National College of Ireland

Improving the Auto scaling mechanism in Cloud computing environment using Support Vector regression and Bi-LSTM

MSc Research Project
Cloud Computing

Jackson Peter
Student ID: x20183305

School of Computing
National College of Ireland

Supervisor: Divyaa Manimaran Elango

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Jackson Peter
Student ID:	x20183305
Programme:	Cloud Computing
Year:	2021
Module:	MSc Research Project
Supervisor:	Divyaa Manimaran Elango
Submission Due Date:	31/01/2022
Project Title:	Improving the Auto scaling mechanism in Cloud computing environment using Support Vector regression and Bi-LSTM
Word Count:	5885
Page Count:	19

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.

Signature:	
Date:	31st January 2022

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):	

Improving the Auto scaling mechanism in Cloud computing environment using Support Vector regression and Bi-LSTM

Jackson Peter
x20183305

Abstract

The availability, reliability and on-demand feature of the cloud computing system has attracted many users to the cloud computing platforms, where the resources can be dynamically allocated to the instance as per the workload demand. An efficient auto-scaling mechanism allocates and de-allocates the resources to meet the performance targets in changing workload conditions. Also, it helps to minimize the resources cost as well as making the resource availability on time in order to maintain the quality of service. In this work, I have developed a cloud-based framework using Python language and experimented with 3 different machine learning and deep learning algorithms (Linear regression, Support Vector regression and Bi-directional LSTM) for implementing the auto-scaling mechanism. After the comparative analysis, I have obtained better results using support vector regression and Bi-LSTM algorithms for dynamic and non-linear behaviour in cloud computing environment.

1 Introduction

Over time, the demand for computational resources has escalated due to the shift in requirements and technological advances. With ever-changing technology, individuals and entities require a periodic modernization of resources with the requirements; though the modernization of resources is not at all economically feasible and require an instant approach for everyone. To overcome these hurdles, availability of the computational resources over the air is a boon in today's time. This virtual system is generally known as Cloud Computing. Cloud resources inhibit a solution for extensive computational power requirements providing higher performance and efficient results. This system includes applications such as data processing, storage, and service utility. It acts as a virtual alternative for the requisite resources. In the current era. Cloud Computing plays a crucial role in technological advancement as it provides various utility variants such as Software as a Service (SaaS), Platform as a Service (PaaS), and Infrastructure as a Service (IaaS). Although this computing system depends on the virtual mode of connectivity, there should be an optimized scaling platform for the user requests and reduced network latency to achieve efficient performance.

In the cloud environment, there are two types of scaling policies which are Horizontal and Vertical resource scaling. In Horizontal Scaling, the virtual machines (VM) are allocated or de-allocated based on the demand, whereas in Vertical Scaling, it assigns the

hardware capacity with respect to the demand. Whenever a user opts for the service from the service providers, a pact called Service Level Agreement (SLA) is generated. SLA ensures the users with the minimum serviceability, reliability, performance, pricing, and time which is ultimately called Quality of Service (QoS). The violation of SLA can lead to penalties for the service providers. Therefore, the regulators must periodically monitor the functioning of the services. When the users keep utilization on the least manner, there is an increased chance of SLA violation due to under-provisioning of the resources. Similarly, in case of over-provisioning, the chances of failure in allocation and SLA violation are minimal.

To overcome these challenges, the automatic scaling of resources is implemented. With the auto-scaling approach, the resource demand can be forecasted with previous utilization of the resources. This can efficiently assist the service providers to ensure the QoS while adhering to the SLA. In the reactive approach, the characteristics such as processing capacity, load capacity, and resource request are assessed based on the threshold rule. On the other hand, the proactive approach assesses the previous traffic for the load allocation. These approaches are known as the threshold-based mechanisms. In this research, I have implemented a time-series-based machine learning framework to optimize the auto-scaling technique of the cloud computing resources and further minimized the latency in the cloud system environment using Machine learning and deep learning algorithms. I have compared the conventional models such as Linear regression, Support Vector regression with more-advanced Bi-LSTM. The performance of each auto-scaling model has been measured in terms of total processed load, idle cycles and delayed load.

1.1 Research Question

- How the Machine learning and Deep Learning Based algorithms can be utilized for implementing Auto-scaling Mechanism ?
- Which algorithm efficiently allocates the resources for dynamically changing workload demand and how the performance of each algorithm has been calculated for selection of optimal model ?

2 Literature Review

In this section, I have discussed about the various studies by multiple researchers in Cloud computing and auto-scaling. The section is further divided into Cloud Resource Scheduling, Threshold-Based Auto-Scaling, Pro-Active Scheduling, Machine Learning Framework, Deep Learning Framework and Hybrid Framework.

2.1 Cloud Resource Scheduling

Mehta et al. (2017) mentioned an efficient method for resource scheduling in the cloud computing system. As resource scheduling performs a crucial role in the enhanced performance of the cloud resources, an appropriate prediction is necessary during the scheduling. Efficient resource scheduling can increase the computing capability of the resource and thereby control the cost. Therefore, this paper utilized the Hidden Markov Model

(HMM) to efficiently dedicate the available resources to the users on-demand. This proposed model would recognize and keep track of unexploited resources. It would then further classify the counterpart based on workload scenarios such as Less, Medium, and High. Relying on the demand, the appropriate scheduling algorithm is implemented on the model under a dedicated workload scenario. Similarly, Islam and Buyya (2018) proposed a method for resource management and scheduling of big data programs in cloud computing environments. There are various modules of Big Data processing frameworks which are Batch, Stream and Hybrid. The big data tools utilized and discussed in this study are Apache Hadoop, Apache Spark, Apache Kafka and many more. For the resource managing tool, the programs used are Google Kubernetes, Apache Hadoop Yarn, etc.

Verma et al. (2016) presented an approach for the dynamic resource prediction and allocation in multi-tenant utility. Due to the increasing demand for a cloud computing system, it is mandatory to have an efficient framework for cloud resource management. Through this system, the users can upscale or downscale the resource anytime on the demand. Furthermore, it also apprehended the latency in the service. Also, the researchers in this study implemented a best-fit heuristic model to match and allocate the Virtual Machines (VM) to the user's physical machine host. Ultimately, this paper showed the performance of the robust approach for this framework. Singh et al. (2019) researched the auto-scaling approach in the cloud computing approach for web programs. Due to the emerging technology of cloud computing, the web application has been considered for the proposed system. The author of this paper surveyed the challenges that occurred thoroughly and mapped the trend in cloud computing environments for web programs. In the final take, the drawback of the systems were discussed and the research areas for future studies were assessed. On the other hand, Aslanpour et al. (2017) also proposed a cost-aware approach for the web programs in the cloud computing environment. The author provided a mechanism to limit the expense through a strategic approach. Here, the MAPE (Monitoring, Analysis, Planning and Execution) loop approach is implemented. Furthermore, the model had an aware selection of cloud characteristics and the extra virtual machine for the utilization in the resource pool. The proposed model inhibited a reduced cost of renting by 7% and thereby enhancing the performance of SLA adherence in the service.

2.2 Threshold-Based Auto-Scaling

Patel et al. (2016) explored an advanced approach for adaptive threshold-based dynamic resource provisioning in cloud systems in the research. This study primarily focused to overcome the conventional approach of resource scheduling which allocates the virtual machines to the users over their previous demand. However, the pre-allocation of the virtual machines can lead to either under-utilization or limitation to the demands of the user. Therefore, the author in this study, proposed a novel approach by allocating the resources in real-time and adjusting to the demand dynamically. Oladoja et al. (2021) also proposed a threshold-based resource allocation approach in Cloud Computing Environment. Through an efficient prediction, the cloud resources are being allocated to the users in contrast to the requested demand. But to enhance the performance of the resources and combat the downtime, an appropriate provisioning algorithm shall be utilized. Therefore, the author had explored the threshold-based tournament selection

probability for virtual machine provisioning. The algorithm utilized in these models was the Median-based optimized Max-Min algorithm. This approach showcased an improved performance to the model.

Chen and Bahsoon (2015) provides an additional self-adaptive methodology that can accomplish automated amplification in a cloud setting. Here, the approach succeeds with the lower pricing ratio, hence, reducing the incidence of numerous difficulties. In this study, colony techniques are employed efficiently to regulate and reduce the occurrence of recurrent trade-off situations. The employment of such techniques results in a precise and high transfer conclusion. The studies undertaken by Evangelidis et al. (2018) are an example of a common type of investigation effort that is centred on executing the auto-scaling procedure utilizing Bayesian networks. In this technique, the cloud-based autonomous amplification strategies are addressed briefly. Amazon EC2 and Microsoft Azure were used to test the approach. Another study in the identical topic took a distinct method relying on auto-scaling performed with threshold-based techniques. Developed on the exposition of cloud architectures is obtained as a primary outcome and deeply analyzed couple of approaches for such auto-scaling techniques that were already suggested.

2.3 Proactive Scheduling

Kaur and Kaur (2017), in the study, surveyed the proactive scheduling approach in a cloud setting. With the increased demand of cloud computing resources, the performance of this system must also be regulated. The most common reason for the downtime of the computational resources is due to failure of the resource scheduling algorithm. Therefore, an approach proposed of fault aware pattern aligning the autonomous provisioning of the computing resources in the cloud system environment. In these models, algorithms such as Round Robin Algorithm, Service Level Agreement (SLA), Facilitated Application Specification Technique (FAST) were implemented. This model was also evaluated in comparison to the traditional approach, where the latter showed an improved performance. Golshani and Ashtiani (2021) conducted investigation to acquire the tools on spotting the cloud system, the approach of adaptive amplification is used here. In this study, the problems were seen as a sequenced approach. To consider the various requirements and demands of the consumers, a framework of autonomous decision development is also used there. Convolution neural networks (CNN) are used to evaluate and forecast the non-linear response of demand. The findings of this study also shown a four percent enhancement in effectiveness and precision. Likewise, Tang et al. (2015) research takes that very identical method whenever it pertains to the procedure of auto-scaling. It wss performed in the virtual architecture utilising pre-trained techniques for increased consistency and reliability. The SRSA approach wss employed well here. In this procedure, the researcher additionally employed the Markov chain approach. The findings were superior to those obtained using threshold-based techniques. The researcher demonstrates the various effects associated with the systematic methodology and how helpful it would be in the cloud platform.

2.4 Machine Learning Framework

In the study, Srirama et al. (2020) suggested a heuristic-based paradigm of auto-scaling. With the perspective of the resource requirements, the supplied systems of technique transmit the indicated programs on the best-fit light modules with a minimal amount of setup effort. An additional key concern in the conscious cloud environment wss the commencement influence, which would be addressed in the suggested approach by applying a benchmark structure for minimizing transmission time and program charges. Furthermore, a distinct technique was meant for delivering programmes to a small number of physical devices while making efficient use of computational resources. Verma and Bala (2021) provided the same methods used in the development of IoT applications. The researchers likewise applied the very identical auto-scaling strategy to satisfy the demands of the clients. Some of the parameters, as well as the necessity for Quality of Service (QoS) norms, are met in those. The technique of dynamic provisioning was indeed a critical, and most significant feature of data frameworks that pledged and reloaded the allotted potent resource in response to the density of requests.

Zhang et al. (2018) proposed a machine learning-based framework for resource allocation during cloud computing resource auction. The paper stated the existing methodologies utilized such as Polynomial Time Approximate Scheme (PTAS) and heuristic algorithms had some major fallacies. These algorithms provided weaker computational performance with unsatisfactory efficiency and accuracy. Therefore, the author implemented multiple machine learning algorithms for resource allocation and de-allocation. Two machine learning regression algorithms were utilized, namely Linear Regression and Logistic Regression. In the primary phase, the model was trained with a minimal sample batch which was further increased. The novel approach was introduced but the performance evaluation among the conventional and novel approaches was not assessed in this study. In another paper, Li (2017) surveyed the cloud computing system provisioning algorithms relying on the machine learning approach. Various factors of cloud computing including the advantages and disadvantages were discussed. The main aim was to incorporate an appropriate resource scheduling algorithm with efficient load balancing without compromising the performance of each system. Several algorithms were shown and experimented on resource scheduling algorithms utilizing the machine learning frameworks. In the final take, the paper exhibited that the implementation of such algorithms will develop an inefficient performance for the cause and an accurate approach required to overcome the limitations.

2.5 Deep Learning Framework

Ye et al. (2018) proposed a novel approach for resource scheduling utilizing the deep reinforcement learning (DRL) framework. Based on the previous conventional approach of DRL, an online resource scheduling utilizing DRL was implemented and it was called as Deep RM 2. Furthermore, the identical resource scheduling algorithm offline was known as Deep RM OFF. After the implementation of these algorithms, the approaches were evaluated among the conventional ones as well. The proposed model showed a high-paced performance and accurate resource provisioning when compared in contrast to some metrics such as average downtime period, task completion period, and rewards. Che et al. (2020) also proposed a deep reinforcement learning approach to regulate the task rescheduling in the data centre. As the increasing load on the data centre was on the

rise due to higher online transformation, the existing task scheduling framework utilized a conventional heuristic algorithm was unable to handle the current time demand and load. Therefore, a novel approach to overcome these challenges and to fulfil the demand, must be introduced. The researchers in this study proposed a unique model implementing DRL which could optimize the resource utilization efficiency with an appropriate task scheduling algorithm. This paper revealed that the proposed algorithm performed efficiently than the traditional algorithms when assessed based on the average delay time of the tasks, task distribution and congestion. Sun and Li (2020) also proposed a DRL approach with dynamic resource allocation for Next Generation Cellular Systems. In this paper, a novel algorithm called contiguous frequency-domain resource allocation (FDRA) which relied on DRL was implemented. The approach would selectively assign users for each resource on the system called resource blocks (RBs). The approach for the scheduler that implemented was Markov Decision Process. With DRL methods, it recognized the users for the resource allocation and determines the step-wise allocation of the RBs. Various parameters of the system were also evaluated. The proposed algorithms when evaluated had outperformed the conventional counterparts and could easily overcome the existing challenges. Over the efficient performance, this model had drastically lowered computational complexity. The author also suggested the future scope of the study by considering various other features as well.

2.6 Hybrid Framework

Guo et al. (2018) dissertation presents a further integrated investigation framework. The primary goal of this study was to boost the frequency of terminal servers while decreasing the volume of additional expenditures associated. The amount of virtualization adaptability frequently resulted in significant inefficiency and increased costs, especially for programs with rapidly variable demands. The researcher proposed a minimalist approach to deal with increasing pragmatic flexibility for cloud computing within scientific work given there. Several hybrid approaches were sometimes used to test the effectiveness of the auto-scaling procedure. Biswas et al. (2017) research gave a detailed road-map underlying the blended resource provisioning approach. Throughout this study paper, the researcher combined and employed both proactive and reactive amplification strategies to get the optimal auto-scaling mechanism. The primary objective of the investigation was to lower greater costs while meeting all the demands of the client. Experiments were carried out to test its feasibility. Utilizing the strategy, respectively on request programs and SLA was maintained under control. The work by Singh et al. (2021) proposed a robust hybrid auto-scaling (RHAS) technique for web programs in cloud computing. Here, both the reactive and proactive approach were utilized based on workload prediction of the demand. In this model, the time-series forecasting approach had been implemented to predict future workload from the users. Furthermore, in the proposed model a threshold-based model and a queue model were incorporated. This proposed framework was validated in two real-time applications and further achieved a reduction by 14% in cost and some other fallacies.

The comparative analysis for the various studies by different researchers is shown in Table 1.

Paper Title	Publish Year	Method	Advantages	Future Scope / Disadvantages
A Threshold-based Tournament Resource Allocation in Cloud Computing Environment	2021	The algorithm utilized in these models was the Median-based optimized Max-Min algorithm	Enhanced the performance of the resources and combat the downtime	More efficient algorithms can be utilized
Proactive Autoscaling for Cloud Environments using Temporal Convolutional Neural Networks	2021	Convolutional neural networks (CNN) were used to evaluate and forecast the nonlinear response of demand	The findings of this study showed a four percent enhancement in effectiveness and precision	No disadvantages found
Machine learning based resource allocation of cloud computing in auction	2018	Two machine learning regression algorithms were utilized namely Linear Regression and Logistic Regression	The novel approach was introduced	The performance evaluation among the conventional and novel approaches was not assessed in this study
Deep-Reinforcement-Learning-Based Scheduling with Contiguous Resource Allocation for Next-Generation Cellular Systems	2020	A novel algorithm called contiguous frequency-domain resource allocation (FDRA) which relied on DRL was implemented in the paper	The proposed algorithms when evaluated outperformed the conventional algorithms and could easily overcome the existing challenges	The author also suggested the future scope of the study by considering various other features as well thereby enhancing the performance

Table 1: Comparison of Different Works for this Study

3 Methodology

As the user requests in the cloud environment are generated dynamically, allocating the resources in the dynamic fashion in cloud computing environment is a challenging task. There are certain set of rule-based methods that has been implemented by many authors and researchers, which were inefficient to handle the dynamic behaviour of allocation. In order to solve such challenges, in this research, I have implemented Machine learning and Deep learning based auto scaling mechanisms, which can allocate and de-allocate the resources dynamically with more efficiency. The Proposed framework of auto scaling consists of many components, the working mechanism of overall framework is shown in Figure 1.

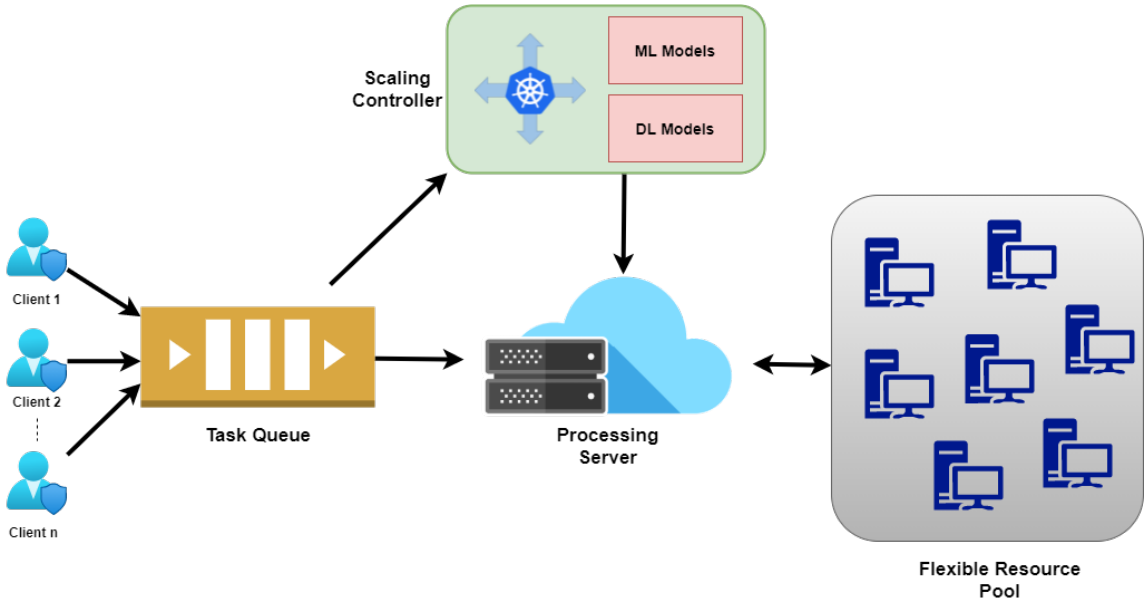

Figure 1: Proposed Framework for Auto-Scaling

3.1 Client/Task Generation

The main responsibility of the client was to generate the tasks. Each client can generate any number of tasks at any point of time. In order to simulate the real-world scenario, I have assigned a random load with each task, and the load generated on system by each task was different from one another as each task required the different time for execution depending on the load. There can be sometime in a day, when the number of requests are at peak in the cloud server, also there is a time when requests are comparatively less as compared to the usual scenario. Considering these aspects, I had generated the random load in the system using sin wave function. These generated tasks, were later sent to task queue for processing.

3.2 Task Queue

All the generated tasks required a temporary storage before being processed by the cloud server, the task queue would fulfil this purpose. Task queue acts like a normal queue of data, it sends the tasks to processing server in First in First out (FIFO) manner. Task

queue also send the information about current number of tasks to the scaling controller, which helps the scaling controller to analyze the information for better prediction.

3.3 Processing Server

Processing Server is the main entity of the cloud computing environment, which was responsible for processing all the generated task available in the task Queue. The computing capabilities of processing server is fixed and is directly connected with Flexible resource pool, which provides the auto scaling capability. The decision of up-scaling and down scaling the resources will be taken by scaling controller.

3.4 Flexible Resource Pool

Flexible Resource pool is the place from where processing server can acquire or release the computing capabilities. This component provides the auto-scaling capabilities to the cloud server, where the amount of resource required to process the upcoming tasks will be predicted by the scaling controller.

3.5 Scaling Controller

Scaling Controller is the most crucial component of proposed auto-scaling system. It analyses the current activities of the system and based on the number of tasks available in the task queue, and predicts the amount of resources required. After predicting the resource requirements, the scaling controller sends the details to processing server, accordingly the processing server upgrade or downgrade its computing capabilities. Scaling controller is a place where the Machine learning and deep learning models are trained based on the historical load data to predict the next cycles of traffic. In this work, I have implemented two machine learning and one deep learning regression models that are Linear regression, Support Vector regression and Bi-directional LSTM respectively.

4 Design Specification

As the load was calculated every cycle, the obtained load was in continuous number, which will be solved using regression analysis with time series prediction. In this work, I have implemented the Linear regression and Support Vector regression algorithm, which are machine learning algorithms. On the other hand, Bi-directional LSTM algorithm was also utilized and it fell under the deep learning category.

4.1 Linear Regression

Linear regression algorithm is the most basic and common algorithm for regression analysis. It is mainly used to find the relationship between the two continuous variables. Among them, one can be independent or dependent variable. In this process of linear regression, I tried to fit the linear model with coefficient by minimizing sum of square between the actual and the predicted values. The equation of the linear regression can be derived by the following formula as shown in Figure 2.

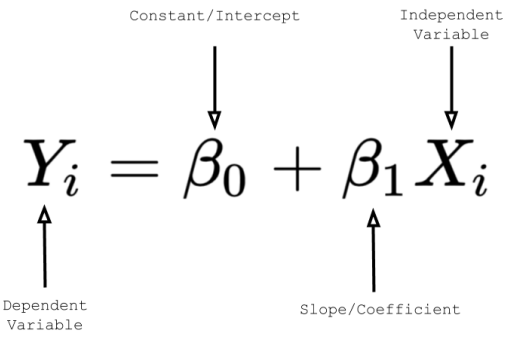

Figure 2: Simple Linear Regression Formula

4.2 Support Vector Regression

Support vector machine is the most popular algorithm for classification problem. However, it can also be utilized for regression problem. It works on the concept of maximum margin, where margin of tolerance (epsilon) was set to the approximated value for prediction. SVR algorithm identifies the non-linearity in the data and generates the better prediction. In my work, the algorithm had been utilized to predict the future load on the system based on the historical load data and number of tasks available in task queue. After regression analysis, the output of support vector regression can be represented as follow.

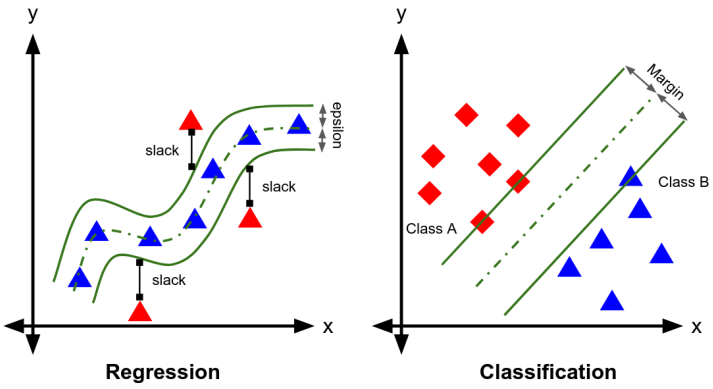

Figure 3: Support Vector Regressor

4.3 Bi-LSTM

When the Learning problem is sequential, the LSTM and Bi-LSTM based on the recurrent neural network architecture are highly utilized. Bi-LSTM architecture is also called Bi-directional LSTM where the one LSTM model take the input in forward direction and other LSTM takes the input in backward direction. In this way, the learning algorithm is fed with data from starting to end and end to beginning, which help the Bi-LSTM model to remember more information and thus helps in better prediction. As the generated load

data was sequential, in this work, Bi-LSTM was utilized in order to obtain the better results. The architecture of Bi-LSTM is shown in Figure 4.

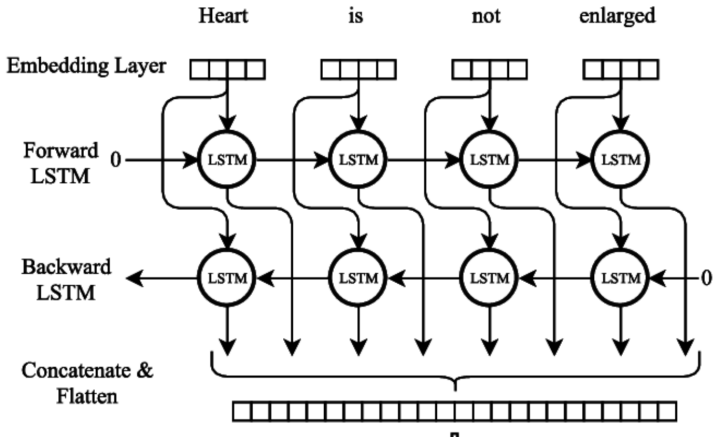

Figure 4: Bi-LSTM Architecture

5 Implementations

The implementation of proposed framework for auto-scaling had been developed using Python programming language. Along with the python, there are several python-libraries had been used to develop the system. In order to generate the dataset, the sin wave function had been utilized using the pandas and the math function. The concept of multi-processing was utilized where each process will utilize a separate core of CPU to process the multiple tasks. Currently 5000 dataset sample with random load had been employed over different cycles. In order to visualize the results, the matplotlib library has been used to generate the line graphs and bar graphs. To perform the matrix calculation, numpy library has been used. For training the machine learning and deep learning models, Scikit-learn and tensorflow libraries has been utilized. While implementing the Bi-LSTM model, the tanh was employed as an activation function and reLu for the dense layer. Discussing about the overall process, the various clients will generate the load on the system, where the task temporarily will be stored in the task queue, which will be fetched by processing server for execution. The processing server can upgrade or downgrade their capabilities using flexible resource pool. Scaling controller predicts the future load based on the selected model. Here, I have trained 3 different model such as Linear regression, Support vector regression and Bi-LSTM for load prediction. As the current system was implemented using python language, this can run on any operating system. However, Linux based operating system is more preferred. The system with following specification is required to implement the system.

- Operating system: Ubuntu(20.04)
- Main Memory (RAM): 8GB
- Hard disk : 10 GB
- Programming Language: Python
- Libraries : Numpy, Matplotlib, multi-processing, sklearn, tensorflow, pandas.

6 Evaluation

The main task of the scaling controller is to predict the requirement of load for N+1 seconds or cycles and provide the instructions to the processing server to scale-up or scale-down the capacity accordingly. There were 3 algorithms that had been implemented in to the scaling controller, those were Linear regression (LR), Support Vector Regressor (SVR) and Bi-LSTM. Each of the algorithm will be evaluated in the proposed framework based on generated load and processed load, idle cycles of machine, total amount of load processed by the system using a specific algorithm and delayed task. Based on these metrics, I have analyzed the optimal algorithm for auto-scaling mechanism in the cloud computing environment. The model with maximum processed load, minimum delayed load and with minimum idle processing cycles will be considered as an optimal model for auto scaling mechanism.

6.1 Experiment 1 / Evaluation of Linear Regression

In this experiment, I have implemented the linear regression algorithm for predicting the future load on the processing server of cloud where the model had trained over the historically generated load. Based on the historical analysis and current number of tasks in the queue, it makes the prediction for future tasks and arranges the resource accordingly. After running the cloud system for 90 cycles, I have calculated the generated load, processed load and idle time for each cycle. Since I used sin wave for data generation, there were spikes with upward and downward movement. With the load generation on each cycle, the scaling controller responsibility was to allocate similar processing capabilities. The generated graph after running the system for 90 cycles, is shown in Figure 5.

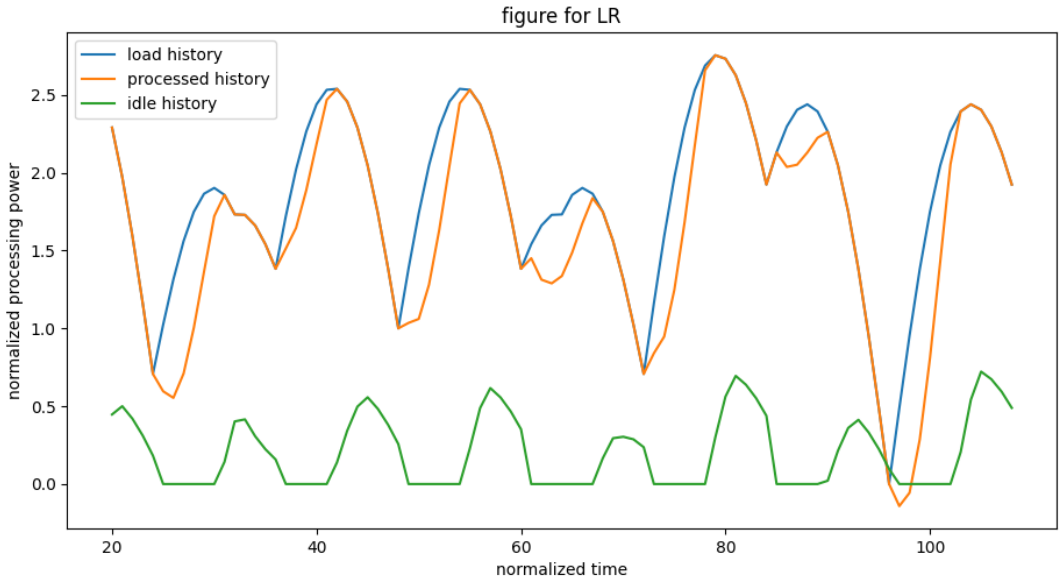

Figure 5: Line graph of Generated Load, Processed Load and Idle Time using Linear Regression

On analysing the graph for linear regression, it has been found that When there was a spike with downward movement, there was a huge difference in the actual load and processed load on the system. The graph represents that algorithm scaled-up the

resources very well, but it causes over allocation when there was decreased amount of load. Therefore, in the graph also it has been observed that when the load decreased due to over-allocation of resources, the idle time also increased considerably as shown in Figure 5.

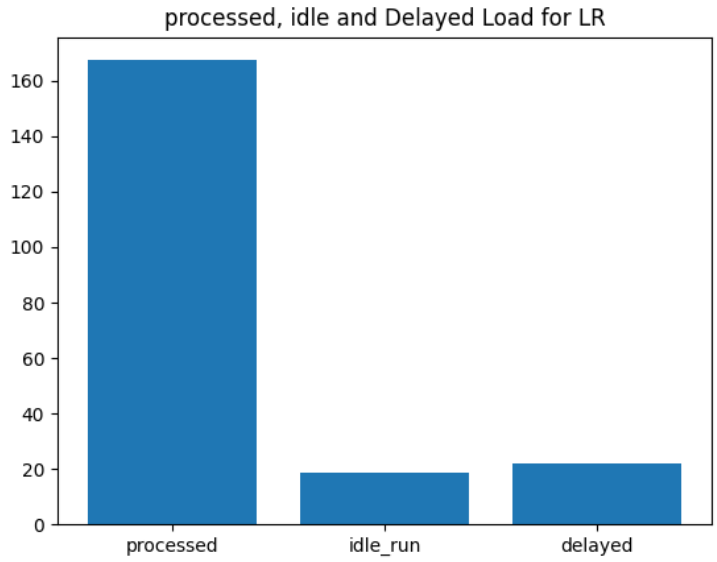

Figure 6: Total Processed, Idle and delayed Load using Linear regression

After running the cloud system over the period of 90 cycles, I had calculated the processed load, idle run and delayed load using linear regression. On analysing the graph as shown in Figure 6, it had been observed that total load processed by the system was 167.44. Meanwhile, the idle cycle were obtained around was 18.60 and delayed load was 21.76. After calculating the results of other algorithms, I could compare the results obtained using linear regression algorithm and would discuss about the analysis in discussion chapter.

6.2 Experiment 2 / Evaluation of Support Vector Regression

In this experiment, I have used the support vector regression (SVR) algorithm for predicting the load on each cycle. The same load had been generated like linear regression in order to compare the results on same benchmark and the future load was predicted for each cycle using support vector regression algorithm. After running the proposed framework for 90 cycles, the generated load, processed load and idle run were calculated for each cycle as shown in Figure 7.

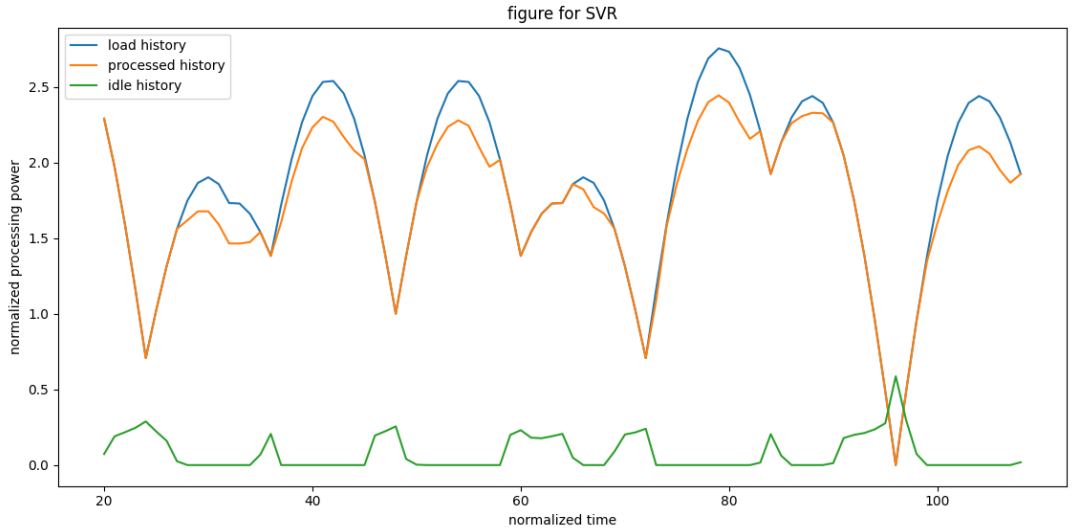

Figure 7: Line graph for Generated Load, Processed Load and Idle Time using SVR

From the following graph, it was observed that the resources were under-allocated at peak, when momentum of graph was very high. When the load on the graph decreased, on observing the graph carefully, over-allocation of resources also had been observed. Also, in terms of idle time in comparison with linear regression, it was analysed as SVR represents the minimum number of idle cycles as compared to the Linear regression algorithm. The minimum number of idle cycle represented the efficient utilization of resources.

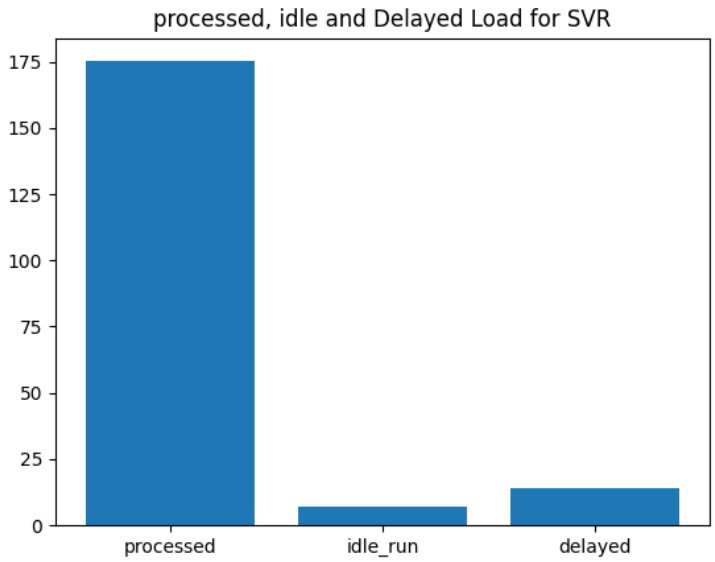

Figure 8: Total Processed, Idle and delayed Load using Support Vector Regression

I have calculated the processed load, idle run and delayed load for Support vector regression algorithm over 90 cycles as shown in Figure 8. System processed the load of 175.17 using SVR, also the idle runs were found to be 6.848 and delayed load has been

observed as 14.04. On comparing the results with linear regression it has been found that processed load by SVR was comparatively high and idle cycles were 3 time lesser in comparison, and also observed minimum delayed load between cycles. So definitely in every aspect, SVR architecture outperformed the linear regression algorithm.

6.3 Experiment 3 / Evaluation of Bi-LSTM

Bi-LSTM was the most complex neural network architecture and was found to be better for predicting the time-series analysis. Predicting the load on every cycle was also considered as a time-series problem, and therefore, Bi-LSTM had been used in my proposed architecture. The results of Bi-LSTM algorithm were analyzed by generating the same graph and results such as the metrics such as generated load, processed load and idle runs were calculated over every cycle. In terms of prediction, the results of Bi-LSTM were found to be more precise and accurate. However, to some extent, Bi-LSTM had also exhibited under and over-provisioning of the resources. The number of idle cycles were also found to be minimal. The line graph for Generated load, processed load and idle time using Bi-LSTM algorithm over every cycle is shown in Figure 9

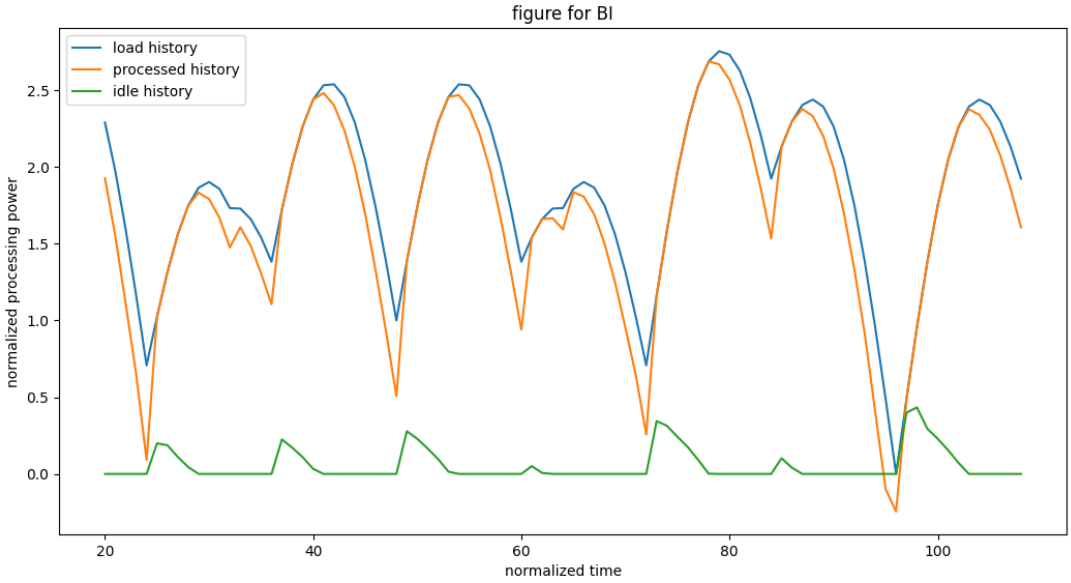

Figure 9: Line graph of Generated Load, Processed Load and Idle Time using Bi-LSTM

However, from the results of line graph, for every cycle it was very difficult to analyze the performance of the algorithm. Therefore, I have also calculated the total processed load, idle cycles and delayed load using Bi-LSTM algorithm, which are shown in Figure 10.

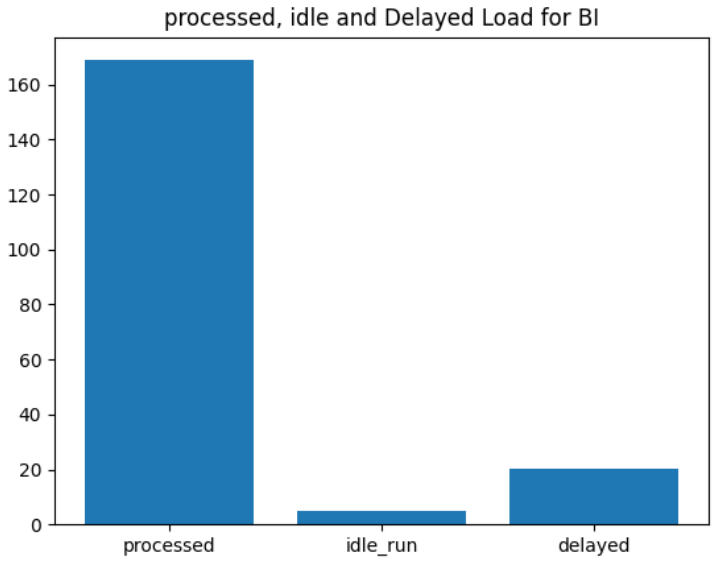

Figure 10: Total Processed, Idle and delayed Load using Bi-LSTM

After calculating these metrics, it has been found that total load processed using Bi-LSTM algorithm was 168.83, where the number of idle runs was 4.886 and delayed load was 20.38. The total number of idle-cycles obtained using Bi-LSTM algorithm was minimum as compared to the SVR and Linear regression algorithms. However, total load processed by Bi-LSTM found to be higher than linear regression algorithm but less than the support vector regression algorithm. Moreover, in terms of delayed load, the Support vector method had achieved the minimum delay in processing the load, followed by Bi-LSTM and Linear regression.

As the final phase of evaluation, the different attributes such as processed load, delayed load and idle cycles are compared against each other and plotted in tabular form as illustrated in Figure 11. From the generated output values, the performance of each of the algorithms can be evaluated. Linear Regression algorithm was found to be the least efficient one with less processed load and significantly more number of idle cycles with delay. Support Vector Regression model exhibited better results in comparison to Linear Regression as it processed more incoming load with very less amount of delays. The last algorithm in this research was Bi-LSTM, performed better than Linear Regression in all aspects with better processed load, less idle cycles and delay. However, Bi-LSTM processed less load than SVR algorithm despite the less idle cycles.

	LR	SVR	Bi-LSTM
Total Processed	167.44	175.17	168.83
Idle Cycle	18.60	6.848	4.886
Delayed	21.76	14.04	20.38

Figure 11: LR, SVR and Bi-LSTM - Performance Comparison

6.4 Discussion

In order to identify the optimal model for auto-scaling mechanism, in the proposed work, I had compared 3 different machine learning and deep learning based algorithms in terms of processed load, idle run and delayed load. Also, the generated load and resource allocation were observed over each cycle. After analyzing the results, it was observed as the linear regression algorithm scaled-up the resources very well but faced issues in prediction when the sudden fall in load, which caused the over allocation of resources and resulted in poor idle times. On the other hand, to some extent, over provisioning and under-provisioning had been observed in both support vector regression and Bi-LSTM models. However, while comparing the total processed load by the system, SVR had processed high amount of load with respect to Bi-LSTM. Contrast to this, Bi-LSTM framework exhibited the minimum number of idle cycles among all the algorithms. On the other hand, comparing the delayed load, the highest delayed load was obtained using linear regression, followed by Bi-SLTM and fianlly Support vector regression. Also. in the case of delayed load, SVR outperformed the other two algorithms with better results on each cycle. Based on the overall analysis, I can mention that Bi-LSTM algorithm utilized the resources more efficiently as the total idle run using Bi-LSTM was minimum. But the delayed load of Bi-LSTM was quite higher due to which the total number of processed load was comparatively less when plotted against support vector regression.

7 Conclusion

In this research, a framework had been proposed for auto-scaling mechanism which was inspired from the real-world cloud computing platform. The main objective of this research was to identify the most optimal algorithm for auto-scaling mechanism, which could enhance the productivity of resources, and could reduce the over-head and latency on the system. Therefore, I had experimented with the 3 different algorithms based on the machine learning and deep learning architectures. The algorithms were Linear regression, Support vector regression and Bi-LSTM, which had been compared with respect to the different metrics and against outcomes. After analyzing the graphs and results of different experimentation, it can be concluded that as the minimum number of idle cycles were achieved using Bi-LSTM algorithm, and it was more convenient for resource utilization. However, due to the complex architecture of Bi-LSTM algorithm, the predictions

get delayed and which resulted in process delays on the load. On the other hand, SVR was a simple architecture and sometimes, it under-allocated the resources at peak load. However, the total load processed by SVR was quite higher unlike the linear regression and Bi-LSTM algorithms. Also, the delayed load were minimum. Thus, I can suggest that Bi-LSTM performs better in terms of resources utilization and in terms of overall analysis, like total processed load, delayed load, the support vector regression will be an optimal choice. As the behaviour of the cloud is dynamic and non-linear in nature, the linear regression algorithm did not performs well for auto-scaling mechanism. My current research utilized the synthetic data generation of load for predicting the resource requirements. However, in the future work the traffic data and resource information on the cloud computing platform can be captured and can be used for prediction. Identifying the optimal algorithm for auto-scaling is still an open area of research where multiple algorithms can be experimented and results can be compared with each other. The behaviour of prediction for every algorithm might change on different test case scenarios. Therefore, cloud providers should identify their needs and appropriate algorithms should be deployed for the desired results.

References

Aslanpour, M. S., Ghobaei-Arani, M. and Toosi, A. (2017). Auto-scaling web applications in clouds: A cost-aware approach, *Journal of Network and Computer Applications* **95**.

Biswas, A., Majumdar, S., Nandy, B. and El-Haraki, A. (2017). A hybrid auto-scaling technique for clouds processing applications with service level agreements, *Journal of Cloud Computing* **6**.

Che, H., Bai, Z., Zuo, R. and Li, H. (2020). A deep reinforcement learning approach to the optimization of data center task scheduling, *Complexity* **2020**: 1–12.

Chen, T. and Bahsoon, R. (2015). Self-adaptive trade-off decision making for autoscaling cloud-based services, *IEEE Transactions on Services Computing* **10**: 1–1.

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

Golshani, E. and Ashtiani, M. (2021). Proactive auto-scaling for cloud environments using temporal convolutional neural networks, *Journal of Parallel and Distributed Computing* **154**.

Guo, Y., Stolyar, A. and Walid, A. (2018). Online vm auto-scaling algorithms for application hosting in a cloud, *IEEE Transactions on Cloud Computing* **PP**: 1–1.

Islam, M. T. and Buyya, R. (2018). Resource management and scheduling for big data applications in cloud computing environments.

Kaur, R. and Kaur, G. (2017). Proactive scheduling in cloud computing, *Bulletin of Electrical Engineering and Informatics* **6**: 174–180.

Li, B. (2017). Research and analysis of resource scheduling algorithm in cloud computing environment, *Agro Food Industry Hi-Tech* **28**: 3192–3196.

Mehta, H., Prasad, V. and Bhavsar, M. (2017). Efficient resource scheduling in cloud computing, *IJARCS*.

Oladoja, I., Adewale, O., Oluwadare, S. and Oyekanmi, E. (2021). A threshold-based tournament resource allocation in cloud computing environment, *Asian Journal of Research in Computer Science* pp. 1–13.

Patel, P., Scholar, A., Dwivedi, A., Richariya, V. and Sabri, M. (2016). Adaptive threshold based dynamic resource provisioning in cloud environment, *SSRN Electronic Journal* **5**: 15–21.

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**: 399–432.

Singh, P., Kaur, A., Gupta, P., Gill, S. S. and Jyoti, K. (2021). Rhas: robust hybrid auto-scaling for web applications in cloud computing, *Cluster Computing* **24**.

Srirama, S., Adhikari, M. and Paul, S. (2020). Application deployment using containers with auto-scaling for microservices in cloud environment, *Journal of Network and Computer Applications* **160**: 102629.

Sun, S. and Li, X. (2020). Deep-reinforcement-learning-based scheduling with contiguous resource allocation for next-generation cellular systems.

Tang, P., Li, F., Zhou, W., Hu, W. and Yang, L. (2015). Efficient auto-scaling approach in the telco cloud using self-learning algorithm, *GLOBECOM*, pp. 1–6.

Verma, M., Gangadharan, G. R., Narendra, N., Vadlamani, R., Inamdar, V., Ramachandran, L., Calheiros, R. and Buyya, R. (2016). Dynamic resource demand prediction and allocation in multi-tenant service clouds, *Concurrency and Computation: Practice and Experience* **28**.

Verma, S. and Bala, A. (2021). Auto-scaling techniques for iot-based cloud applications: a review, *Cluster Computing* **24**.

Ye, Y., Ren, X., Wang, J., Xu, L., Guo, W., Huang, W. and Tian, W. (2018). A new approach for resource scheduling with deep reinforcement learning.

Zhang, J., Xie, N., Zhang, X., Yue, K., Li, W. and Kumar, D. (2018). Machine learning based resource allocation of cloud computing in auction, *Computers, Materials and Continua* **56**: 123–135.