National College of Ireland

Improvement in auto scaling mechanism of cloud computing resources using Composite ANN

Research Project
Msc Cloud Computing

Ashish Ekhande
Student ID: x18148221

School of Computing
National College of Ireland

Supervisor: Manuel Tova-Izquierdo

National College of Ireland
Project Submission Sheet
School of Computing

National College of Ireland

Student Name:	Ashish Ekhande
Student ID:	x18148221
Programme:	Msc Cloud Computing
Year:	2020
Module:	Research Project
Supervisor:	Manuel Tova-Izquierdo
Submission Due Date:	20/12/2018
Project Title:	Improvement in auto scaling mechanism of cloud computing resources using Composite ANN
Word Count:	4040
Page Count:	16

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

I agree to an electronic copy of my thesis being made publicly available on TRAP the National College of Ireland's Institutional Repository for consultation.

Signature:	
Date:	4th June 2020

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	☐
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	☐
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	☐

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Improvement in auto scaling mechanism of cloud computing resources using Composite ANN

Ashish Ekhande
x18148221

Abstract

In cloud computing, auto scaling has attracted considerable attention from researchers and organizations for acquiring and releasing resources based on real-time demand from large data centers. Elasticity enables auto scaling resources on-demand; thus, a pay-as-you-go model can be implemented. Auto scaling suffers from various complexities and challenges because allocation and de-allocation of resources is dynamic as per the workload demand. The false prediction of resource allocation may lead to reduced quality of service and violation of service level agreements. For predicting resource requirements, various machine learning and deep learning (DL) technologies have been used. Although DL methods are accurate with predictions, they require Big Data. With increase in the size of input data, the model's complexity increases, which leads to overhead on the system. In this study, an artificial neural network (ANN) with linear regression is used to efficiently predict resource requirements. Aim is to achieve optimal resource allocation for systems with limited computing capacity and minimum overhead on the system. To minimize the input and utilize less hardware resources for prediction while retaining accuracy in the output that was produced, ANN with optimized linear regression is used. The results demonstrate that linear regression optimally minimizes weights on the nodes through each layer. Furthermore, when this minimized output is allocated to an ANN, it produces an ameliorated output with minimum overhead on the system.

1 Introduction

1.1 Cloud computing at a glance

Cloud Computing provides virtual resources which are accessible almost anytime anywhere. It is broadly bifurcated into SaaS, IaaS and PaaS. Each form having its own unique usage. Businesses have greatly boomed over the years due to the upsurge of Cloud Computing technology. In the Cloud technology, we use remote resources which in turn prove to be beneficial for the organizations in saving costs for servers and other hardware equipment. Dynamic provisioning of IT services is employed and carried out in a faster, cost effective manner.

There are plethora of reasons to deploy this mechanism, one of them being reduced costs where hardware implementation is least as everything is stored on the cloud servers. Secondly, it is quite secure as the information stored on cloud is encrypted and regular backups are initiated at defined intervals. The best part comes in terms of storage, where the end user gets tremendous amount of storage virtually which is the need of the hour in today's world especially for any kind of data handling.

Cloud computing provides scalable yet flexible solutions to customers. It is quite elastic and ubiquitous in nature and from there we can derive auto-scaling, where users can literally have as much or as little as they require at a given time. Workload resilience is implemented for better performance. It ensures efficient utilization of resources, minimizes rent expenditures, and ensures sufficient capacity for processing.

1.2 Challenges in deploying cloud techniques

Employing Cloud Computing alone also comes with a few drawbacks. We need more understanding of the complex, diverse and unstructured data lying around. There is a high chance where this data is quite inter connected but not understandable to the human brain at a glance. We need more powerful computational tools to assess this data with combinations, patterns and working algorithms persuaded by human brains to configure the "unseen hidden" data. This is achieved by interlinking the concepts of Deep Learning in Cloud Computing. The big cloud computing companies are venturing on democratizing artificial intelligence on a much larger scale.

One of the drawbacks in cloud computing is managing the variable resources and allocation. This creates a set back to the system and cannot be used to its maximum efficiency. With the auto scaling technique that I have proposed to use in my work, this drawback is eradicated and cloud resource management is used at its fullest through auto scaling. Load balancing is maintained without causing any issues to the live users.

Required resources should not be over-provisioned As it leads to increase in cost; however, when under-provisioned, the resources will definitely not have the capacity to meet the customer's requirements (Nikravesh et al.; 2015). Because the workload demand of customers is volatile, the automatic scaling provision is essential to scale resources on demand. This provision also ensures that human intervention is completely eliminated and overall cost is reduced while simultaneously fulfilling the quality of service (QoS) Qu et al. (2018).

The on-demand pricing model is a strategy in which customers can be provided with

real-time cloud resources for a fixed cost. Amazon S3 and Microsoft Azure are well-renowned cloud service providers that have been offering services using this model Evangelidis et al. (2018). Many scaling architectures have been used for both horizontal and vertical scaling, and researchers are trying to resolve problems in scaling the resources using machine learning (ML) techniques and strategies. In fact, threshold mechanisms have been used the researchers, but they have proven to be inadequate.

1.3 Motivation to use ML for Auto scaling

Auto scaling is being used to scale resources because it propose solutions to the ever-growing demand along with performance. To implement auto-scaling for dynamic allocation of resources, certain rules are required. In the literature, the two major approaches for auto-scaling are proactive and reactive approaches Gajjar and Shah (2015). In the proactive approach, auto-scaling makes use of ML and predicts the future utilization of resources; however, in the reactive approach, it focuses on to the current state of the system and allocates resources accordingly.

Artificial neural networks, are a combination of outer layers with interconnecting neurons .The layers help in understanding the complex patters of the objects. I believe that with this technique, the workload prediction in cloud computing can be improved to a great extent. If a model is implemented with ANN linear regression, it is capable to produce an appropriate mutation strategy coupled with an optimum crossover rate. With the proper input and feedbacks given to these models, we can imagine what great heights we can achieve with the computations.

Many researchers have attempted to address auto scaling using proactive approaches and artificial intelligence, ; however, it lead to overhead on the system and performance degradation. Therefore, in this study, I have used linear regression to accurately predict time series-based inputs of data and then resources are allocated by passing minimized weights to ANN.

It is quite positive to believe that Cloud Computing coupled with ANN linear regression will achieve the goals predefined. Over the next few decades this technology is prone to emerge among IT decision-makers and business leaders are likely to invest and use the benefits offered by this model. This technology will prove to be a top picked destination to operate on the higher cloud platform providers. To continue to supply unlimited services on demand, auto scaling is pre-eminent with the help of such modified tools and makes the platform an "Intelligent Cloud".

1.4 Research objectives

- **How efficiently can composite learning (i.e., ANN with linear regression) perform auto-scaling as compared to other techniques?**
- **Does minimizing the system overhead impact the model's accuracy?**

The aim of this research is to implement a ANN with linear regression to improve the auto scaling mechanism of cloud resources for different workloads.

2 Literature Review

Many studies have studied the development of efficient auto-scaling systems for cloud computing. In this section, I am going to review some of the relevant studies in this area.

2.1 Previous work in auto-scaling

Singh and team Singh et al. (2019) reviewed the existing techniques in the literature and categorized the challenges in auto scaling multi-tier web applications. They addressed all existing techniques of auto-scaling that tend to maintain service level agreements (SLAs) while simultaneously improving the QoS. Qu et al. (2018) surveyed auto-scaling of web applications with resource estimation performed using auto scaling techniques such as rule-based strategies, fuzzy inference, analytical modeling, artificial intelligence, and hybrid techniques. Gajjar and Shah (2015)reported that hybrid techniques that use the strength of ML techniques and analytical modeling are successful in resource estimation.

In Guan et al. (2017) ,authors have introduced the strategy to efficiently train deep learning model with asynchronous inputs. When compared with Fazayeli (2014) individual Adagrad and RMSProp algorithms, demonstrated strategy accelerates training process with low consumption of memory. From results of the proposed model, it is evident that almost linear speedup is achieved when model is trained with multiple inputs.

2.2 Threshold based traditional systems for Auto-scaling

Liao et al. (2015)) consider that workload demands in cloud computing are compromised to reduce the operating of cost of virtual machines. They mitigated increased cost and improved the response time of the system by proposing a dynamic threshold adjustment strategy. Biswas et al. (2015)introduced a broker that acquired resources as per demand from the cloud and ensured that QoS is maintained and cost is reduced. The results were obtained by analyzing the impact of the load factor, arrival rate, laxity factor, service time, and user charge.

Fallah et al. (2015) proposed a novel auto scaling approach based on learning automata (NASLA). They proved that the proposed approach tends to minimize overhead of scaling procedure and outperforms threshold-based algorithms. Mahmud et al. (2015) proposed an algorithm called budget-constrained auto scaling (BATS). For performance evaluation, the method was implemented on Microsoft Azure with a decrease in delay by 34%. Furthermore, the budget was evenly distributed and cost was reduced to 10% compared with threshold-based systems.

T. Chen (Chen and Bahsoon; 2015) worked on self-aware and self-adaptive cloud auto-scaling systems and achieved maximum throughput and minimized the cost. They offered a solution for optimal trade-off using ant colony algorithms. Similarly,(Biswas et al.; 2015) reported that the proposed technique transcends other techniques when making a quality trade-off decision. However, with this trade-off, there is considerable violation of requirements. Evangelidis et al. (2018) used a probabilistic model and proposed an approach that amends the time and cost of testing the cloud-based applications while maintaining accuracy in results.

Tseng et al. (2018) introduced the fog platform using a hypervisor and confirmed with help of their proposed technique that an enhanced service scale can be achieved. Guo et al.

(2018)worked on auto-scaling of applications using a shadow routing-based approach to minimize the use of physical machines. Han et al. (2012)) introduced a light-weight and cost-effective scaling approach using LSU and LSD algorithms.

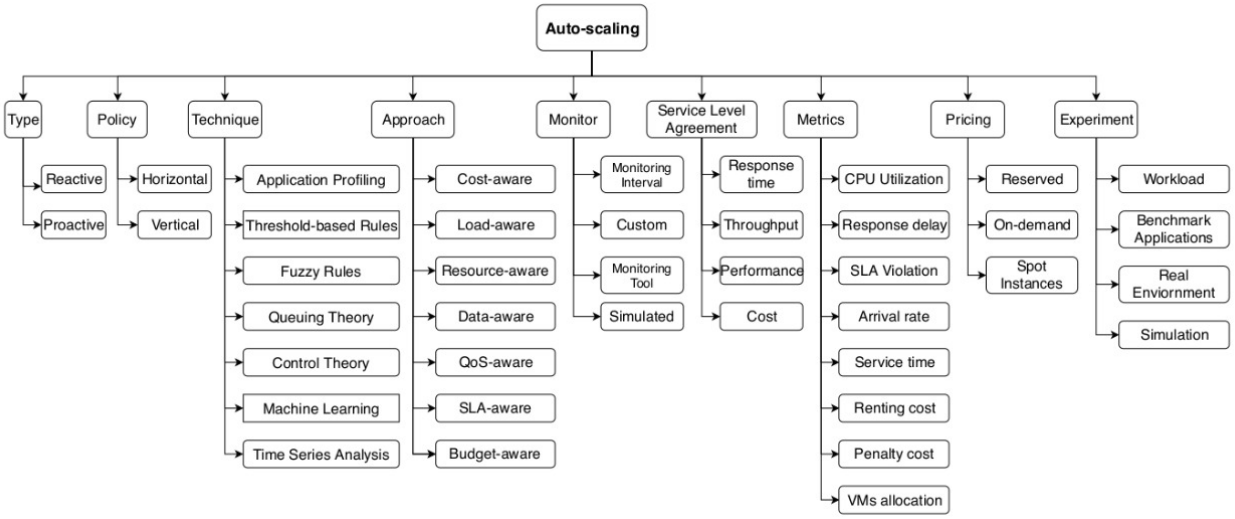

Figure 1: Auto Scaling Taxonomy

2.3 Proactive approaches for Scaling

According to Arabnejad et al. (2016),ML techniques can provide improved solutions to resolve problems associated with auto scaling compared to the traditional threshold-based techniques. This algorithm is a combination of fuzzy control that is responsible for mapping auto-scaling variables and fuzzy Q-learning algorithm responsible for the learning process and for allocating policies at run time. Hwang et al. (2015) reviewed approaches such as implementation of proactive scalability in hybrid clouds using ML and proactive memory scaling. They claimed to have increased accuracy of predictions and improved resource allocation; however, the results produced were not promising, and the detailed report of the events and experiments was missing and trade-off was unclear.

Benifa and Dejey (2019) proposed reinforcement learning-based pro-active auto-scaler (RLPAS) algorithm that learns from the parallel operating environment and allocates resources. The algorithm provides the optimized solution as resource utilization and the throughput is improved with reduction in response time. Parekh and Pandi (n.d.)proposed a ML-based auto-scaling approach for a cloud environment that helps in predicting accurate workloads. They used a broker management that handles incoming requests using a match-making algorithm.

In dynamic traffic changes, a series of experiments were performed by Rahman et al. (2018) to observe response. The algorithms used include random forest (RF), J48, REPtree, decision layer, multilayer perceptron and Bayesian network. They confirmed that the RF technique outperforms all other techniques. Qu et al. (2018) classified ML approaches for auto-scaling web applications.Souza and Netto (2015) proposed a load algorithm that computes the distribution of delays and an adaptation algorithm that handles the peaks. These algorithms were compared to the state-of-the-art threshold

algorithm. The results demonstrate that the load algorithm consumes fewer resources compared to others, whereas the threshold algorithm performs better when a large set of data is used.

Biswas et al. (2015) proposed a hybrid auto-scaling algorithm that works on SLS-driven architecture that caters advanced reservations and on-demand requests. They used various workload parameters such as load factor, MST, MAR, laxity factor, user, and broker cost rate. All of these parameters studied with each other and confirmed that higher profits can be achieved. Islam et al. (2012) worked on empirical prediction models, and the proposed approach uses error correction neural network (ECNN) and linear regression combined with a sliding window technique for the effective forecasting of resource utilization.

Nikravesh et al. (2015) worked on proactive auto scaling and combined it with time-series prediction algorithms. To predict the future characteristics of a virtual system, they used neural networks and support vector machines. Moreover, three types of workloads were considered such as growing workload, periodic workload and unpredicted workload. After establishing the parameters for multi-layer perceptrons, both support vector machine (SVM) and NNs were trained for learning for prediction. The proposed method confirmed that when there are predicted workload patterns, the SVM provides improved accuracy compared to NNs, although in an unpredicted environment a NN performs better than SVM. Table 2.3 shows a comparative analysis for different auto scaling approaches.

2.4 Review on Gaps in Literature

Many researchers have come up with very innovative ideas to the automatic scaling mechanism for computing resources. We discussed this technique previously in 2. However, traditional systems with threshold based auto-scaling mechanism fail to accommodate fluctuating demand of workload which ultimately results in over or under-provision of resources. Machine Learning based techniques uses large volume of historical data generated from the different sensors of the system. As the number of input increases, complexity of the model also increases and this may result in increasing system overhead. Adam algorithm Guan et al. (2017), which is used to train model faster using asynchronous inputs can be utilised to reduce the model complexity while maintaining good accuracy. Linear regression coupled with ANN can be used to achieve these results. The proposed work is divided into various stages, and each section will be discussed in detail in the upcoming sections.

The comparative analysis for different auto scaling approaches is shown in Table 2.4

Paper Title	Method	Advantages	Gaps in Literature
An auto-scaling frame-work for controlling enterprise resources on cloud	Threshold based broker	Variety of factors are considered for auto-scaling	Real-time provision of resources can not predict fluctuating demands.
An Efficient Approach for Resource Auto-Scaling in Cloud Environments	Reinforcement Learning with Markov Decision Process (MDP)	Cost Aware model is better in terms of SLA violation and system stability.	High Vectorial spatial complexity.
Auto-Scaling Network Resources using Machine Learning to Improve QoS and Reduce Cost	Random forest, Decision Tree, J48, MLP, BayesNet, Decision table and Random tree	ML classifier learns from historic VNF auto-scaling decisions and accuracy of 96.5%.	Computing resources are not considered in experiment
A Reinforcement Learning Based Auto-Scaling Approach for SaaS Providers in Dynamic Cloud Environment	Q-Learning based self adaptive method (Implemented in Matlab)	Performed Optimal resource allocation in dynamic cloud environment by considering different VM pricing mechanism	learning process is expensive to cover optimal policy.
Towards an Autonomic Auto-Scaling Prediction System for Cloud Resource Provisioning	Support vector machine (SVM) and Neural Network (NN)	SVM predicted better results for growing and periodic workloads, while the neural network Focused only on the predicted workloads.	Performed better for unperiodic workloads. Accuracy depends upon the nature of workload.

Table 2 : Comparison of Different Works for Auto Scaling Mechanism

3 Methodology

The key step while performing auto scaling tasks is the scaling controller component. In this section, methodology for an ANN and linear regression is discussed to provide efficient solution by maximizing the accuracy.

3.1 Artificial Neural Network

Based on a biological neural network, an ANN is a computing system that has the ability to learn from data to perform a specific set of tasks without applying any rule-based methods. ANNs are called as multi-layer perceptrons, and nodes in the ANN that are connected to each other are neurons. The connection with each neuron is called edges, and a weight is associated with both neurons and edges, which changes itself as learning proceeds. A Neural network comprises three layers: input layer, hidden layer, and output layer. In the Neural network, except the input node, each node has the weighted sum of inputs, which is forwarded using a non-linear activation function. The direction of signals is in the forward direction from left to right.

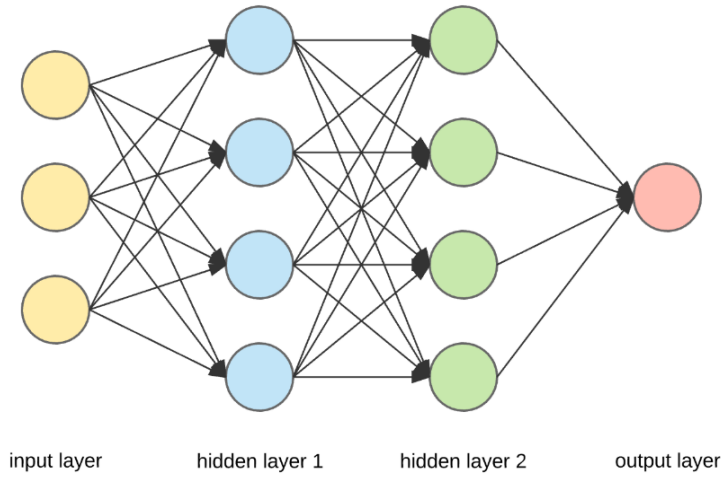

Figure 2: Simple ANN architecture

3.1.1 Tunning of Hyper Parameters

A manual tuning of hyper parameters is essential to efficiently train the model. It involves the selection of activation function, optimization algorithm, input size, output size, number of hidden layers, number of nodes in hidden layers, input format, and output format. A rectified linear unit (ReLU) activation function is responsible for determining weights and hidden neurons associated. In our experiment, the input size is 9, the size of output layer is 1, and two hidden layers are being used. The number of hidden layers primarily depends on the input. If the number of nodes in the hidden layer is more than the input size, the model may suffer from over-fitting. To overcome this limitation, different number of nodes were used for both models, and both models of ANN were trained over 200 epochs. Moreover, to optimize the algorithm with training data, the Adam optimization algorithm has been used. Adam optimization combines the property of adaptive gradient algorithm (AdaGrad) and root mean square propagation (RMSProp).

3.2 Linear Regression

Linear regression is the linear modelling approach for performing forecasting and predictions. The model can then be trained with datasets containing explanatory variables, which either can be independent or dependent in nature. Linear regression is useful for identifying a relationship between predicted and actual values by fitting a linear equation. It can be used for classification, regression, and for time series prediction analysis. A simple linear regression can be written as follows:

$$y = b_0 + b_1 * x_j \tag{1}$$

where y is the dependent variable, b_0 is constant, b_1 is the coefficient and x_j is the independent variable.

4 Design Specification

The proposed framework has been used to dynamically allocate resources as per the requirement. The development of this system involved various steps such as data generation, task queue, virtual machine of processing tasks and, most importantly, a scaling controller that required to dynamically allocate the resources as per the system's requirement and the number of tasks that are allocated to the system. All histories were plotted with the help of an activity plotter.

4.1 Data Generation

To perform any predictive analysis historical To perform any predictive analysis, historical data are required. In this work, time-series based synthetic dataset is generated with the help of a task generator, which is a class that can generate random work load in a system. It has the ability to generate M new tasks in a second and uses the sine function to manipulate load according to the sine wave. The generated tasks will be stored in a task queue. To randomly generate tasks with different workloads, mathematical sine function is used with a randomization feature.

4.2 Task Queue

The task queue simulates a virtual task storage system. After generating random tasks, all tasks will be temporarily stored in a task queue. Task queue is a place where all tasks can be stored and later will be used to process tasks in machine.

4.3 Machine

Machine simulates the working of any computing resources such as GPU, CPU, or RAM within a cloud environment. It fetches tasks from a task queue and processes them. Before the tasks are fed, the machine has a processing capacity of N, which can be dynamically changed depending on the work load of the machine. Moreover, machine maintains the history about the changes of its resources.

4.4 Scaling-Controller

Scaling Controller is an important component of the proposed system; it scales the resources up or down based on the predicted future requirements. To have a comparative analysis between different architectures of simple ANN and proposed ANN (Composite ANN), two types of controllers; simple scaling controller (SSC) and composite scaling controller (CSC) are simulated. Architecture and results for both systems are discussed in subsequent section.

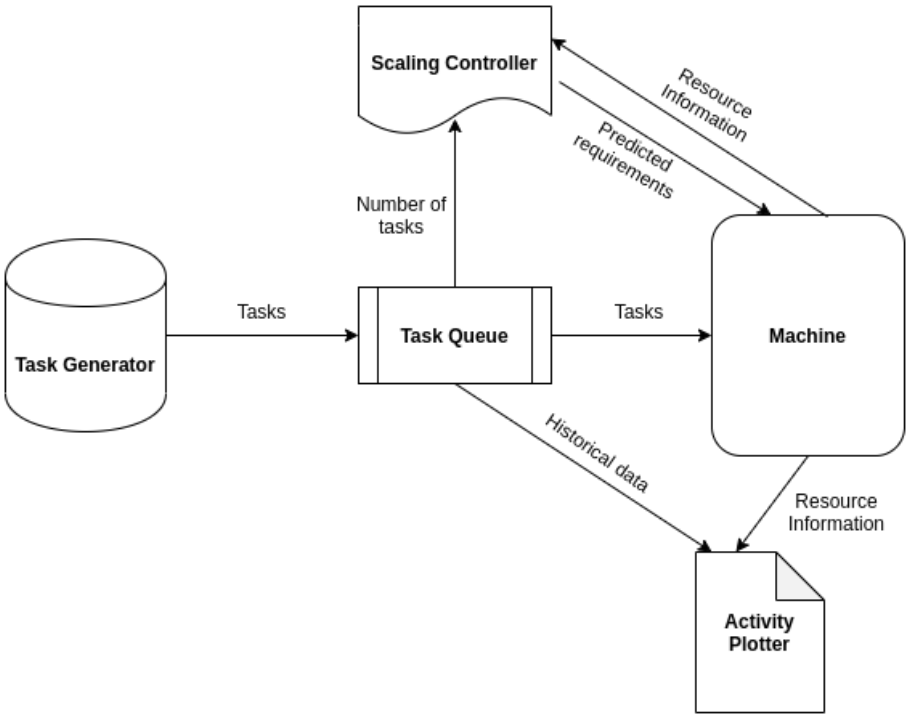

Figure 3: Flow Diagram of Proposed Work

The primary difference between SSC and CSC is that SSC uses a simple ANN to process data and perform predictive analysis. However, the CSC uses the combination of linear regression and ANN to process data. Here, the linear regression is used to reduce the number of inputs; subsequently, the reduced inputs will be processed into an ANN. This change can reduce complexity to a considerable extent and will result in reducing system overhead and improve predictions.

4.5 Activity Plotter

Activity Plotter collects information from the machine and task queue, and then provides a graphical representation to analyze system activities in an efficient manner. It contains the time-series data and has information about current execution and predicted requirements. The result section shows the graphical representation of Activity Plotter.

5 Implementation

In this work, a virtual system is demonstrated to incorporate the auto scaling mechanism. To implement the proposed system described in 3 Python programming language is used. Python is high level language equipped with extensive libraries which can take advantage of machine learning capabilities. The mentioned system has been run on a virtual machine with specifications.

- Operating system : Ubuntu 18.04 LTS (64 bit)
- CPU : 4 CPU's
- RAM : 4 GB
- Storage : 10 GB
- Language used : Python
- Libraries Required : Numpy, sklearn, tensorflow 2.0, matplotlib

Virtual tasks were generated with the concept of threading. Classes such as task queue and machine are defined to process these threads, which were randomly generated and used to manipulate the system's load. Two ML approaches: linear regression and ANNs are used for predictions and resource consumption. To perform mathematical operations, NumPy was used. As ANN is being used to train the model, Tensorflow 2.0 is a ML library for python which is leveraged to efficiently train the model. To plot the comparison graph in Activity Plotter, Matplotlib was used. 200 epochs are used to train both models. ReLU was used as an activation function. Then, to optimize the algorithm with training data, Adam optimization algorithm was used. Adam optimization combined the property of AdaGrad and RMSProp to develop an optimized algorithm.

6 Results and Discussion

In this section, the comparative analysis between both methods is discussed.

6.1 Experiment 1 / Simple Scaling Controller

SSC uses the simple ANN to process data and perform predictive analysis. However, with increase in the input size of the model, there is a considerable increase in the number of weights, which increases the model's size. If the input size is 9 and two hidden layers are being used, i.e. one with eight nodes and another with six nodes, respectively, with output layers of one node; Then, the total number of trained weights is 9 * 8 + 8 * 6 + 6= 126. Here, 126 are the total number of trained weights, and Figure 4 shows the result for SCC.

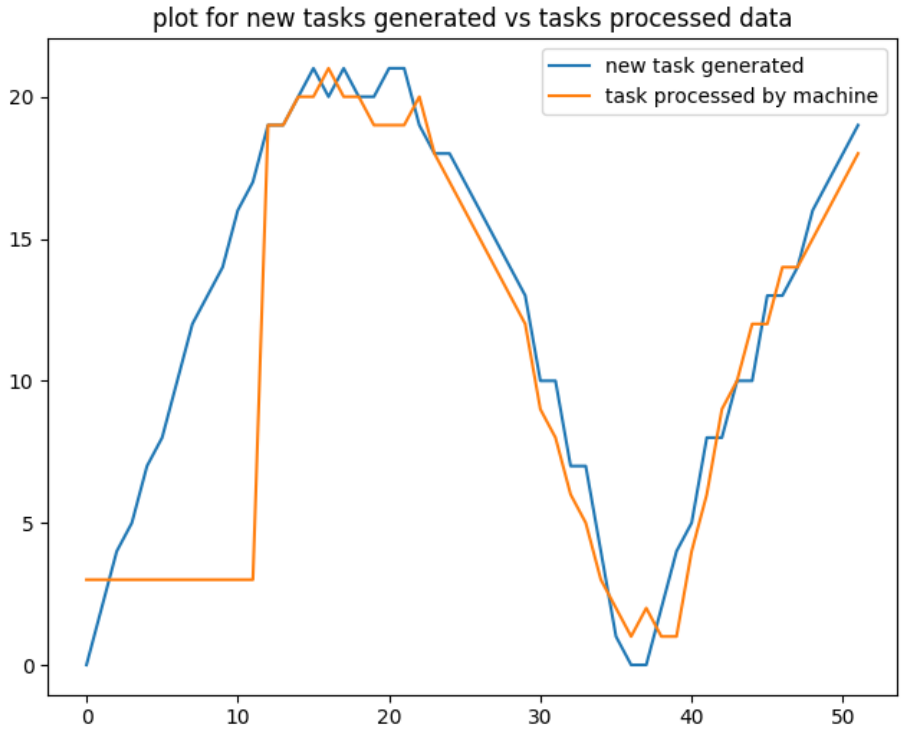

Figure 4: Graphical representation for Simple Scaling Controller

Here the orange line shows the predicted requirements and blue line shows the processed tasks.

6.2 Experiment 2 / Composite Scaling Controller

CSC combines linear regression and ANN to process data. In CSC, linear regression is used to simplify the number of inputs, and then the reduced inputs are processed into the ANN. Now Suppose, if the input size is 5 and 2 hidden layers with 5 nodes each are used respectively with output layers of 1 node. Then, the total number of trained weight is 5 * 5 + 5 * 5 + 5 = 55. Here, 55 is the total number of trained weights, and the size of input is reduced. Therefore, Using the proposed method, model's complexity is reduced to a considerable extent. Figure 5 shows the result for CSC.

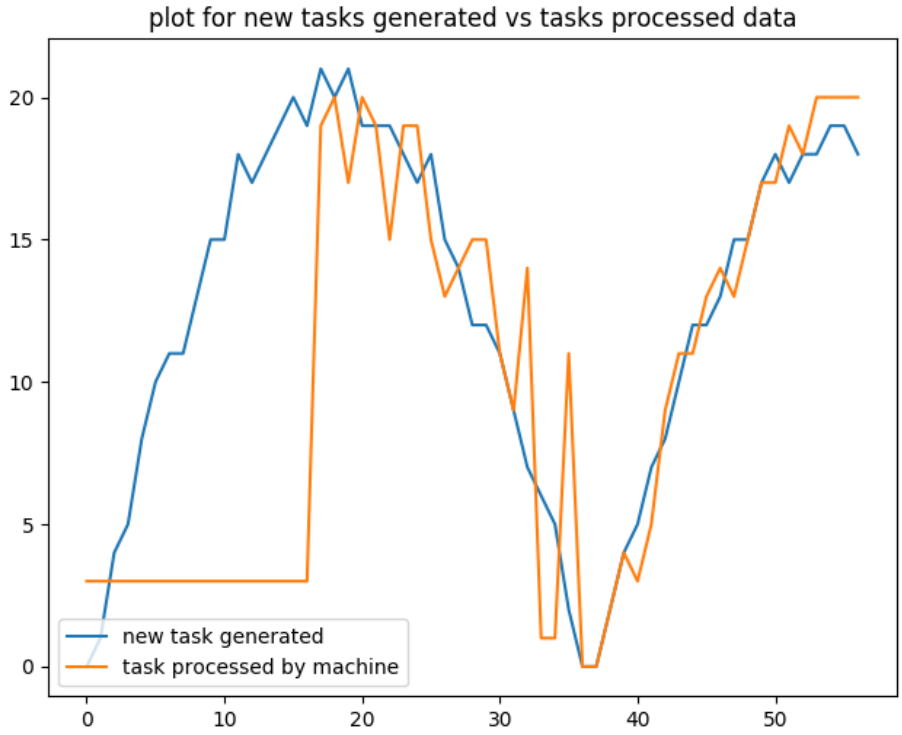

Figure 5: Graphical representation for Composite Scaling Controller

Here, in Graph 5, the orange line shows the predicted requirements and the blue line shows the processed tasks. This will affect the model's accuracy but provides an auto scaling mechanism with minimum overhead on the system.

6.3 Critical Analysis & Discussion

In this section, the results are being critically analyzed. ANN identifies the amount of resources based on predicting the previous time-series data. After observing the results of both models, SSC's accuracy is analyzed and ANNs have more accuracy compared to composite controllers. As the size of input increase, the complexity of ANN exponentially increases. To reduce the complexity, linear regression is used and then simplified the number of inputs using linear regression. Subsequently, this input was forwarded to an ANN and will be trained over 200 epochs. The proposed approach was simple but very effective to reduce the complexity to a considerable extent. Simplifying the inputs with linear regression reduces the model's accuracy to small acceptable extent; however, by compromising with little accuracy, a considerable amount of reduced complexity is achieved.

7 Conclusion and Future work

The proposed system (CSC) can correctly predict the resource requirements and accurately allocate or de-allocate resources with limited computing capacity and minimum overhead on the system. This method is more useful when the number of inputs are enormous. The model's complexity can be reduced to a considerable extent, and the advantage of the proposed system is that it can be on CPU and a less-powerful GPU.

The model can be helpful for startups or small companies where they have limited infrastructure and computing capabilities and want to provide auto scaling to customers. We always strive to optimize solutions and in future, an attempt to combine multiple AI algorithms to obtain the best cost-effective performance can be made. The proposed model can then be extended to any time-series data such that it can be useful for various applications.

References

Arabnejad, H., Jamshidi, P., Estrada, G., El Ioini, N. and Pahl, C. (2016). An auto-scaling cloud controller using fuzzy q-learning-implementation in openstack, *European Conference on Service-Oriented and Cloud Computing*, Springer, pp. 152–167.

Benifa, J. B. and Dejey, D. (2019). Rlpas: Reinforcement learning-based proactive auto-scaler for resource provisioning in cloud environment, *Mobile Networks and Applications* **24**(4): 1348–1363.

Biswas, A., Majumdar, S., Nandy, B. and El-Haraki, A. (2015). An auto-scaling framework for controlling enterprise resources on clouds, *2015 15th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, IEEE, pp. 971–980.

Chen, T. and Bahsoon, R. (2015). Self-adaptive trade-off decision making for autoscaling cloud-based services, *IEEE Transactions on Services Computing* **10**(4): 618–632.

Evangelidis, A., Parker, D. and Bahsoon, R. (2018). Performance modelling and verification of cloud-based auto-scaling policies, *Future Generation Computer Systems* **87**: 629–638.

Fallah, M., Arani, M. G. and Maeen, M. (2015). Nasla: Novel auto scaling approach based on learning automata for web application in cloud computing environment, *International Journal of Computer Applications* **113**(2).

Fazayeli, F. (2014). Adaptive subgradient methods for online learning and stochastic optimization.

Gajjar, P. and Shah, B. (2015). Survey on different auto scaling techniques in cloud computing environment, *International Journal of Advanced Research in Computer and Communication Engineering* **4**(12): 2278–1021.

Guan, N., Shan, L., Yang, C., Xu, W. and Zhang, M. (2017). Delay compensated asynchronous adam algorithm for deep neural networks, *2017 IEEE International Symposium on Parallel and Distributed Processing with Applications and 2017 IEEE International Conference on Ubiquitous Computing and Communications (ISPA/IUCC)*, IEEE, pp. 852–859.

Guo, Y., Stolyar, A. and Walid, A. (2018). Online vm auto-scaling algorithms for application hosting in a cloud, *IEEE Transactions on Cloud Computing* .

Han, R., Guo, L., Ghanem, M. M. and Guo, Y. (2012). Lightweight resource scaling for cloud applications, *2012 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (ccgrid 2012)*, IEEE, pp. 644–651.

Hwang, K., Bai, X., Shi, Y., Li, M., Chen, W.-G. and Wu, Y. (2015). Cloud performance modeling with benchmark evaluation of elastic scaling strategies, *IEEE Transactions on parallel and distributed systems* **27**(1): 130–143.

Islam, S., Keung, J., Lee, K. and Liu, A. (2012). Empirical prediction models for adaptive resource provisioning in the cloud, *Future Generation Computer Systems* **28**(1): 155–162.

Liao, W.-H., Kuai, S.-C. and Leau, Y.-R. (2015). Auto-scaling strategy for amazon web services in cloud computing, *2015 IEEE International Conference on Smart City/SocialCom/SustainCom (SmartCity)*, IEEE, pp. 1059–1064.

Mahmud, A. H., He, Y. and Ren, S. (2015). Bats: Budget-constrained autoscaling for cloud performance optimization, *2015 IEEE 23rd International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems*, IEEE, pp. 232–241.

Nikravesh, A. Y., Ajila, S. A. and Lung, C.-H. (2015). Towards an autonomic auto-scaling prediction system for cloud resource provisioning, *2015 IEEE/ACM 10th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, IEEE, pp. 35–45.

Parekh, A. and Pandi, G. S. (n.d.). Efficient auto-scaling mechanism in the cloud environment using proactive approach.

Qu, C., Calheiros, R. N. and Buyya, R. (2018). Auto-scaling web applications in clouds: A taxonomy and survey, *ACM Computing Surveys (CSUR)* **51**(4): 1–33.

Rahman, S., Ahmed, T., Huynh, M., Tornatore, M. and Mukherjee, B. (2018). Auto-scaling network resources using machine learning to improve qos and reduce cost, *arXiv preprint arXiv:1808.02975* .

Singh, P., Gupta, P., Jyoti, K. and Nayyar, A. (2019). Research on auto-scaling of web applications in cloud: survey, trends and future directions, *Scalable Computing: Practice and Experience* **20**(2): 399–432.

Souza, A. A. D. and Netto, M. A. (2015). Using application data for sla-aware auto-scaling in cloud environments, *2015 IEEE 23rd International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems*, IEEE, pp. 252–255.

Tseng, F.-H., Tsai, M.-S., Tseng, C.-W., Yang, Y.-T., Liu, C.-C. and Chou, L.-D. (2018). A lightweight autoscaling mechanism for fog computing in industrial applications, *IEEE Transactions on Industrial Informatics* **14**(10): 4529–4537.