Workload Analysis of Cloud Resources using Time Series and Machine Learning Prediction

Sivasankari Bhagavathiperumal
Centre for Artifical Intelligence
School of Computer Science
University of Technology Sydney
Australia
sivasankari.bhagavathiperumal@student.uts.edu.au

Madhu Goyal
Centre for Artifical Intelligence
School of Computer Science
University of Technology Sydney
Australia
madhu.goyal-2@uts.edu.au

Abstract—**Most of the businesses now-a-days have started using cloud platforms to host their software applications. A Cloud platform is shared resource that provides various services like software as a service (SAAS), infrastructure as a service (IAAS) or anything as a service (XAAS) that is required to develop and deploy any business application. These cloud services are provided as virtual machines (VM) that can handle the end user's requirements. The cloud providers must ensure efficient resource handling mechanisms for different time intervals to avoid wastage of resources. Auto-scaling mechanisms would take care of using these resources appropriately along with providing an excellent quality of service. Auto-scaling supports the cloud service providers achieve the goal of supplying the required resources automatically. It use methods that will calculate the number of requests and decides the resources to release based on workload. The workload consists of some quantity of application program running on the machine and usually some number of users connected to and communicating with the computer's applications. The researchers have used various approaches to perform autoscaling which is a process to predict the workload that is required to handle the end users request and provide required resources as Virtual Machines (VM) disruptively. Along with providing uninterrupted service, the businesses also only pay for the service they use, thus increasing the popularity of Cloud computing. Based on the workload identified the resources are provisioned. The resource provisioning techniques is a model used for enabling convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, applications, and services) required resources are released. In this regard, the aim of this paper is to develop a framework to predict the workload using deep learning which would be able to handle provisioning of cloud resources dynamically. This framework would handle the user request efficiently and allocate the required virtual machines. As a result, an efficient dynamic method of provisioning of cloud services would be implemented supporting both the cloud providers and users.**

Keywords— Auto-scaling, Time Series, Machine Learning, Deep learning, Virtual machine, Cloud server, Resource Provisioning, Load predictor

I. INTRODUCTION

Cloud Computing is an influenced technology that provides the distributed infrastructure that include environments to host vendor applications, network resources, build and deploy applications. Many businesses prefer to use these cloud services which are supplied as Virtual machines (VM) by the cloud service providers (CSP). This service is provided by large external datacenters and accessed by the business through the internet [1]. This paradigm removes the expense of setting up the infrastructure with reducing the cost by letting the users pay for the resources that is used without affecting the performance [2]. On the other hand, these data centers face a worrying challenge of uncontrollable usage of electricity. To protect our environment, it is our responsibility to ensure the energy consumption is reduced. It is projected that by 2020, these data centers would be utilizing 140 billion kilowatts-hour of electricity which would cost around 13 billion US dollars annually [3]. To ensure the services are uninterrupted, the datacenters are forced to run their servers all the times irrespective of its needed or not. To deal with providing continuous resources auto-scaling is implemented by service providers. It is an automatic approach that predicts the future requirements and allocates the required resources dynamically, seamlessly when the demand is high reducing the power consumption [4]. The Cloud servers offer horizontal scaling like connecting more servers at the same time to do the same work increasing the speed or availability of the logical units and vertical scaling which is the ability to raise the power of the server like increasing the RAM of the computer to improve the speed and performance of the computer [5]. The automated solution to this horizontal and vertical scaling would benefit both the cloud providers and the users of this cloud services concerning with utilization of resources wisely and cost effectively along with increasing the performance [5]. However, identifying the required resources would be challenging as the demand fluctuates from time to time. Therefore, it is important to build the frequently occurring events in the system so that the cloud systems predict the requirements of the users.

There are many approaches proposed with related to autoscaling based on CPU load or artificial neural networks, linear regression, ARIMA models [6] . The popular service providers like Amazon implement autoscaling using user metrics, again it is difficult to tune and optimize [7]. The auto scaling mechanism starts with receiving the end users request to the application through a device with the internet [8]. The application then forwards this request to the virtual machines which should have the auto scaling mechanism deployed in it. Based on the load request received the number of VM to be used is identified and those VM's starts to run to accommodate the end users request [4]. This ensures that there is no wastage of resources along with serving the customers.

With the growing usage of technology [9] and data, using a linear model to implement autoscaling would not be enough to meet the demands. With number of factors to consider in determining the resource required, there is a need for a model that can predict the workload based on training the network. Deep learning is a part of machine learning which ensures the calculations are made considering various factors that could affect the business applications [10]. The aim of this paper is to use deep learning algorithm to predict the workload with which a framework would be developed that would handle the provisioning of cloud resources dynamically. This framework would process the user request efficiently and allocate the required virtual machines. As a result, an efficient dynamic method of provisioning of cloud services would be implemented supporting both the cloud providers and users.

II. RELATED WORK

Edge computing which is aimed at providing the ability to process the data within the edge of the network instead of processing from cloud servers are getting popular currently[11]. In IoT related applications it becomes crucial to distribute the resources. In another paper deep things were proposed which aims at minimizing memory footprint especially when getting exposed to parallelism [12]. The authors focused on partitioning and distribution at the early stage using convolutional neural network interference. They proposed FTP method firstly and then they developed a distributed work stealing runtime system [12].

A linear regression model which will predict the work load of the services used in the cloud was also proposed[13] in which the prediction of the workload was based on the request number at time interval. Though there are a number of techniques exist, the aim of the author was to identify a method which can adjust its model quickly according to the variation trend of workloads [13]. They observed that the workload trend is linear in a relatively short period of time and used linear regression to solve the problem [13].

An approach implemented by Amazon that aimed at selling their unused capacities based on auction like mechanism which were called as spot instances was suggested[14]. These spot instances were suitable mainly for fault-tolerance flexible web applications. The cost of this instance was low compared with the cost that is in demand. Moreover, this approach can be used for applications that can be interrupted[14] only. Applications like background processing, batch jobs can utilise this approach compared to the critical applications. On the other hand, the spot instances also take more time to boot when compared to the on-demand instances. The authors suggested a heterogeneous approach where a mix of both spot and on-demand instances can be used to meet the end users demand[14].

With increasing usage of cloud computing, there are scenarios where cloud service providers lose their customers for various reasons. The authors identified a data driven iterative churn prediction framework[15]. They implemented deep learning approach and proposed Nascency, Intermediate and Latest (NIL) analysis model. The NIL model proposed that identifying customer retention would be profitable and so analysed the data of customers who would stop using their services. They calculated three variables NIL where N is the activities in the initial month of plan, I is activities in middle month of plan and L is latest month of the plan. With the challenge where the customer activities are unpredictable deep convolutional network was used for investigation. Finally, they clustered the customers who paid and the business team engaged them accordingly[15].

Researchers [6] suggested that ARIMA is based on statistical analysis and hence designed a prediction model to capture the workload variation patterns. Workload was classified according to the basic resource used in terms of CPU, RAM memory and service request[6]. The prediction was based on mean absolute percentage error(MAPE)[6]. For a distributed and complex system it is important to predict the load based on high level features than working on with one particular machines[6]. So, a DBN approach was proposed, which composed of logistic regression layer[6] to monitor the CPU utilization. The results showed a prediction accuracy of more than 2.5% and hence was supported for the cloud system in terms of CPU[6].

In another paper deep learning mechanism was used for multimedia analysis and in turn scaling the cloud service[16]. The paper mainly discussed the technical details of image analysis. This analysis would scale horizontally based on user's request. The system able to provide high accuracy, handled request along with scaling GPU and CPU [16]. The authors proposed a platform and framework. They assigned a unique key for each image submission. Once the image is uploaded, the decoding of message received from the customer is authenticated and served. The authors came up with model which is a combination of Fast RCNN and VGG neural networks[16].

Deep learning is been proposed mainly focusing on energy efficiency along with providing the cloud resources in data centers [3]. According to a Deep reinforcement learning (DRL) model, a cloud framework is proposed which aims at reducing the energy used to provision the resources which in turn leads to reducing the cost to the business users [3]. The model considers two aspects one where the user request is accepted followed by queuing and the other aspect where energy is minimized [3]. DRL cloud consist of user workload, cloud platform, energy consumption and realistic model working together performing and reducing the energy, cost for both the cloud providers and end users[3].

With more business moving towards cloud, the energy usage of the datacenters eventually increased. The datacenters are forced to increase the power of CPU to enable the better service to the end users[17]. So they proposed a model based on recurrent neural network (RNN) of deep learning and named it LSTM (Long Short-term memory)[17]. The model works on two phases where in the first phase the system uses the historical time series algorithm to predict the load and then using the results the values are further enhanced to improve the earlier results making it more precise. The trend on workload was classified to be static, periodic, unpredictable, changing continuously[17]. Based on the trends of the load the CPU was released. They experimented the results and identified the request to CPU was not static. The authors introduced their model which works on RNN mechanism where the information is cycled through loops and decision is made after considering the current and previous inputs [17].

Another model based on CNN [12] was proposed where a cloud-based image analysis platform, was implemented. A unique private key is assigned to each customer for image submission. Once they submit an image, it will be uploaded and stored in the cloud server[16]. When the submission is acknowledged, Redis will decide which algorithm to be called, and which queue it needs to submit the messages. Models get messages from the corresponding queues, perform analysis, and push the results to Redis. Results may go through certain post-processing if necessary, and then go back to the customers via RabbitMQ[16]. Once the user submits an image, two fields will be contained in the coming message: user private key, and the images or their URLs. Users are distinguished as per their respective private keys. Redis decodes submissions and determines which algorithm needs to be executed. It submits the message to corresponding queues[16]. Algorithms pulls messages from corresponding queues and analyses the relevant images. Once all the analysis is done, it pushes messages back to RabbitMQ, fetches more messages and continues the routine[16]. Redis also keeps a counter of the number of algorithms in the message. Every time an algorithm finishes the analysis, Redis will deduct one from the counter. When the counter comes to the last, output messages will be pushed to RabbitMQ[16]. In this paper, we would be predicting the values based on forward propagation.

III. Proposed Framework for Resource Provisioning

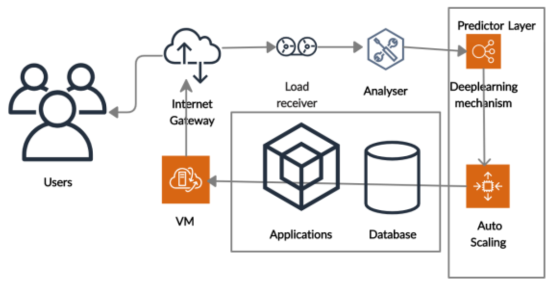


Fig. 1. Framework for dynamic provisioning of Cloud resources

The above figure (Fig.1.) represents the framework to implement autoscaling based on workload prediction using deep learning mechanism. The framework that consists of load-receiver, load analyser that will take care of load entering the system and analysing the load along with computing the calculation for response received with bytes transferred to serve the request. The load receiver and load analyser layers in the framework will receive and analyse the load entering the business application. The load predictor layer access the weblog data and will implement the forward and back propagation. The predictor layer will be loaded with a training data. Using this training data, through forward propagation the expected load to the server would be calculated. To ensure the predication is more accurate, the results computed through forward propagation is taken to compute more precise value using back propagation. Now the deep learning mechanism control will evaluate the workload and send the details to the load controller. The final process of provisioning of the VM happens in this layer. Controller will control the VM's and release or cease the resources based on the report it received from the predictor.

These layers will send the data to the load predictor which would use that information in deep learning mechanism layer to work on the popular forward and back propagation. This layer sends the related graphs and notes to release the resources based on the request, server response and the file are transferred. Load predictor layer will have the control over VM's and release or cease the resources based on the report it received from the analyser.

In the above framework (Fig.1.), the load receiver would detect the load entering the cloud server. After receiving the monitored load requests, the data would send to the load analyser. The request is analysed and then controller handles the release of the resources in the form of Virtual Machines. Based on the forward algorithm in deep learning, the predictor layer calculates the series and predicts the resources required on regular basis. With prediction prevailing with the controller, the dynamic provisioning of resources would happen consequently. Without human intervention, the release of resources would function in maintaining the quality of service to the end users.

The cloud providers provision resources through virtual machines ensuring the quality of service in accordance with the Service Level Agreements (SLA) [6]. For example, consider a situation where X number of machines are serving N number of customers at the peak time of Stock exchange. Suddenly the unanticipated number of customers uses the same application increasing the number of nodes. This is when the performance of the application becomes down and violates the SLA. The end users access the business application to perform their required task through internet. The request for accessing the applications varies at different time intervals and thus based on the number of end users request the VM's should be released. At certain times there might be enormous number of users trying to reach the application layer and there might be less or no requests also. Fig.2 shows the workload where users are sending request to NASA servers at a certain time on a day.

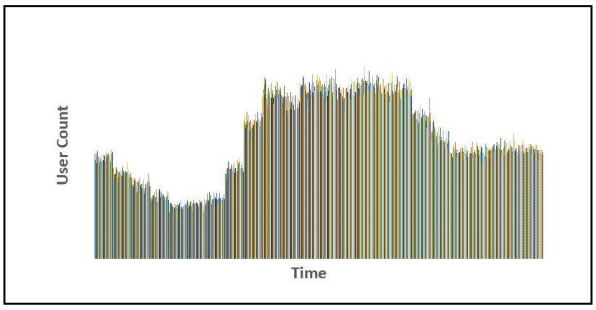


Fig.2. Plot showing Number of Users and Time

This is the layer where the business applications are hosted. This layer may consist of numerous services, applications and databases. The end users would send the request to this layer. To maintain a quality of service, this layer should be active always irrespective of the demands.

The prediction layer is the main component of this paper where the scaling process happens. The predictor layer consists of life cycle that detects the signal received from the end user measuring the request, analysing the request and finally predicting the required VM's. This data is sent to the load controller present in the VM layer.

This process receives the request from the end user to the application along with detecting the number of virtual machines released to the handle the demands. The load will fluctuate time to time, and this gets recorded by the load detector actively. The results will be shared with the load measurer.

The load analyser task is to analyse the load that is measured. This will help the controller control the release or ceasing of the VM. The analyser will take the responsibility of analysing both resource release and time. The end user request is analysed and validated by the load analyser. After care analysis, the data is supplied to the load controller. The load analyser user the auto regressive or linear regression method to analyse the load so that the data of analysed load helps the load controller predict the future loads.

The VM layer is where the virtual machines are ready to auto-scale based on the requirement. This layer consists of the n number of virtual machines and the load controller. The load controller receives data from the predictor layer and ensures the required VM's are released to the meet the end users demand.

IV. DATA SOURCES DETAILS

We used the workload characterization study that is composed of the access logs collected from NASA servers. Fig. 2 is the representation of web users request to application server on the month of July. From the workload, it is very clear that the load fluctuates from time to time and day on day. We have used the web log data which is the records of activity information when a web user submitted the request to a Web Server. The main source of raw data is the web access log which we shall The weblog data consisted of IP address, URL, response, bytes transacted along with date and time.

The NASA workload data consisted of the following information:

- Host which is remote IP address or domain name-An IP address is a 32-bit host address defined by the Internet Protocol;
- a log name is used to determine a unique Internet address for any host on the internet which is unused and always –
- Date and time.
- Modes of request: GET, POST or HEAD method of CGI (Common Gateway Interface) with HTTP status code returned to the client, e.g., 200 is "ok" and 404 are "not found".
- Bytes: The content-length of the document transferred.

The NASA workload represents genuine load changes over time that can be used to compute results and conclusions more authentic and reliable to be used in real environments. The workload comprises 100960 user requests sent to NASA Web servers. The graph in above figure (Fig.2) shows an extreme fluctuation in this workload. Any business applications in real time would have a similar pattern. To summarise the noted results of usage of server, most load to the web application was less during the early hours and increased steadily during the mid-day along with slope decreasing towards the business closing hours.

V. METHODS

The main aim of our research is to find the workload, wherein we identified the number of requests to the business applications. The time series data can be classified into four types like trend, seasonal, cyclicity and residuals and every timeseries can be a mix of all or few of the types [18]. Predicting the arrival of request can also help in improving the performance of the website along with getting prepared ahead. As seen in Fig.1, the request is varying at certain times on certain days and thus can understand that the workload is mix of trend and seasonal.

A. Naive approach

Firstly, we used Naive approach to predict the workload. It is a very simplest forecasting method and uses the most recent observation. It can be implemented in a namesake function which is the best that can be done for many time series data. Even though it is not a good forecasting method, it can be used as a benchmark for other forecasting methods. To summarise, naive method is forecasting technique which assumes that the next expected point is equal to the last observed point and can be obtained by the formula below:

$$Y_t = Y_{t-1} \tag{1}$$

where Y_t is forecast at time t and Y_{t-1} is actual data at time t. Below Fig.4 is the implementation of naive method for the NASA weblog data. This method produces a root mean square error of around 45% and so we can infer that Naive method isn't suited for datasets with high variability unlike a stable dataset.

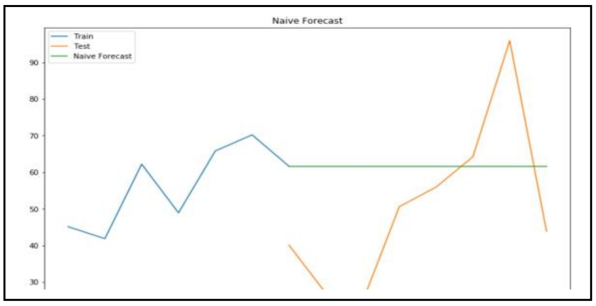


Fig. 4. Forecast using Naive approach

B. *Exponential smoothing approach*

Then we used the exponential smoothing approach which is used for univariate data that can be extended to support data with a systematic trend or seasonal component in time series predictions. It uses the forecasting methods that are similar to prediction that is a weighted sum of past observations, but the model explicitly uses an exponentially decreasing weight for past observations. It uses the following formula:

$$a_t = \alpha(X_t - F_{t-s}) + (1 - \alpha)(a_{t-1} + b_{t-1}) \qquad (2)$$

$$b_t = \beta(a_t - a_{t-1}) + (1 - \beta)b_{t-1} \qquad (3)$$

$$Ft = (X_t - a_t) + (-)F_{t-s} \qquad (4)$$

Here α, β, and γ are smoothing constants which are between zero and one. Again, at gives the y-intercept (or level) at time t, while b_t is the slope at time t and S is the given value of period. As we can see, 1−α is multiplied by the previous expected value which makes the expression recursive. Below fig.5 is the prediction of workload based on exponential smoothing approach.

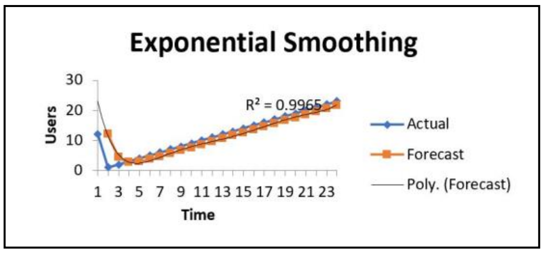


Fig.5. Workload prediction using exponential smoothing

C. *ARIMA method*

Another common and popular time series model is ARIMA which is Autoregressive Integrated Moving average. While exponential smoothing models were based on a description of trend and seasonality in the data, ARIMA models aim to describe the correlations in the data with each other. It is a stochastic modeling approach that can be used to calculate the possible future value lying between two specified limits. To build an ARIMA model, we need to understand the time series. Followed by that we have to identify p, d, q where p is the number of autoregressive terms, d is the number of nonseasonal differences needed for stationarity, and q is the number of lagged forecast errors in the prediction equation. In terms of y, the general forecasting equation is where ŷt is predicted load (5):

$$\hat{y}_t = \mu + \phi_1 y_{t-1} + \ldots + \phi_p y_{t-p} - \theta_1 e_{t-1} - \ldots - \theta_q e_{t-q} \qquad (5)$$

To identify the appropriate ARIMA model for Y, we first begin by determining the order of differencing (d) and remove the gross features of seasonality.

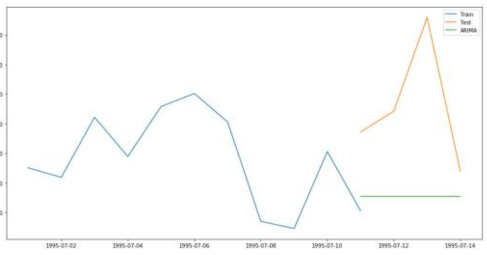


Fig.6. Workload forecast using ARIMA

Fig.6 shows the prediction of workload based on ARIMA method. The ARIMA forecasting for a time series is nothing but like a linear regression equation. The predictions mainly depend on the parameters (p,d,q) of the ARIMA model. The number of AR (Auto-Regressive) terms (p) and AR terms are just lagging of dependent variable. For example, if p is 10, the predictors for x(t) will be x(t-1)….x(t-10). The number of MA (Moving Average) terms (q) where MA terms are lagged forecast errors in prediction equation. For instance if q is 10, the predictors for x(t) will be e(t-1)….e(t-10) where e(i) is the difference between the moving average at ith instant and actual value. Finally, the number of Differences (d) are the number of nonseasonal differences, i.e. in this case we took the first order difference. So, either we can pass that variable and put d=0 or pass the original variable and put d=1. Both will generate same results. The fig.6 is generated by passing the value d=0.An important point with ARIMA is how to determine the value of 'p' and 'q'.

Another researchers [13] compared predicted workloads with actual workloads, and a set of alternative workload prediction algorithms using the old. They compared the load with a second order autoregressive moving average method(6) filter (ARMA) with the below equation

$$Y_{t+1} = \beta * Y_t + \gamma * Y_{t-1} + (1 - (\beta + \gamma)) * Y_t-2 \qquad (6)$$

The value for the variables β and γ were given by the values 0.8 and 0.15, respectively. Mean is the mean workload over the workloads in the window and Max is the maximum workload over the workloads in the window[13]. Fig.7 is the formula to calculate root mean square error (RMSE) with which the prediction results can be tested for errors. The summary of forecasting and the respective root mean square error is calculated and represented in the Table.1

$$RMSE=\sqrt{\frac{\sum_{i=1}^{N}(Predicted_i - Actual_i)^2}{N}} \qquad (6)$$

RMSE is the standard deviation of prediction that measure the regression line data points. It can tells us how the data is fitting the in to produce best results that can be used in climatology, forecasting, and regression analysis to verify experimental results.

TABLE 1

Methods	RMSE
Naive Method	68.01
Exponential Smoothing	28.06
ARIMA	25.08
Deep Learning (Forward propagation)	18.98

D. *Deep Learning*

Deep learning is a machine learning technique that computes calculation to assist computers what to do. A computer model understands to perform various tasks directly from images, text, structured or any unstructured data. Deep learning models are trained by using a large set of data that contain many layers. Deep in deep learning refers to the number of hidden layers in the neural network. Deep learning networks can have hidden layers up to 150. With workload variations, it is important to learn the features and relativeness [6]. Since deep learning is strongly recommended to learn various patterns, it can also be used to learn the variations in workload data.

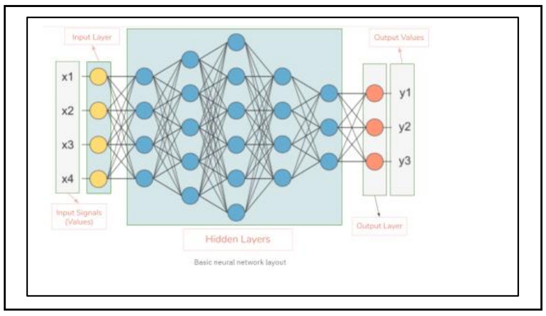


Fig.7. Basic layout of neural network

Fig.7 is a basic layout of the neural network. The layout consists of three basic layers like input layer, hidden layer and output layer. Input layer is the first layer which can take the input values and pass it to the next layer called hidden layer. The output to the hidden layer is provided based on calculation which is sum of the product of input value and weight. A weight represents the strength of connection between units, which means if the weight of node 1 is greater than node 2 then node 1 has greater influence over node 2 and vice versa. Basically, the weight decides the value of input. If the weight is zero, then the input has no effect and if the weight is negative then increasing the input will decrease the output on hidden layer has neurons (nodes) which apply different conversions to the input data. All the nodes in a hidden layer are connected to each node in the next layer which is output layer where we can predict the desired number of values in a certain range.

In forward propagation, the input values are served to the neural network's first layer without any operation. Then the second layer takes the value and apply multiplication, adding and perform activation operation. The output of second layer is fed to the next layer which again follows similar operation to produce the result. Then perform the backpropagation with the output value which is the predicted value through forward propagation. To calculate error, we compare the predicted value with the actual output value. We use a loss function (mentioned below) to calculate the error value. Then we calculate the derivative of the error value with respect to every weight in the neural network. Back-Propagation uses chain rule of Differential Calculus. In chain rule first we calculate the derivatives of error value with respect to the weight values of the last layer. We call these derivatives, gradients and use these gradient values to calculate the gradients of the second last layer. We repeat this process until we get gradients for every weight in our neural network. Then we subtract this gradient value from the weight value to reduce the error value.

The forward propagation can be performed by first initialising the input layer with certain weight typically between -1 and 1. Then we can iterate through the input-layer to compute the values of hidden layer. This type of architecture in multilayer neural network is referred as feed-forward networks as they move forward to feed the values for the next layer [19]. Below Fig.8, is the representation of forward propagation.

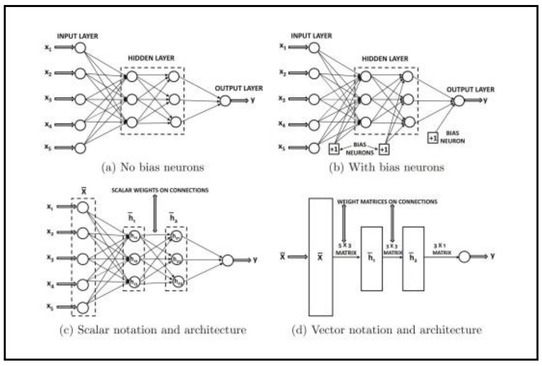
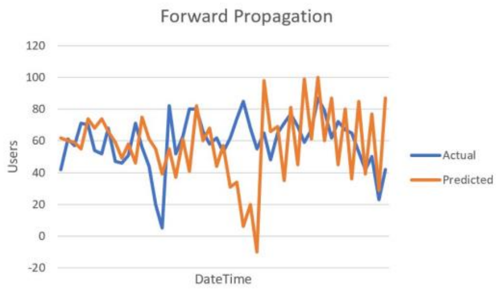


Fig.8.Basic architecture of feed forward network[19]

With results varying in different methods, we built a deep learning model to predict the workload. We determined the input layer with time and number of requests. To make it simple, we assigned a weight between -1 to 1 for the input layer and calculated the values of hidden layer and assigned the output in an array. With the values of hidden layer, we then calculated the predicted values again with assigning similar weights. Fig.9 represents the workload prediction by training the dataset with forward propagation.

Fig.9. Actual and predicted values using forward propagation



With the forward propagation, the results were improved reducing the RMSE. These results can further be tuned to have more accuracy using the back propagation.

VII. DISCUSSION

We presented a naive method in which the dataset was divided into train and test randomly with 75% and 25% respectively and calculated the predicted load which assumed the previous values. This method produced a result which was used as benchmark for the later analysis. Then we used the same dataset with partition as train and test to calculate the predicted load using exponential smoothing. The exponential smoothing method used the smoothing constant and calculated the predicted load using the polynomial expression of 0.6. The results from the exponential smoothing method improved from the naïve approach. The value of smoothing constants determines how fast the weights of timeseries decays and values are chosen subjectively or objectively. When the values of a smoothing constant near one put almost all weight on the most recent observations and on the other hand values of a smoothing constant near zero allow the distant past observations to have a large influence. To choose the values subjectively, we can use our experience.

Then popular ARIMA method for time series was used to calculate the load. With ARIMA, the results improved dramatically. The dataset was used to implement deep learning algorithm and the setup of data is based on the trial and error procedure to find a good number of neurons, m, in hidden layer (HL). This can be implemented in the predictor layer of the framework suggested in Fig.1 to serve the requests from the end users. Although, there are possibilities of some deviations from the actual and predicted workload, the deep model shows the ability to reduce the RMSE. The neural mechanism works on the neurons and their corresponding weight applied [7]. Moreover, with more training data and more neurons added will produce closer results of prediction. Therefore, the proposed framework can be used to predict the workload to ensure a quality service is provided by the cloud providers to the users.

VIII. CONCLUSION

In this paper, we discussed about provisioning of cloud resources using time series and machine learning in the cloud environment. We introduced a dynamic provisioning framework to predict the workload. Using the framework, we calculated the expected load by both timeseries and machine learning techniques. The model comprised of predictor layer to predict the load. This layer also analyses the new request and ensure the analysed results are sent to the next layer. This would help in controlling the increasing volume of data throughput of enterprises and businesses which are moving towards the cloud computing services. It will also be helpful with analysis of configuration issues on provisioning these resources dynamically especially when the applications are running during the peak hours. We found that deep learning is efficient in identifying the load and predict the workload along with improving the efficiency of VM. We can also use the above predictions to determine the CPU and memory usages. The future studies will specifically focus on prediction of provisioning vertical or parallel processing of resources by exploring back propagation, RNN and other deep learning algorithms in order to predict efficient usage of the CPU requirements.

REFERENCES

[1] D. Durkee, "Why Cloud Computing Will Never Be Free," *Queue*, vol. 8, no. 4, pp. 20-29, 2010.

[2] M. Mao and M. Humphrey, "Auto-scaling to minimize cost and meet application deadlines in cloud workflows," in *SC '11: Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis*, 2011, pp. 1-12.

[3] M. Cheng, J. Li, and S. Nazarian, "DRL-cloud: Deep reinforcement learning-based resource provisioning and task scheduling for cloud service providers," in *2018 23rd Asia and South Pacific Design Automation Conference (ASP-DAC)*, 2018, pp. 129-134.

[4] M. Hassan, H. Chen, and Y. Liu, "DEARS: A Deep Learning Based Elastic and Automatic Resource Scheduling Framework for Cloud Applications," in *2018 IEEE Intl Conf on Parallel & Distributed Processing with Applications, Ubiquitous Computing & Communications, Big Data & Cloud Computing, Social Computing & Networking, Sustainable Computing & Communications (ISPA/IUCC/BDCloud/SocialCom/SustainCom)*, 2018, pp. 541-548.

[5] C. Liu, M. Shie, Y. Lee, Y. Lin, and K. Lai, "Vertical/Horizontal Resource Scaling Mechanism for Federated Clouds," in *2014 International Conference on Information Science & Applications (ICISA)*, 2014, pp. 1-4.

[6] F. Qiu, B. Zhang, and J. Guo, "A deep learning approach for VM workload prediction in the cloud," in *2016 17th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD)*, 2016, pp. 319-324.

[7] C. Bitsakos, I. Konstantinou, and N. Koziris, "DERP: A Deep Reinforcement Learning Cloud System for Elastic Resource Provisioning," in *2018 IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*, 2018, pp. 21-29.

[8] M. F. Arlitt and C. L. Williamson, "Internet Web servers: workload characterization and performance implications," *IEEE/ACM Transactions on Networking*, vol. 5, no. 5, pp. 631-645, 1997.

[9] H. Yan, P. Yu, and D. Long, "Study on Deep Unsupervised Learning Optimization Algorithm Based on Cloud Computing," in *2019 International Conference on Intelligent Transportation, Big Data & Smart City (ICITBS)*, 2019, pp. 679-681.

[10] J. Brownlee. (2019, 16/08/2019). *Deep Learning & Artificial Neural Networks*. Available: https://machinelearningmastery.com/what-is-deep-learning/

[11] Y. Huang, X. Ma, X. Fan, J. Liu, and W. Gong, "When deep learning meets edge computing," in *2017 IEEE 25th International Conference on Network Protocols (ICNP)*, 2017, pp. 1-2.

[12] Z. Zhao, K. M. Barijough, and A. Gerstlauer, "DeepThings: Distributed Adaptive Deep Learning Inference on Resource-Constrained IoT Edge Clusters," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, no. 11, pp. 2348-2359, 2018.

[13] J. Yang *et al.*, "A cost-aware auto-scaling approach using the workload prediction in service clouds," (in English), *Information Systems Frontiers*, vol. 16, no. 1, pp. 7-18, Mar 2014
2014-08-30 2014.

[14] C. Qu, R. N. Calheiros, and R. Buyya, "A reliable and cost-efficient auto-scaling system for web applications using heterogeneous spot instances," *Journal of Network and Computer Applications*, vol. 65, pp. 167-180, 2016/04/01/ 2016.

[15] C. Sung, C. Y. Higgins, B. Zhang, and Y. Choe, "Evaluating deep learning in chum prediction for everything-as-a-service in the cloud," in *2017 International Joint Conference on Neural Networks (IJCNN)*, 2017, pp. 3664-3669.

[16] B. Bao, Y. Xiang, L. Li, S. Lyu, H. Munshi, and H. Zhu, "Scalable Cloud Service For Multimedia Analysis Based on Deep Learning," in *2018 IEEE International Conference on Multimedia & Expo Workshops (ICMEW)*, 2018, pp. 1-4.

[17] H. Wang, J. Pannereselvam, L. Liu, Y. Lu, X. Zhai, and H. Ali, "Cloud Workload Analytics for Real-Time Prediction of User Request Patterns," in *Proceedings - 20th International Conference on High Performance Computing and Communications*, *16th International Conference on Smart City and 4th International Conference on Data Science and Systems, HPCC/SmartCity/DSS 2018*, 2019, pp. 1677-1684.

[18] B. Etienne. (2019). *Time Series in Python — Exponential Smoothing and ARIMA processes*. Available: https://towardsdatascience.com/time-series-in-python-exponential-smoothing-and-arima-processes-2c67f2a52788

[19] C. C. Aggarwal, "Neural Networks and Deep Learning," *Springer International Publishing*, 2018.