root Mean Squared Error (RMSE):

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \tag{3.18}$$

From the above error measures we obtain training error and real error when applied to the training set and testing set of time series data.

**3.1. Queuing model.** Cloud Computing provides an elastic service to users. To fulfill elastic service with a dynamic workload, it is required to allocate the optimal number of resources (servers). Predictive models discussed earlier provide the estimation of requests that will be required to fulfill SLA contracted with users. As shown in Figure 2, such estimation of requests can be processed by a proposed Queuing model (M/M/c) to plan the resources. The purpose of using the Queuing model is to find a minimum number of servers(c) with system utilization. In Queueing model(M/M/c), parameters such as $c$, $\lambda$ and $\mu$ are considered to find system utilization $\rho$ using following

$$\rho = \frac{\lambda}{c\mu} \tag{3.19}$$

$$c = \frac{\lambda}{\rho\mu} \tag{3.20}$$

where $c$ is number of servers , $\lambda$ is arrival rate of user requests (real and predicted) and $\mu$ is considered as service rate of a server per time unit. In order to serve the users, system utilization $\mu$ should be less than 1. There are other parameters like $W_p$ (queue time) and $Q_w$ (response time) for modeling system performance. In this proposal we have considered time period of one hour to predict average system utilization and then based of proactive prediction model and using Queuing model to find the oscillation of resources to assign for future time period. One hour time period is considered as generalization scenario for user to obtain service from cloud providers. Following framework has been considered to achieve the task of resource management.

In this paper, we have considered a time period of one hour to predict average system utilization and then based on a proposed proactive prediction model. The Queuing model is proposed to find the oscillation of resources to assign for future periods. One hour time period is considered as a generalization scenario for users to obtain service from cloud providers. The following framework has been considered to achieve the task of resource management.

**4. The predictive framework of a hybrid cloud.** This section presents the proposed predictive model that employs an auto scale approach to improve the allocation of resources. The framework is depicted in Figure 4.1. Consider a hybrid cloud where private and public data centers have $M$ machines and hosting $A$ applications. Each application needs a certain amount of different resources. The system receives input from the data center's workload data and performs time series analysis to estimate the future workload and resources required to handle it. An autoscale module analyses such n time instances and forecasts the expected resources as a function of demand. Thus, the objective is to minimize the variance between forecast resources and demand resources to optimize resource allocation. The framework considers the queuing theory to plan estimated resources for the future workload. The execute process signals a resource manager to allocate or deallocate the resources. The Resource manager optimally associates the resources to a private cloud or a public cloud environment as per the availability.

The AutoScaleHybrid Model analyses user requests to forecast the number of resources in a hybrid cloud using a closed Queueing network. The closed Queueing network is considered where constant numbers of users will circulate in the system and are replaced by new users. The AutScaleHybrid algorithm (1) uses a historical load of user requests. It uses Predictor algorithm (2) to estimate the resources using ML techniques and Queueing methods. The algorithm will compare the allocated and forecast result and provide the current status of resources to scale up or down. Here resource managers will allocate the resources to private or public clouds based on decision parameters.

The time complexity of the proposed algorithm is $O(N)$, where $N$ is the number of samples to be taken for analysis. So the system with linear time complexity is scalable and expanded with respect to time.

**5. Experiments and analysis.** The experiment was conducted on a dataset obtained from the university server log. The private cloud was set up to measure the utilization of the system with a web server. The unit of log data considered for the experiment was for an hour. Figure 4 shows the hourly average load of a web server of one month. The following setup had been configured (Table 5.1) for the experiment. The total number of hours is 6072. The data were extrapolated. As per algorithm 2 it is required to choose the lag period to identify

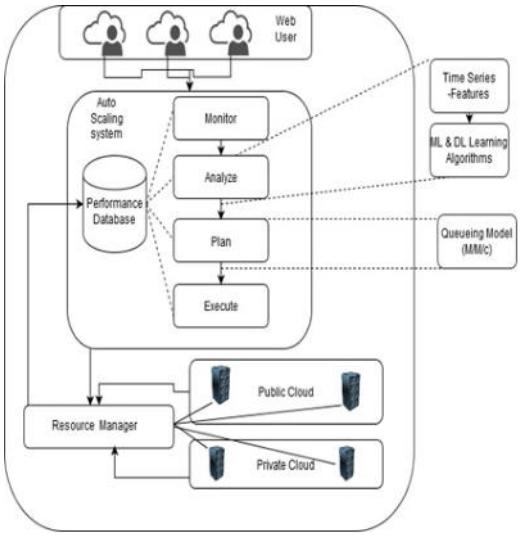


FIG. 4.1. *A Hybrid Cloud system with a predictive framework*

**Algorithm 1** AutoScaleHybrid

1: **Initialization:** History of UserRequests ($R_{req}$) , Monitortime, UserRequest,status ,decision= $NULL$.
2: **repeat** for every monitortime t {
3: status =Predictor ( History of Resources Usage )
4: If status =UP and UserRequest=Private then
*PrivateScaleUp(new VMs)*
5: Else If status =UP and UserRequest=Public then
*PublicScaleUp(new VMs)*
6: Else If status =DOWN and UserRequest=Private then
*PrivateScaleDown(VMs)*
7: Else
PublicScaleDown(VMs)
8: }

the footprint of seasonal data. The R programming tool is used to find autocorrelation of lag in time series data set. As can be observed from figure 5, data shows autocorrelation for different lag intervals. As per the data with lag (L)=24 gives the highest correlation value for seasonal data. So Input training data set can be prepared with the chosen lag value.

In order to apply ML (Machine Learning) based predictive models discussed earlier, it is required to set some basic parameters to obtain reasonable forecasting accuracy. In the SVM regression model, the parameter of loss function and the *"c"* parameter of regularized risk must be adjusted by the user. There are many directions given in the proposal for selection of parameters [34, 33]. Based on the literature, we have chosen the best value to build a good model. In a Bidirectional LSTM model, there are two recurrent layers side-by-side supplying the input data as it is and reverse copy of input data to the second. Here Relu activation function is used for hidden layers. In this work we have used adam optimizer which captures optimal learning rate with gradient. The R programming is configured with keras environment for this work. In this work, we suggest that the Bidirectional LSTM regression model outperforms other predictive methods. Here we have compared LSTM model with other statistical and non-statistical forecasting methods:

- Naive Method : $\hat{y}(t) = y(t-1)$ where $y(t-1)$ is the load of server of previous time interval and $y(t)$ is new estimated load.
- Auto-Regressive Method: Auto-regressive method is based on linear regression with lag interval of 24.

**Algorithm 2** Predictor

**Inputs:** $R_{req}$; $L$.
**Output:Status**
Compute autocorrelation of data trace upto L
$L \leftarrow R_{req}$ /* Determine the lag with significant autocorrelation*/
sample $\leftarrow$ getsample($R_{req}$,L) /*Prepare the input data according to the lag L*/
$R_{fut} \leftarrow$ **Predict**(sample)
$R_{actual} \leftarrow Q_c(R_{req})$
$RR_{fut} \leftarrow Q_c(R_{fut})$
If $RR_{fut} > R_{actual}$ then status=UP
If $RR_{fut} < R_{actual}$ then status=Down
retrun status

TABLE 5.1
*Server configuration*

| | |
|---|---|
| Model | HP DL380 10TH GENRATION |
| CPU | INTEL XEON SILVER 4110 (2 NOS.) |
| RAM | 128 GB (32GB*4) DDR4-2666 MHZ |
| HTTP server | Apache 2.4 |
| Guest OS | Ubuntu Server 14.04 LTS |
| Host OS | Centos 7.0 |

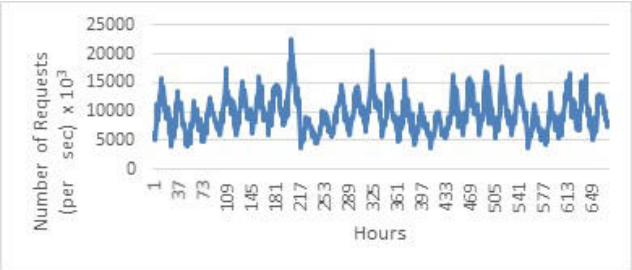


FIG. 5.1. *Historical workload of a web server*

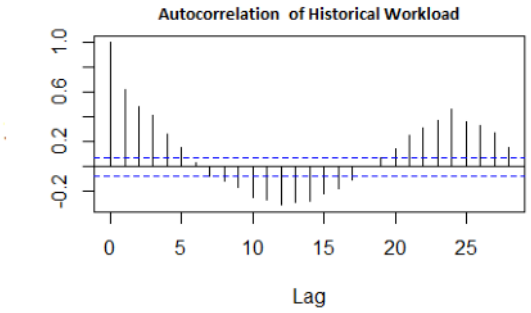


FIG. 5.2. *Autocorrelation of server workload*

TABLE 5.2
*Parameters selection and Performance Measures Accuracy of Predictive Methods*

| Methods | Parameters and Values | MAE | RMSE |
|---|---|---|---|
| Naive | - | 0.135 | 0.167 |
| ARIMA | p=0,q=2,d=2 | 0.146 | 0.174 |
| KNN | k=5 | 0.156 | 0.184 |
| SVR (linear Kernel) | - | 0.125 | 0.154 |
| SVR(Poly. Kernel) | c=1.9,$\epsilon$=0.027, p=2 | 0.113 | 0.138 |
| Bi-LSTM (Proposed) | Batch size=64,learning rate =0.01 , layers=1 | 0.093 | 0.112 |

Here *autoarima* function of R programming environment has been considered. We have performed grid search to choose the parameter values (p=0,q=2,d=2) with AIC criteria.

- K-Nearest Neighbor: This is based on K - Nearest neighbors algorithm which forecast load of server based on similarity measure. Here in this work we have considered five neighbors for similarity measure and k-fold cross-validation to enhance the model.
- Support Vector Regression with Kernel Method: This method estimates the load of server with using equation 3.5 with degree of p=2. Here we have used grid search method to tune the hyper parameters ($c$ and $\epsilon$).
- Bidirectional LSTM Method: Here input layer that specifies length of input data with one feature. The two copies of hidden layer are available with *Relu* as activation function. Table 5.2 shows accuracy of different predictive methods with prediction error obtained by comparing with real values.

As seen from Table 5.2, SVR (Polynomial Kernel) and Bi-LSTM are considered as improved models with respect to prediction errors. The Bi-LSTM method provides better results as compared to other methods. As per the estimated load(requests) obtained from the above methods, we have implemented an M/M/c Queuing model to plan the number of servers that must be allocated with performance criteria. Based on the results of the Queuing model, we can plan the near-optimal resources allocated to the cloud. The resources (servers) which are over-provisioned if near-optimal allocation of resources are less and estimated resource utilization is more.

However, under-provisioned will occur when near-optimal allocation is more than the estimated resource allocation. In this work, we have considered Response Time as one of the performance criteria. The Response Time for user service is imposed by SLA. In the proposed Queuing model, service rate $\mu$ is considered as the throughput of the server (number of requests processed per unit of time). In an ideal scenario, a server having dynamic content will process 200 request/s [35]. The system utilization $\rho$ must be less than 1 to provide efficient service to the user. In this work, the maximum response time which is 5ms is considered as per the SLA. Following the performance, criteria provide the effectiveness of the methods discussed above. Table 5.3 shows the estimation of resources using the methods discussed in Table 5.2.

- The number of allocated resources: The cost of infrastructure will increase when the estimated load is more than the optimal load. In such scenarios, the service rate will be improved. The best model will have an estimated load closer to the optimal load which leads to less number of over provisional resources.
- The number of SLA violations: The more number of SLA violations will occur when the number of allocated resources is less than the optimal. The best estimation model has less SLA violation which will solve the problem of under provision.

**6. Results and Discussion.** We have measured the performance of the proposed model and compared with other models. From the experiment results shown in tables 5.3 and table 5.4, we can see the elasticity of resources as per the estimated load compared to the server's real load. The dataset contains a log of user requests for one month. The model uses time lag data of 24-hour consecutive steps and forecast next day resources. The model's input vector is 3D and divided into training and testing sets for LSTM and 2D for other methods. During the training process of the proposed method, a mini-batch gradient descent method is used. The model optimizes the mean squared error (MSE) loss using adam optimizer and an early stopping mechanism

TABLE 5.3
*Comparison of estimated resources using the Proposed and other Predictive methods with optimal resource allocation derived using Queuing Model.*

| Time (hour) | Optimal Load | Naive | ARIMA | KNN | SVR-L | SVR-P | Bi-LSTM |
|---|---|---|---|---|---|---|---|
| 0:00 | 44 | 44 | 49 | 41 | 42 | 44 | 43 |
| 1:00 | 42 | 37 | 46 | 37 | 39 | 40 | 43 |
| 2:00 | 34 | 42 | 42 | 26 | 29 | 27 | 29 |
| 3:00 | 45 | 34 | 41 | 46 | 47 | 46 | 44 |
| 4:00 | 52 | 45 | 42 | 55 | 54 | 55 | 54 |
| 5:00 | 41 | 52 | 43 | 32 | 32 | 32 | 33 |
| 6:00 | 58 | 41 | 49 | 66 | 68 | 66 | 65 |
| 7:00 | 55 | 58 | 51 | 52 | 53 | 50 | 53 |
| 8:00 | 70 | 55 | 55 | 79 | 80 | 77 | 78 |
| 9:00 | 78 | 70 | 65 | 80 | 80 | 80 | 76 |
| 10:00 | 87 | 78 | 72 | 91 | 93 | 91 | 92 |
| 11:00 | 74 | 87 | 78 | 64 | 64 | 66 | 65 |
| 12:00 | 81 | 74 | 80 | 84 | 84 | 86 | 85 |
| 13:00 | 89 | 81 | 82 | 99 | 101 | 98 | 96 |
| 14:00 | 74 | 89 | 80 | 69 | 70 | 71 | 68 |
| 15:00 | 73 | 74 | 79 | 68 | 69 | 72 | 70 |
| 16:00 | 76 | 73 | 78 | 74 | 75 | 79 | 82 |
| 17:00 | 51 | 76 | 70 | 34 | 35 | 37 | 38 |
| 18:00 | 68 | 51 | 68 | 71 | 74 | 74 | 73 |
| 19:00 | 61 | 68 | 64 | 60 | 60 | 60 | 66 |
| 20:00 | 62 | 61 | 60 | 71 | 72 | 67 | 69 |
| 21:00 | 51 | 62 | 61 | 46 | 48 | 45 | 45 |
| 22:00 | 49 | 51 | 55 | 45 | 46 | 45 | 49 |
| 23:00 | 48 | 49 | 52 | 57 | 57 | 51 | 48 |

TABLE 5.4
*Prediction of Resource status using Predictive Models with Queuing system*

| Resource status Estimation Method | Naive | ARIMA | KNN | SVR-L | SVR-P | Bi-LSTM |
|---|---|---|---|---|---|---|
| Under Provision | 108 | 80 | 77 | 63 | 60 | 56 |
| Over Provision | 97 | 79 | 61 | 72 | 56 | 57 |
| % SLA Violation | 15 | 20 | 12 | 15 | 12 | 10 |

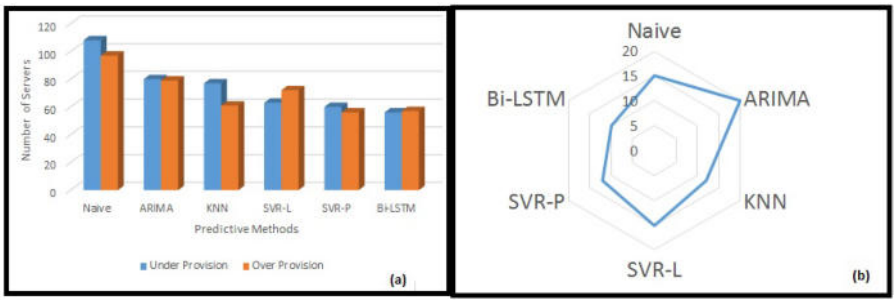


FIG. 5.3. *a. Forecasted resources (servers) suggested by Predictive Methods. b. Forecasted SLA violation (Percentage) of Predictive Methods*

is used to avoid over-fitting. The baseline model (Naive) is considered and compared with the proposed model. We compared the performance of the proposed model with other forecasting models with MAE and RMSE. As per Table 5.2, which suggests the proposed model performs well compared to other models. Also the results shown in Table 5.4 suggest the proposed model having less number of oscillations (5%)in predicted resources with respect to optimal load (Figure 5.3). Hence it suggests less number of SLA violations with other models, which imply that the model performs well in the future horizon.

**7. Conclusions.** In this paper, a Bi-directional LSTM based approach for resource allocation in a hybrid cloud environment is proposed. Further, the predictive methods based on ML and DL are evaluated using time series data. The Autoscale Hybrid model optimizes the SLA violation which leads to an effective response time of the service with optimal resource demand forecast. The auto scale module has very less time to execute and predict the resources. In the cloud environment, instances are forked which takes less than 10 minutes. The AutoscaleHybridModel framework considered data on an hourly basis in which instances can be configured in less time. The results show that the Bi-directional LSTM model provides closer to the real resource demand than the other models. In the future, the model with more optimization parameters can be incorporated to improve cloud resources usage predictions.

REFERENCES


[1] Masdari, Mohammad, et al., "Towards workflow scheduling in cloud computing: a comprehensive analysis." Journal of Network and Computer Applications 66 (2016): 64-82.

[2] Toosi, Adel Nadjaran, Richard O. Sinnott, and Rajkumar Buyya, "Resource provisioning for data-intensive applications with deadline constraints on hybrid clouds using Aneka." Future Generation Computer Systems 79 (2018): 765-775.

[3] Bresciani, Stefano, Alberto Ferraris, and Manlio Del Giudice, "The management of organizational ambidexterity through alliances in a new context of analysis: Internet of Things (IoT) smart city projects." Technological Forecasting and Social Change 136 (2018): 331-338.

[4] Xu, Xiangqiang, and Xinghui Zhao, "A framework for privacy-aware computing on hybrid clouds with mixed-sensitivity data." 2015 IEEE 17th International Conference on High Performance Computing and Communications, 2015 IEEE 7th International Symposium on Cyberspace Safety and Security, and 2015 IEEE 12th International Conference on Embedded Software and Systems. IEEE, 2015.

[5] Gandhi, Anshul, et al, "Optimal power allocation in server farms." ACM SIGMETRICS Performance Evaluation Review 37.1 (2009): 157-168.

[6] Ye K, Huang D, Jiang X, Chen H, Wu S, Virtual machine based energy-efficient data center architecture for cloud computing: A performance perspective In: Green Computing and Communications (GreenCom) 2010 IEEE/ACM Int'l Conference on Int'l Conference on Cyber, Physical and Social Computing (CPSCom), 171–178.

[7] Messias, Valter Rogério, et al., "Combining time series prediction models using genetic algorithm to autoscaling web applications hosted in the cloud infrastructure." Neural Computing and Applications 27.8 (2016): 2383-2406.

[8] Calheiros, Rodrigo N., et al., "Workload prediction using ARIMA model and its impact on cloud applications' QoS." IEEE Transactions on Cloud Computing 3.4 (2014): 449-458.

[9] Islam, Sadeka, et al., "Empirical prediction models for adaptive resource provisioning in the cloud." Future Generation Computer Systems 28.1 (2012): 155-162.

[10] Kumar, Jitendra, and Ashutosh Kumar Singh., "Workload prediction in cloud using artificial neural network and adaptive differential evolution." Future Generation Computer Systems 81 (2018): 41-52.

[11] Amiri, Maryam, and Leyli Mohammad-Khanli., "Survey on prediction models of applications for resources provisioning in cloud." Journal of Network and Computer Applications 82 (2017): 93-113.

[12] Manvi, Sunilkumar S., and Gopal Krishna Shyam., "Resource management for Infrastructure as a Service (IaaS) in cloud computing: A survey." Journal of network and computer applications 41 (2014): 424-440.

[13] Weingärtner, Rafael, Gabriel Beims Bräscher, and Carlos Becker Westphall., "Cloud resource management: A survey on forecasting and profiling models." Journal of Network and Computer Applications 47 (2015): 99-106.

[14] Panneerselvam, J., Liu, L., Antonopoulos, N., Bo, Y., Workload analysis for the scope of user demand prediction model evaluations in cloud environments. In: Proceedings of the 2014 IEEE ACM 7th International Conference on Utility and Cloud Computing, pp. 883–889. IEEE Computer Society (2014)

[15] Messias, Valter Rogério, et al., "Combining time series prediction models using genetic algorithm to autoscaling web applications hosted in the cloud infrastructure." Neural Computing and Applications 27.8 (2016): 2383-2406.

[16] Roy, Nilabja, Abhishek Dubey, and Aniruddha Gokhale, "Efficient autoscaling in the cloud using predictive models for workload forecasting." 2011 IEEE 4th International Conference on Cloud Computing. IEEE, 2011.

[17] Moren Vozmediano, Rafael, Ruben S. Montero, and Ignacio M. Llorente, "Elastic management of web server clusters on distributed virtual infrastructures." Concurrency and Computation: Practice and Experience 23.13 (2011): 1474-1490.

[18] Zhang, H., Jiang, G., Yoshihira, K., Chen, H., Proactive workload management in hybrid cloud computing. IEEE Trans. Netw. Serv. Manag. 11(1), 90–100 (2014)

[19] Hasan, Masum Z., et al., "Integrated and autonomic cloud resource scaling." 2012 IEEE network operations and management symposium. IEEE, 2012.
[20] Mason, Karl, et al., "Predicting host CPU utilization in the cloud using evolutionary neural networks." Future Generation Computer Systems 86 (2018): 162-173.
[21] Barrett, Enda, Enda Howley, and Jim Duggan, "Applying reinforcement learning towards automating resource allocation and application scalability in the cloud." Concurrency and Computation: Practice and Experience 25.12 (2013): 1656-1674.
[22] Tesauro, Gerald, et al., "A hybrid reinforcement learning approach to autonomic resource allocation." 2006 IEEE International Conference on Autonomic Computing. IEEE, 2006.
[23] Liu, Ning, et al., "A hierarchical framework of cloud resource allocation and power management using deep reinforcement learning." 2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS). IEEE, 2017.
[24] Kumar, Jitendra, Rimsha Goomer, and Ashutosh Kumar Singh, "Long short term memory recurrent neural network (lstm-rnn) based workload forecasting model for cloud datacenters." Procedia Computer Science 125 (2018): 676-682.
[25] Zhang, Qingchen, et al., "An efficient deep learning model to predict cloud workload for industry informatics." IEEE transactions on industrial informatics 14.7 (2018): 3170-3178.
[26] Kaur, Pankaj Deep, and Inderveer Chana, "A resource elasticity framework for QoS-aware execution of cloud applications." Future Generation Computer Systems 37 (2014): 14-25.
[27] Hani, Ahmad Fadzil M., Irving Vitra Paputungan, and M. Fadzil Hassan, "Support vector regression for service level agreement violation prediction." 2013 International Conference on Computer, Control, Informatics and Its Applications (IC3INA). IEEE, 2013.
[28] Jiang, Yexi, et al., "Asap: A self-adaptive prediction system for instant cloud resource demand provisioning." 2011 IEEE 11th International Conference on Data Mining. IEEE, 2011.
[29] Allende H, Moraga C, Salas R, (2002) Artificial neural networks in time series forecasting: a comparative analysis. Kybernetika 38(6):685–707.
[30] Wang Y, Huang M, Zhao L, et al., (2016) Attention-based lstm for aspect-level sentiment classification In: Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, 606–615.
[31] Cherkassky, Vladimir, and Yunqian Ma., "Practical selection of SVM parameters and noise estimation for SVM regression." Neural networks 17.1 (2004): 113-126.
[32] Schuster, Mike, and Kuldip K. Paliwal., "Bidirectional recurrent neural networks." IEEE transactions on Signal Processing 45.11 (1997): 2673-2681.
[33] Boardman M, Trappenberg T., (2006) A heuristic for free parameter optimization with support vector machines In: The 2006 IEEE International Joint Conference on Neural Network Proceedings, 610–617.
[34] Cherkassky V, Ma Y, (2004) Practical selection of svm parameters and noise estimation for svm regression. Neural Netw. 17(1):113–126
[35] Iqbal W, Dailey M, Carrera D, Janecek P, (2011) Adaptive resource provisioning for read intensive multi-tier applications in the cloud. Future Gener. Comput. Syst. 27(6):871–879.

*Edited by:* Dana Petcu
*Received:* Sep 11, 2020
*Accepted:* Dec 13, 2020