International Journal of Aquatic Science
ISSN: 2008-8019
Vol 12, Issue 02, 2021

LOAD BALANCING USING LSTM NETWORK AND DOCKER

G. Malar Selvi[1], S.Girirajan[2], J.Briskilal[3]

[1]Department of Computer Science and Engineering, SRM Institute of Science and Technology, Kattankulathur, Chennai, India
[2]Department of Computer Science and Engineering, Vel Tech Rangarajan Dr.Sagunthala R&D Institute of Science and Technology, Chennai, India

[3]Department of Computer Science and Engineering, SRM Institute of Science and Technology, Kattankulathur, Chennai, India

Abstract – Recurrent Neural Network is widely used in Natural Language processing (NLP) based task like Automatic Speech Recognition(ASR), Speaker Gender Identification, Speaker Identification, Speaker Emotion recognition. It is proved that RNN works well in time series data and also provide better accuracy in above mentioned Research task. In this proposed work we have implemented Long short-term memory (LSTM) a type of RNN for Load balancing in web server. Storing information and providing service to client based on their request is the primary work of web server. One of the major issues in web server is Load balancing. Existing methodologies for load balancing usually depend upon both hand-crafted infrastructure scales up and/or rule based algorithms such as scaling up when the server loads (CPU/memory) hit high enough to trigger rule based load balancing servers or by observing the requests manually and scaling up as needed. These techniques can result in significant delays during the rush hour. Our methods provide a supervised learning methodology. A recurrent neural network was devised to predict loads on the servers. The proposed system utilized a LSTM network to predict the said loads. The data from the LSTM network will be used to create additional containers (docker containers) to handle the load before it even happens hence preventing the system from any down time.

***Index Terms* – Load Balancing, LSTM, RNN, TCP, Web Server.**

I. INTRODUCTION

In recent years we can see the increase of usage in World Wide Web (WWW). Without any restriction almost all age group of peoples are using WWW in their day-to-day life. Due to increase in usage the architecture of WWW become more complex in recent years. There are several new methodologies are introduced to monitor or manage the performance of server. It is essential to manage the workload or resource allocation for server to get better performance and to improve the quality of service. The ability of classic load balancers depends upon the fact that whenever the CPU loads reach a certain point in

threshold, the infrastructure can be scaled up accordingly [3].

This leads to increased times and ultimately leads to delay in serving the content to the consumer. Several services use load balancer to handle the loads such as if you are watching a video on YouTube, searching on Google, connecting to other people on Reddit.

We all have seen errors stating that the said service is not able to handle the current traffic loads. In this approach, we will first be collecting data from the servers, such as access logs from Apache/Nginx web servers.

Upon, pre-processing of the data and realizing it in Number of requests/Per day format, we were be able to train our LSTM network to accurately predict future loads[5].

Upon predicting the future loads, we are able to deploy our containers for cost reduction/load balancing. This paper focuses on using LSTM network and containers to do prediction of load and handling it as well. This paper implements the a neural network assisted load balancer.

II.RELATED WORK

In the cloud computing various types of algorithm are used to balance the load [2]. The static algorithms used are as round robin, active clustering load balancing, central load balancing decision model, map reduced based entity resolution model. The dynamic load balancing algorithm used are equally spread current execution algorithm, throttled load balancing algorithm, CARTON Load balancing algorithm, ant bee optimization algorithm[8].

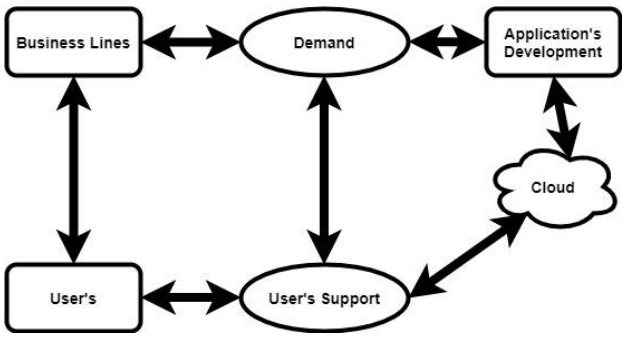

Fig 2.1 Carton Load Balancing

In recent year RNN are widely used in sequential data's such as speech and text. Especially LSTM generate better accuracy in sequential data. Due to this we tried to implement LSTM to handle the load balancing issue which we faced in cloud computing. Since log details and other features we used also a type of sequential data. The major reason behind the proposed work is to analyze the present methodology used to handle the load balancing in cloud environment by which we can able to use the resource more efficient way by allocating the equal amount of load to all available resources. By efficient load balancing methodology we

can able to provide the cloud storage with minimum cost since we were using the resources in more efficient way. For this we have discussed the pros and cons of the previous algorithms used for Load balancing [12-14].

The policies such as load balancing network traffic in TCP / IP protocol are used to increase the performance. Some research work is carried out by using those policies which gave efficient result [18]. By considering the economy of scale efficient design of network is introduced in network principles that well suitable for handling the traffic due to load in network. To handle the load traffic in network five methods are used apart from linear network design policies (server load, round robin, number of connection, random, and weighted fair cuing) against baseline scenario of no-load balancing (NLB)[17,20].

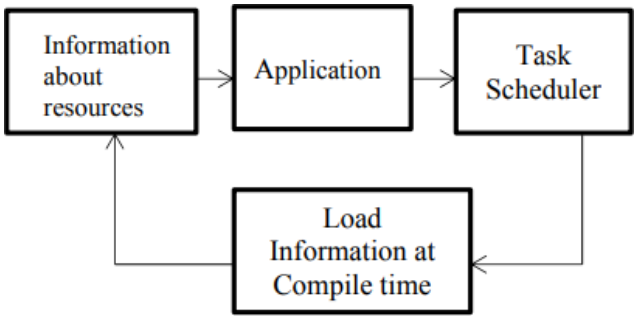

Fig 2.2 Static Load Balancing

Most of the organization uses the linear networking architecture in the proposed work we designed a Local Area Network that finds the similarity that occurs in those architecture. It is designed to not support the case for principles of load balancing but be limited to the server's computer architecture model load management strategies, but will also apply topical level of a network [9,10].

Most of the cloud architecture is used for providing multiple services in concurrent phase with minimum or no time. So we need to understand the architecture of such cloud environment since in recent years most of the internet sources shares there content based on cloud storage. For high level of end user network flexibility is experienced as high as the end of availability and short response time data rate increase along with end delays and shocks help. The assumption of such analysis allows user service quality expression, Individual results obtained by different L.B[12,13].

Due to policies included in network architecture price of that application increased huge. Consider the database applications where the response time will be minimum along with that if we use round robin weighted fair then we can able achieve better result in handling the load traffic in web server as well as in cloud environment. Such a architecture fully dependents on the number of connection made to the server or cloud architecture. From there the results of this simulation work seem to be clear application and protocol is determinant in behavior

Achieving preferential LB results for various policies. The proposed model is limited to linear networks architecture, but useful for reassembling discussions modeling for the evaluation of load balancing of a typical network infrastructure.
Future work will use this basis to study the effects of random networks and structured networks architectures, implement the quality of service provisions and network server failure recovery across multiple servers[14-16].

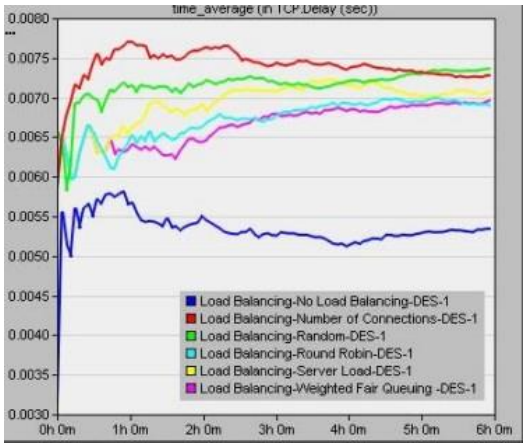

Fig 2.3 TCP Delay (sec) Time average for non-load balancing and load balancing policies

III. IMPLEMENTATION

A. Obtaining the dataset for load balancing

We collected our dataset by setting up a server by using nginx and it has been deployed in production environment. Then we collected the log details that are request made to server in single day. Based on the above set up we gathered nearly 198 log files from web server nodes. The log files details are in the form of URL. Apart from URL log file contains delay in request, no.of bytes transferred. We also tested performance by using following 3 parameters.

1. Timestamp of the request.
2. Requested URL
3. User agent for the request.

Based on the above mentioned parameter we have found that request delay is maximum 100ms. Since we have limitation in usage of GPU we trained the model with the node that contains maximum log details.

B. Pre-processing the dataset

As the obtained dataset cannot be directly used with the model, preprocessing needs to be done in order to make the dataset into a usable form. We just removed the request that made for image or json files. Later the valid URL is alone arranged in Alphabetical order. The dataset was stripped from the requested URL and user agent for the request and the rest of the data was aggregated such that it contains number of requests per 24 hours. From the raw data, information such as:

1. 'High' represents the maximum server requests that are processed
2. 'Low' represents the minimum server requests that are processed.
3. 'Average' represents the daily average server requests that processed.

C. Model

After the dataset was preprocessed and converted into usable form it was be able to fed into the model. The visual representation of the model can be seen in Fig 3.3. Various optimizers were tested with the dataset and optimizer 'Adam' was chosen as it gave the best results. The comparison between the various testing of optimizers can be seen in Fig 3.2.
Various filters are applied the data to remove the data from the bots such as Google/Bing/Yandex bot. Then, the data is used for training. The data is also split between testing and training purposes, in the ratio of 20:80.
This model is based upon a sequential model which contains 64 LSTM neurons with the input shape as 1 x 60. The activation function used is RELU.
The output layer is of 1 x 1.

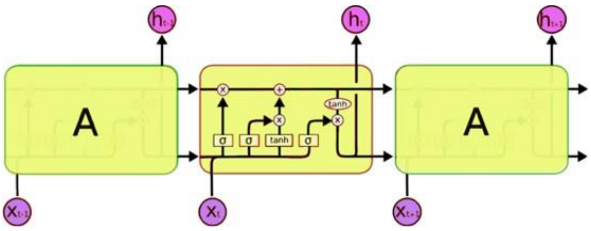

Fig 3.1 LSTM Layers

The loss function that was used is mean squared error to evaluate how a particular algorithm models the given data.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \tilde{y}_i)^2 \quad (1)$$

Measuring the difference between actual value and predicted value are carried out by MSE that stands for mean squared error. MSE is just the average of the squared difference between the predicted and the actual data points.

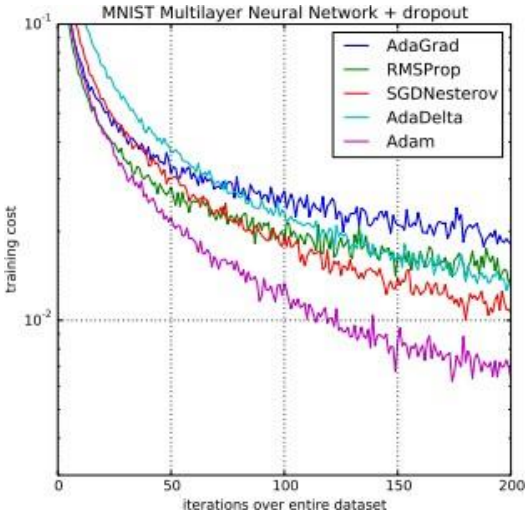

Fig 3.2 Testing of various optimizers

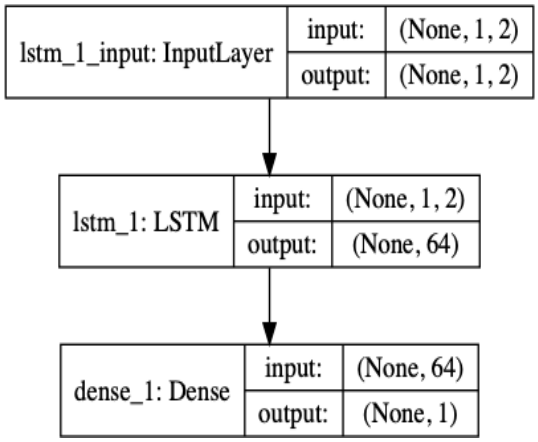

Fig 3.3 Network Architecture

D. Training

For training of the dataset, 80% of the dataset is kept for training and 20% is kept aside for testing. As due to the nature of our project, we don't require up-sampling or down-sampling of our dataset. After the training, the model was ready for deployment. Fig 3.4 shows the model loss after training was complete. Various methodologies were used to complete the training and verify the results in as little time as possible.

Tensorflow's early stopping callbacks were used in order to stop training when sufficient accuracy was achieved. The training time for 1 epoch was approximately 10 seconds. Hyper parameters that are used for the network are:

1. Epochs = 100
2. Batch Size = 8

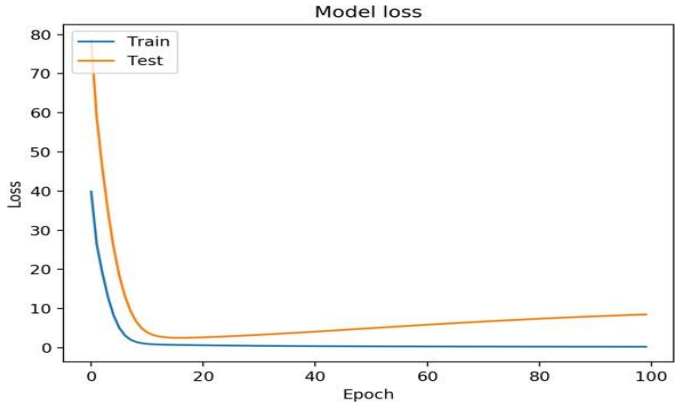

Fig 3.4 Model Loss

E. Custom Load Balancer

A custom load balancer was developed such that it is able to utilize the neural network and is able to predict the 'average' load each day. Whenever the average load predicted by the neural network exceeded that of the threshold additional containers are deployed. The number of current containers active at a given day is given by:

$$\boldsymbol{n = \left(\frac{a}{b}\right), n = ceil(n)} \qquad \textbf{(2)}$$

Where,
n: denotes the number of containers required
a: denotes the load predicted by the network
b: denotes capacity for each container
ceil: Ceiling(rounding up) function

F. Deployment

The deployed architecture consisted of a web server which was running Nginx (web server) on a container. The load-balancer was setup in a reverse proxy configuration which allowed it to pass the requests to multiple backend/containers. The load balancer continuously polls the network for the update in predicted network load and deploys the containers accordingly. Fig 3.5 shows the deployed architecture which was used to evaluate the results. In the given figure, Fig 3.5, the values of x, y and z are:

1. x = 5
2. y = 2
3. z = 2.5, ceil(2.5) = 3

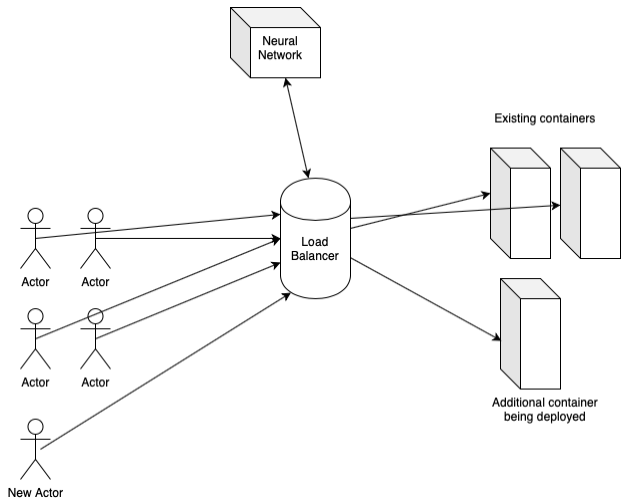

Fig 3.5 Deployed Architecture

IV. RESULTS

The combination of LSTM network to predict the load and container based system such as Docker and/or Kubernettes lead to promising results on the future of load balancing. The below graph shows the network's prediction on the number of requests per second/per day. Subsequently, the results were compared to the prediction done by decision tree regression as show in Fig 4.1 and 4.2 respectively.

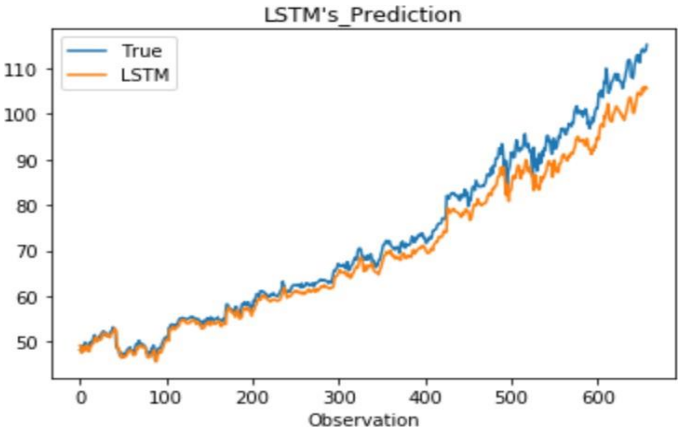

Fig 4.1 LSTM Model's prediction

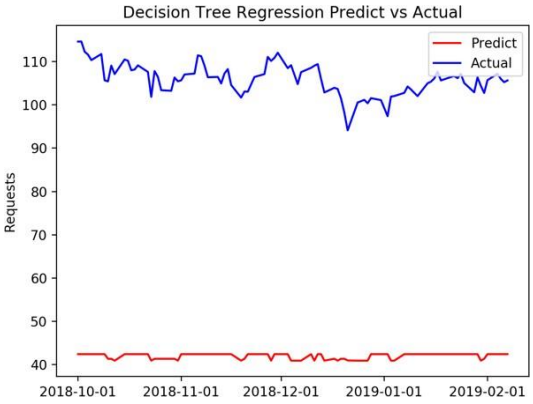

Fig 4.2 Decision Tree Regression Prediction

Compared to traditional load-balancer, our neural network assisted load balancer performed remarkably better in predicting loads and acting upon them.

V.CONCLUSION

In this proposed work we have used LSTM for load balancing in web server. The proposed can able read the features during the training phase itself without any knowing anything about it. Based on the experiments conducted on the our own dataset, we came to the conclusion that our model can able perform well in balancing load when compare with previous state of art methodology.

REFERENCES

[1] Klaithem Al Nuaimi, Nader Mohamed, Mariam Al Nuaimi and Jameela Al-Jaroodi "A Survey of Load Balancing in Cloud Computing: Challenges and Algorithms" IEEE 2012.

[2] Surbhi Kapoor, Dr. Chetna Dabas "Cluster Based Load Balancing in Cloud Computing" IEEE 2015.

[3] Abhinav Hans, Sheetal Kalra "Comparative Study of Different Cloud Computing Load Balancing Techniques" IEEE 2014.

[4] Reena Panwar, Prof. Dr. Bhawna Mallick "Load Balancing in Cloud Computing Using Dynamic Load Management Algorithm" IEEE 2015.

[5] Kumar Nishant, Pratik Sharma, Vishal Krishna, Chhavi Gupta and Kuwar Pratap Singh "Load Balancing of Nodes in Cloud Using Ant Colony Optimization" International Conference on Modelling and Simulation IEEE 2012.

[6] J.Prasanna, Ajit jadhav, and V.Neel Narayan "Towards an Analysis of Load Balancing Algorithms to Enhance Efficient Management of Cloud Data Centres," Springer, 2016.

[7] Aarti Singha, Dimple Junejab, "Manisha Malhotra "Autonomous Agent Based Load Balancing Algorithm in CloudComputing" Elsevier,2015.

[8] Tushar Desai, Jignesh Prajapati "A Survey Of Various Load Balancing Techniques And Challenges In Cloud Computing" IJSTR,2013

[9] Reena Panwar , Prof. Dr. Bhawna Mallick "Load Balancing in Cloud Computing Using Dynamic Load Management Algorithm"IEEE 2015

[10] Sandeep Negi, "An Improved Round Robin Approach using Dynamic Time Quantum for Improving Average Waiting Time" International Journal of Computer Applications Vol. 69 : No 14, May 2013.

[11] M.N.O. Sadiku, S.M. Musa and O.D. Momoh, "Cloud Computing: Opportunities and Challenges", Potentials IEEE, vol. 33, no. 1, pp. 34-36, Jan.–Feb. 2014.

[12] Frederico Alvares, Gwenaël Delaval, Eric Rutten, and Lionel Seinturier. 2017. Language support for modular autonomic managers in reconfigurable software components. In Proceedings of the 2017 IEEE International Conference on Autonomic Computing. IEEE, 271--278.

[13] Mohammad Sadegh Aslanpour, Mostafa Ghobaei-Arani, and Adel Nadjaran Toosi. 2017. Auto-scaling web applications in clouds. Journal of Network Computer Applications 95 (2017), 26--41.

[14] Reena Panwar and Bhawna Mallick, "Load Balancing in Cloud Computing Using Dynamic Load Management Algorithm", IEEE International Conference on Green Computing and Internet of Things, pp. 773-778, 2015.

[15] Xiaofang Li, Yingchi Mao, Xianjian Xiao and Yanbin Zhuang, "An Improved Max-Min Task-Scheduling Algorithm for Elastic Cloud", IEEE International Symposium on Computer Consumer and Control, pp. 340-343, 2015.

[16] Bokhari MU, Shallal QM, Tamandani YK (2016, March) Cloud computing service models: a comparative study. In: 3rd international conference on computing for sustainable global development (INDIACom), 16–18, March 2016, pp 890–895

[17] Jain N, Choudhary S (2016, March) Overview of virtualization in cloud computing. In: Symposium on colossal data analysis and networking (CDAN), pp 1–4

[18] Alouane M, El Bakkali H (2016, May) Virtualization in cloud computing: no hype vs HyperWall new approach. In: 2016 International Conference on Electrical and Information Technologies (ICEIT), pp 49–54

[19] Noshy M, Ibrahim A, Ali HA (2018) Optimization of live virtual machine migration in cloud computing: a survey and future directions. J Netw Comput Appl:1–10

[20] Raviteja S, Atmakuri R, Vengaiah C (2017) A review on cloud computing migration and issues