Tuijin Jishu/Journal of Propulsion Technology
ISSN: 1001-4055
Vol. 44 No. 2 (2023)

A Review on Cloud Computing Elasticity

Hirenkumar R. Patel[1], Dr. Tejas P. Patalia[2]

1. Research Scholar, Computer/IT Engineering, Gujarat Technological University, Chandkheda, Ahmedabad.
2. Head & Professor, Computer Engineering Department, VVP Engineering College, Rajkot, Gujarat, India.

Abstract:- Cloud computing is an internet-based computing environment where computing resources (hardware, software) are managed, pooled, and provided on demand by the cloud service providers to cloud service users. Users may rent resources with fixed capacity by considering the application's peak workload, but the application workload may vary. The workload variation may depend on many situations like time of the day, day of the week, etc., So resources may be underutilized when the workload is lower than the peak workload and overutilized when the workload is higher than the peak workload so some strategies are required to manage resources that can add or remove resource capacity as per the variation in workload. For managing cloud resource capacity elasticity plays a vital role. Elasticity is the dynamic property of the cloud and is used for the provisioning and de-provisioning of cloud resources to map between workload and cloud resources. This study reviews the elasticity for "VM only", "Container only" and "VM and containers together".

Keywords*: Cloud Computing, Resource Management, Autoscaling, Elasticity, Virtual Machine, Containers*

1. Introduction

Cloud computing can be defined as an Internet-based computing environment where computing resources are managed, pooled, and provided to cloud users on-demand basis[1] . Cloud computing environment has at most two parties-cloud service user and cloud service provider. The other type of users that accesses an application hosted in the cloud are the clients of the cloud service user. In the cloud computing environment, it is considered that unlimited resource pool is available, and it is available on-demand and cloud service user has to pay as per the resource usage.

According to the service provided by the cloud. it can be categorized into three service models, Infrastructures as a Service (IaaS), Platforms as a Service (PaaS), Software as a Service (SaaS) [2]. Along with the Iaas, Pass, and Saas, an earlier new type of service introduced is known as Container as a Service (CaaS).one such example of a container management system is Docker. Docker is a tool, and using it developers can define containers for applications. The position of CaaS is between Iaas and Paas. IaaS Provides Infrastructure related services, and PaaS provides an application development environment, so CaaS layer was basically missing layer previously which attaches both Iaas and PaaS [3].

Service level agreement (SLA) can be defined as a mutual contract between the service provider and consumer, which determines the agreed service level objective (SLO) [4]. Service level agreement is very important to pursue a profitable business relationship between the service user and the service provider. Common performance SLOs include availability, response time, capacity, etc[1].

In the cloud computing environment, a service provider's goal is to maximize profit and to achieve better resource utilization. The service user's goal is to minimize cost and maximize performance. Elasticity plays an important role and helps for better resource utilization, in minimizing the service level agreement (SLA) violations.

Elasticity is the dynamic property of the cloud which deals with provisioning, de-provisioning of a resource in an automatic manner in such a way that it closely matches the resources with the incoming workload in an automatic manner [5].

This paper discusses the classification of elasticity, work done on elasticity by considering "VM only", "Container only" and "VM and containers together" scenarios.

2. Resource Management

Cloud computing environment is a highly dynamic environment and resource management is a challenging while managing SLAs. Elasticity can be used for managing cloud resources to achieve SLA objectives.

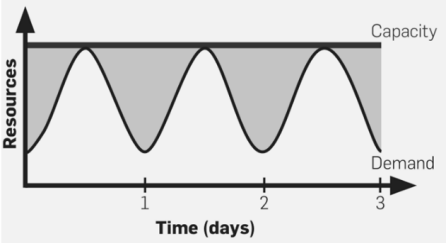

Figure 1. Provisioning for peak load[6]

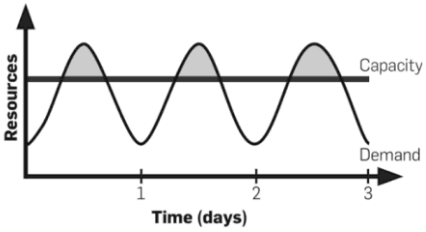

Figure 2. Under provisioning[6]

Figure 1 and Figure 2 show the scenario of using a static resource capacity when the variable workload is given to the server. Figure 1 shows the case when the resource is selected of fixed capacity by considering peak load. In this case, resources not fully utilized all the time, and most of the time, it stays underutilized. Underutilization is shown with the shaded area in figure. Figure 2 shows the case when resource capacity is selected based on the average workload. In this case, resources are over-utilized, and customer demand cannot be served as shown with the shaded area in the figure.
Instead of using static resource capacity, Cloud service providers may offer dynamic scaling mechanism to its users, so according to the incoming workload resources capacity can be changed by the scale in or scale out the resources [5].

Elasticity plays an important role in dynamic resource management. The next section contains a detailed discussion of the elasticity.

3. Elasticity

Elasticity is a cloud computing dynamic property. It allows scaling of the system in or out on-demand in an operational system. Elasticity is the feature available for clients to quickly demand, receive, and release resources as needed [7]. Some other related and similar terms are scalability, efficiency, and auto-scaling.

Scalability can be defined as the ability to handle the additional load by adding extra capacity using extra resources. for elasticity, scalability is the first requirement. Scalability does not consider time parameters like how fast and often and at what granularity scaling actions can be taken. So, it can be written that scalability is the way of increasing the capacity of the system, and it handles the increase in workload by using additional resources. It is not time bounded; hence as elasticity, it does not care about how actual resource demands match with the provided resources at a particular point in time [5].

Another term efficiency is expressed as the number of resources used for processing a workload. As in the case of elasticity, efficiency is the best utilization of resources. Normally better elasticity gains higher efficiency [5].

Auto-scaling and elasticity used with the same meaning in literature, but there is a Minor difference between them. Scalability with automation is known as auto-scaling. Auto-scaling with optimization is known as elasticity.

Elasticity can be classified from various aspects. A detailed discussion of the classification of cloud elasticity is given as below.

3.0.1 Method

Elasticity can be classified as vertical, horizontal and hybrid from method point of view.

• Vertical: Vertical elasticity used to resize instance capacity in terms of CPU, Memory, and Storage or both at run time. It provides good flexibility to cloud provider to easily manage the resources. Some hypervisor that

supports vertical elasticity is Xen, VMware. Papers like [7],[8],[9],[10] ,[11], [12],[13], [14], [15],[16], [17],[18] discussed and applied vertical elasticity.

• Horizontal: Horizontal elasticity used to add/remove instances as per the change in workload. It is very simple approach and so used by many cloud providers. Some time it is inefficient to use as it fails to perfectly fit in requirement. In Papers [19], [20], [21], [22],[23], [24], [25], [26], [27], [28],[29], [30], [31],[32] ,[33],[34], [35],[36], [37],[38] discussed and used horizontal elasticity.

• Hydride: Hybrid elasticity allows both vertical and horizontal scaling together. Few research papers found that uses this approach like [5],[7],[23],[39]. It provides much flexibility in resource management compared to vertical only or horizontal only approach.

3.0.2 Policy

The decision for adding or removing cloud resources can be taken from the current utilization Of resources (like CPU, Memory utilization) or based on future resource demand. So reactive and proactive are the two possible policies for resource addition or removal.

• Reactive: In Reactive policy decision for triggering is done based on some threshold or rules and no prediction is used. Again, the threshold value can be changed during the operation so it can be further classified as a fixed threshold and dynamic threshold. In some papers this policy is used are [8],[10],[12],[18],[26],[31], [35], [36],[40],[41],[42],[43],[44]. it can be further detailed as Fixed Threshold and Dynamic Threshold.

• Proactive: In Proactive policy, forecasting technique is used for getting trends and future resource requirements. Scaling decisions are taken accordingly. In some papers used this policy are [27], [28],[29],[45]. some proactive techniques can be discussed as below.

– Time Series Analysis: In cloud computing for auto-scaling purpose time series, analysis-based methods can be used. time series analysis contains various prediction methods. It is used to predict the next value in the series. Accuracy of the prediction is depended on the kind of pattern for which it is used and also depends heavily on associated parameters [46]. It contains several techniques like moving average, autoregression, ARMA, ARIMA, Holt Winter etc. In some papers time series analysis used are [7], [19],[28],[45],[29],[47],[48],[49],[50].

– Control Theory: Sharing of resources among cloud applications is the aim of the control model. Control systems have mainly two groups 1) open-loop 2) closed-loop control systems. In the open-loop control system, the output depends on the input, and no feedback loop is used to improve the output. While in closed-loop control system feedback loop is used to compare the existing output with the desired output, it manages the set of resources to achieve the desired output. In some papers that used control theory are [11],[51],[18].

– Queuing Theory: Queueing theory is based on a mathematical model, and it is used to study the queue or waiting line. A queueing model is built in order to predict queue lengths and waiting time. It is best suited for the system with a stationary nature. In papers that used queuing theory are [20],[37],[52],[53].

– Reinforcement Learning: Reinforcement learning is a method of learning, and it develops optimal policies on a given state. The reinforcement learning model continuously submits the action in a given environment for better results. In this model, agents submit their actions based on the environment status, and agents get rewards from the environment. The objective considered for the agent is to maximize reward function value or to minimize the risk. In papers that used reinforcement learning are [34],[54].

• Reactive and Proactive Both [31],[33],[55].

3.0.3 Scope

Elasticity actions can be taken at the application level or infrastructure level if actions are taken at the application level is also known as embedded elasticity. Again, the application can be considered with a single-tier or multi-tier. The elasticity controller is used for monitoring the system. It takes a decision at the infrastructure level.

Infrastructure considered are Virtual Machine (VM), Containers. In a few papers, virtual machines and containers considered together.

• **Application**
– Single tier [18],[28],[35]
– Multi-tier [20],[52],[53]
• **Infrastructure**
– VM [18],[28],[45],[29],[53]
– Container [8],[36],[42],[44],[48]
– VM and Container Together [10],[38]
• **Both Application and Infrastructure** [18],[44],[48]

3.0.4 Architecture

from the architecture point of view elasticity solutions are 1) centralized and 2) decentralized. only one elasticity controller is used in centralized elasticity solutions. Single point of failure may occur in a centralized controller. In decentralized elasticity solutions, the architecture includes multiple elasticity controller.

• **Centralized** [18],[28],[35],[36],[29]
• **Decentralized** [56],[57],[58]

3.0.5 Provider

Elasticity mechanism can be provided for a single cloud or for multiple clouds. Single cloud means a single cloud service provider and may have multiple data centers distributed at multiple regions. Multi-cloud refers to multiple cloud providers. In literature, many papers found supporting a single cloud provider but found few that support and discusses multi-cloud.

• **Single** [18],[28],[35],[36],[45],[29]
• **Multiple** [37],[59],[60]

3.0.6 Scaling Action Indicators

Elasticity scaling action indicators can be memory utilization, response time or CPU utilization. At particular interval scaling algorithm may check this indicator's value for triggering scaling actions.

• **Memory Utilization** [12],[18]
• **Response Time** [35],[61]
• **CPU Utilization** [18],[30],[36],[40],[45]
• **Number Of Requests** [28],[29]

Here it is also important to discuss the MAPE loop proposed first by IBM for architecting a self-adaptive system. MAPE is widely used in the autonomic system. It contains a loop of four steps Monitor (M), Analyze (A), Plan (P), Execute (E). In the step of monitoring, monitoring information is collected. In analyzing step, the analysis of the present status of the system is performed. As per the analysis phase, the decision regarding auto-scaling is planned in the plan step. In executing step planned scaling operations are executed.

4. Realted Work

This section discusses research work on elasticity. It contains the basic terminology and classification of elasticity solutions.

M. Tighe and M. Bauer [35] proposed a new algorithm that combines both automatic scaling of application and dynamic allocation of a virtual machine that meets the goal of both user and provider. Used reactive policy for autoscaling evaluations shows that in the integrated algorithm (autoscaling and dynamic allocation) provides

better application performance and reduces VM live migrations. This algorithm satisfies QoS and minimizes the operational cost.

R.N.Calheiros, E.Masoumi, R.Ranjan, and R.Buyya [29] In this paper cloud workload prediction module is proposed, which uses ARIMA model. The author considered the SaaS service model. Real traces of requests to a web server is used to evaluate the accuracy of the ARIMA model. Results describe that this proposed model gains an average accuracy of 91 percent. It increases the efficiency in resource utilization with minimum effect on QoS.

Y. Hu, B. Deng, F. Peng, and D. Wang [28]. In this paper, the author proposed three prediction models for predicting workload from monitoring data. The time-series approach is used for analyzing the monitoring workload, and the Kalman filter model is used to forecast the cloud workload. It also introduced a novel model of pattern matching for predicting and analysis of workload. Results describe that the proposed approach improves the prediction accuracy and also reduce the automatic scaling delay.

P. D. Kaur and I. Chana [37] proposed a QoS-Aware Resource Elasticity (QRE) framework which contains components like Workload Analyzer, Application Centric Behavior Analyzer, QoS Mapper, Resource Centric Behavior analyzer, Performance Database.it allows the cloud provider to study the application behavior and allows to dynamically scale the cloud resources that host the application components. An experiment performed using amazon ec2 setup and result indicates the effectiveness of this approach, which complies with the agreed QoS attributes of the users.

P.Singh, P.Gupta, and K.Jyoti [47] proposed an adaptive prediction model. This model uses linear regression, ARIMA, and SVR. The considered scenario is a web application. The Model is selected as per the workload feature, and the workload classifier has been proposed for that. A heuristic approach is used to select model parameters and used real traces for evaluation. The proposed approach provides a significant improvement in RMSQE and MAPE metrics and improves the QoS for the web application.

Y. Ogawa, G. Hasegawa, and M. Murata [49] In this paper author proposed an approach in which private cloud rents resources from a public cloud provider. The bursting cloud approach is discussed here. It uses both long- and short-term predictions of requests. In a private cloud pool of VMS are as signed by considering the one-week predictions while one- hour prediction is taken for VM activation in private and public clouds. Experimentation results show that even if bursty requests are received by website and one-hours of predictions include a MAPE error of 0.2, in private data center alone the total cost of provisioning get minimizes to half of the existing and response time is kept below 0.15 seconds for 95 percent of the response time.

Q. Zhang, L. T. Yang, Z. Yan, Z. Chen, and P. Li [62] in this paper author proposed deep learning model. It is based on canonical polyadic decomposition. This model is used for predicting the cloud workload for industry informatics. Weight metrics are compressed significantly to canonical polyadic format. In this paper, an efficient algorithm is developed to train parameters. This learning model is used for the VM workload prediction in the cloud. The PlanetLab dataset is used. The proposed model gains a higher training efficiency, workload prediction accuracy than another machine learning-based approach.

P. Tang, F. Li, W. Zhou, TW. Hu, and L. Yang [63] in this paper author presented SRSA - SLA aware resource-efficient self-learning approach. This approach is for making an auto-scaling policy decision. The scenario of the service volatility is classified into the daily busy and ideal scenario, and burst traffic scenario, Formulated workload as a discrete-time series. It treats the policy-making procedure as MDP. Parameters in the reinforcement learning process are also tuned. The results show that the proposed solution outperforms the threshold-based policy and voting policy adopted by the right scale in oscillation suppression, QoS guarantee, and energy saving.

T. C. Chieu, A. Mohindra, A. A. Karve, and A. Segal [64] in this paper author proposed a dynamic scaling algorithm. This algorithm manages the automatic VM resource provisioning based on the threshold. Front end load balancer is used to route and balances the request to deployed web applications on VM. an active session is considered for making scaling decisions.

J. Huang, C. Li, and J. Yu [65] proposed a resource prediction model. This model used double exponential smoothing, and it considers the present state of resource and history record. A cloudsim simulator is used for the experiment. Results indicate better prediction accuracy and performance.

N. Roy, A. Dubey, and A. Gokhale [19] in this paper, authors contributed to workload forecasting and optimal resource allocation. The author discussed the challenges involved in autoscaling. The next model prediction algorithm for workload forecasting is developed, which is used for resource autoscaling. Results show that the proposed algorithm satisfies both application QoS while minimizing operational cost. For workload prediction, the ARMA method has been used.

M. Wajahat, A. Gandhi, A. Karve, and A. Kochut [66] proposed ML Scale that is a machine learning-based auto-scaler and also it is an application-agnostic. This auto-scaler contains a neural network-based online performance modeler, regression-based metric predictor to estimate post scaling application, and system metric. The result shows that it reduces the resource cost of about 50% compared to the optimal static policy. No carefully tuned threshold policy is required.

A. Khan, X. Yan, S. Tao, and N. Anerousis [45] proposed a multiple time series based approach. This approach first search for repeatable workload patterns of applications running on different virtual machines. Emphasis is given on identifying a group of VMS which frequently exhibits similar workload patterns during a time period for which VM groups are active by developing a co-clustering technique using workload data samples as time series. An introduced method based on the hidden Markov model to discover the VM cluster and to predict variation in workload patterns. Results show that this approach can help in understanding the workload of group-level features and just make an accurate prediction for the workload that changes in a cloud.

H.Arabnejad, C.Pahl,P. Jamshidi, and G.Estrada [54] this paper compares two dynamic learning strategies that are based on a fuzzy logic system, and that learns as well as modifies fuzzy scaling rules during runtime. The fuzzy logic controller is used with two reinforcement learning (RL) approaches that is Fuzzy SARSA Learning and Fuzzy Q-Learning. The result shows that the proposed approaches can handle different load traffic situations while reducing operating costs and prevents SLA violations.

B. B. Bibal and D. Dharma [31] proposed HAS - Hybrid auto-scaler for adjusting the required resource automatically. HAS forecast the future behavior of the system by using the time series analysis method. The queuing model has been used to compute the required capacity. A reactive approach is used to scale out resources. Used CTMM - continuous-time Markov model to balance the load efficiently.

L. Lu et al. [7] proposed AppRM tool. for SLOs, this tool automatically sets resource control for VM and resource pool. This tool contains the multilevel of virtual application managers and also resource pool managers. AppRM scales VM vertically, It manages vertical scaling by resource adjustment at the Virtual Machine level or at the resource pool level.

J. F. Naomi and S. Roobini [55] proposed Independent Recurrent Neural Network (IndRNN) method for workload prediction. In this paper, both reactive and proactive approaches are used. Here the reactive approach is used to minimize the negative return from the proactive policy.

E. G. Radhika, G. Sudha Sadasivam, and J. Fenila Naomi [70] in this paper used a deep learning technique known as Recurrent Neural Network with Long Short Term Memory (RNN-LSTM) for predicting the future demand using historical data. Here the reactive approach is used to minimize the negative return from the proactive policy. The results show that, when the massive workload arrives, the proposed system predicts future demand and spins the instance accordingly.

W. Dawoud, I. Takouna, and C. Meinel [18] proposed an elastic system architecture in which three controllers implemented for CPU, Memory, and Application. All three controllers work in a parallel manner to better manage the resource and application. The proposed architecture reduces SLO violations.

G. MoltÃ³, M. Caballer, and C. De Alfonso [12] proposed a CloudVamp framework cloudVamp means cloud virtual machine automatic memory procurement framework. It considered memory oversubscriptions. It can be

integrated with an on-premise cloud management platform (CMP) for automatically monitoring the VMs, to allocate memory dynamically in order to get requirement regarding present memory for the applications that are in execution. This framework enhances VM consolidation per Physical machine node.

S. Sotiriadis, N. Bessis, C. Amza, and R. Buyya [41] proposed an inter-cloud elasticity framework. In this work, the author introduced an inter-cloud load balancer (ICLB). It allows scaling operations by neglecting down times and communication failures. It distributes incoming user HTTP traffic to multiple instances. This instance belongs to inter-cloud applications, services. It performs dynamic reconfiguration of resources as per the real-time requirements.

X. Tang and F. Zhang [36] proposed a framework that monitors containers resource utilization and performs container scaling operations as per requirements. The auto-scaler has four parts that are monitoring mechanism, decision mechanism, history recorder, and execution mechanism. For the same configuration, in repeat mode auto-scaler perform a good job, here workload contains the recurring pattern, but for shuffle, repeat shuffle modes elasticity mechanism result is noticeably worse compared to the other two modes. It is found that to balance system stability and good elasticity, the length of the cool-down period should be proper. The auto scaler is tested by taking Stress workload on DC/OS cloud infrastructure.

S. Taherizadeh and V. Stankovski [44] proposed a dynamic multilevel auto-scaling method. This method can work with a dynamically changing threshold that uses monitoring data from both infrastructure and application. DM is compared to the existing seven auto-scaling methods with different workload scenarios. The experimental results show that the DM method provides better performance under the different amounts of the workload, so this method is implemented in the SWITCH (software engineering system for time-critical cloud applications).

B. Xie, [42] in this paper, an intelligent scheduling system is proposed. The aim of this system is to improve resource utilization. An auto-scaling algorithm mechanism is integrated into the intelligent kernel for analyzing and predicting the application behavior. The minimum amount of instances is calculated for pre-allocation and placement To reduce the response latency and lower deployment cost. Only one group of periodic requests are considered. The performance parameter used is the CPU utilization.

Y. Meng, R. Rao, X. Zhang, and P. Hong [50] proposed the CRUPA algorithm. It is a response utilization prediction algorithm that used the ARIMA model from time series analysis. Experimental results show that this algorithm has high prediction accuracy, and it can scale resources well.

O. A. B. U. Oun [43] proposed JQueuer and CAutoScaler that offer job-queueing and automated scaling for the level of containers. It takes jobs list and auto-scaling policy, starts the containers in the cloud and dispatches the jobs to containers and perform scaling operations in order to complete the jobs according to the applied policy.

Y. Al-Dhuraibi, F. Paraiso, N. Djarallah, and P. Merle [8] in this paper author proposed ElasticDocker. That is a system powering vertical elasticity.it uses the principle of IBM MAPE-K. It controls the scaling of both CPU and Memory that is assigned to each container as per the application workload. Container migration is performed by CRIU when host capacity reaches to its maximum limit. The author used Scalair's private cloud infrastructure for experiment results shows that the proposed approach outperforms the Kubernetes elasticity by 37.63 percentage.

T. Ye, X. Guangtao, Q. Shiyou, and L. Minglu [48] in this paper author proposed an auto-scaler for containerized elastic applications. The author prepared a hybrid scaling strategy. This strategy is based on the resource demand prediction model to achieve SLA for quickly varying workloads. This work only considered CPU resources. The proposed approach ensures a low ratio of a violation.

A. Barros, D. Grigori, N. C. Narendra, and H. K. Dam [38] considered four dimensions of scaling. Containers and virtual machines can adjust horizontally and vertically. This paper addresses the four-fold auto-scaling. This paper formulated a scaling decision problem in the form of a multi-objective optimization problem. The result shows that the proposed approach chooses and executes scaling actions that achieve twenty to twenty-eight percent of cost reduction over the baselines.

Y. Al-dhuraibi et al [10] In this paper author proposed an approach to consider both virtual machines and containers for vertical elasticity. They proposed a technique for applications based on the container to adjust resources at both container and VM level. Major components of the system are monitoring components and docker controller. The approach is evaluated using the RUBiS benchmark application. This approach reacts quickly and improves the performance of the application. This approach outperforms the vertical elasticity controller of the container by 18.34 % , horizontal elasticity by 39.6%, and vertical elasticity controller of VM by 70%.

Table 1 Review of Cloud Elasticity

Paper Reference No.	Scope (VM/ Container / Both)	Metric	Mode (Reactive/ Proactive/ Both)	Policy (Horizontal/ Vertical/ Both)
[35]	VM	Active Hosts, Number of Migrations, Response Time	Reactive	Horizontal
[29]	VM	Average Service Time, Minimum number of VM , Max number of VM, VM Hours, response time	Proactive (Arima – time series analysis)	Horizontal
[28]	VM	CPU Utilization	Proactive (time series analysis, kalman filter, , pattern matching approach)	Horizontal
[37]	VM	No. of VMS, Response Time, Resource Utilization	Proactive	Horizontal
[47]	VM	Number of Requests	Proactive (linear regression,Arima , SVM)	Horizontal
[49]	VM	Response Time	Proactive (ARIMA)	Horizontal
[62]	VM	CPU Utilization	Proactive (deep learning model that is based on the canonical polyadic decomposition)	Horizontal
[63]	VM	Workload	Proactive (reinforcement learning)	Horizontal
[64]	VM	Number of Active Sessions	Proactive (moving average)	Horizontal
[65]	VM	CPU Utilization	Proactive (time series analysis-double exponential smooting)	Horizontal
[19]	VM	Configuration Cost	Proactive (ARMA time series analysis)	Horizontal
[66]	VM	Response Time	Proactive (machine learning)	Horizontal
[45]	VM	CPU Utilization	Proactive (multiple time series apporach)	Horizontal
[54]	VM	Workload, Response Time , Number of Virtual Machines.	Proactive (Reinforcement Learning)	Horizontal
[31]	VM	Average. Response Time, Future Arrival Rate, System Utilization Rate, Capacity Available and Used	Reactive and Proactive (autoregression) both	Horizontal

Paper Reference No.	Scope (VM/ Container / Both)	Metric	Mode (Reactive/ Proactive/ Both)	Policy (Horizontal/ Vertical/ Both)
[7]	VM	Response Time, Application Throghput	Proactive (time series ARMA model)	Horizontal
[55]	VM	Workload	Proactive Independent Recurrent Neural Network (IndRNN)	Horizontal
[70]	VM	CPU Utilization	Proactive (RNN-LSTM)	Horizontal
[18]	VM	CPU,Memory Utilization , Response Time	Reactive (threshold based)	Vertical
[12]	VM	Memory Utilization	Reactive	Vertical
[41]	VM	CPU, Memory, Disk Usage	Reactive (CPU utilization)	Horizontal and vertical
[36]	Container	CPU Utilization Per Container and PM,Response Time	Reactive(\% of cpu utilized is a threshold)	Horizontal
[44]	Container	Uses both Application (Response time, Application Throughput) and infrastructure (CPU, Memory) level metrics	Reactive (multi-level monitoring system)	Horizontal
[42]	Container	CPU Utilization	Reactive (CPU utilization percentage as threshold)	Horizontal
[50]	Container	Resource Demand	Proactive (ARIMA)	Horizontal
[43]	Container	Number of Jobs in the Queue	Reactive	Horizontal
[8]	Container	CPU Utilization , Memory Utilization	Reactive (\% of cpu utilization as threshold)	Vertical
[48]	Container	CPU Utilization , Response Time	Proactive (ARIMA)	Horizontal and Vertical
[38]	VM and Container	Response Time	Using optimizaiton model	Horizontal
[10]	VM and Container	CPU and Memory Utilization	Reactive (\% of CPU utilization is threshold)	Vertical

5. Discussion And Future Directions

Difficulties in implementing general-purpose auto-scaler mechanism are related to

• Scaling Timing: when to take scaling decision is a very important question. The auto-scaler should answer that. It is because different applications running inside a cloud computing environment have different workload patterns and preferences for QoS.

• Resource Estimation: to estimate resources (how many resources to provision or de-provision?) during the next scaling action is a critical question. Over-provisioning can result in higher costs and improper resource utilization. Under-provisioning can result in SLA Violations.

• Scaling Method: types of scaling methods are horizontal scaling and vertical scaling.it is also possible to use both in a combination called diagonal scaling. selecting a particular method for managing resources is also an important factor that must be considered during the scaling plan. According to the study of Elasticity as above. we can list out possible future directions as below.

• For cloud elasticity prediction method/technique plays a vital role, so it is required to select the prediction method/technique that provides the best prediction accuracy. To choose the best prediction method for a particular application can be a potential future work.

• VM may fail at runtime. A cloud environment is subjected to such uncertainties. So, consideration of these aspects can be a potential future work.

• Horizontal elasticity is discussed much, but Vertical elasticity is still not much explored. In vertical elasticity also either CPU or Memory is considered as performance parameter but not both together. So, consideration of CPU and Memory together can be a potential future work.

• Only considered homogeneous style of resources. Work can be extended by considering the mixture of on-demand, reserved, and related resources.

• Major autoscaling work considered only SLA achievement, but the energy and carbon aware auto-scaling is still needed to explore.

• Cloud container gained much popularity recently. A container can be placed on a physical machine or virtual machine. No work is found that addresses the management of physical machines or virtual machines for a container. Only the auto-scaling of a container or virtual machine is addressed.

• Hybrid elasticity (horizontal + vertical elasticity) is still not much explored.

• Major cloud service provider rent resources and charge on per hour basis. Work can be extended by introducing the concept of smart kill (turn down the VM just before it completes the hour to lower the cost and use this extra resource to address a sudden increase in workload).

6. Conclusion

Elasticity is the dynamic property of the cloud and is used for provisioning and de-provisioning cloud resources to map between workload and cloud resources. In this paper, we classified elasticity with various aspects. elasticity classification with respect to scope, mode, and policy is our major interest. From this review we also proposed promising future directions that can be extended by the research community in the future.

References

[1] S. Chowhan, A. Kumar, and S. Shirwaikar, “Data Driven Resource Provisioning for Efficient Utilization of Cloud Resources,” *Int. J. Inf. Technol. Electr. Eng.*, vol. 8, no. 6, pp. 45–49, 2019.

[2] M. M. Khan, H. Ahmed, S. M. Pirzada, and A. Sajid, “Survey on Cloud Computing Services, Platforms and Architecture,” *Int. J. Inf. Technol. Electr. Eng.*, vol. 8, no. 2, pp. 17–21, 2019.

[3] S. F. Piraghaj, A. V. Dastjerdi, R. N. Calheiros, and R. Buyya, “Efficient Virtual Machine Sizing for Hosting Containers as a Service (SERVICES 2015),” in *2015 IEEE World Congress on Services*, IEEE, Jun. 2015, pp. 31–38. doi: 10.1109/SERVICES.2015.14.

[4] A. Mosallanejad, “A HIERARCHICAL SELF-HEALING SLA FOR CLOUD COMPUTING,” *Int. J. Digit. Inf. Wirel. Commun.*, vol. 4, no. 1, pp. 43–52, 2014, doi: 10.17781/P001082.

[5] N. Ma, A. Maghari, N. Sarkissian, and W. Clark Lambert, “Elasticity in Cloud Computing: What It Is, and What It Is Not,” *Skinmed*, vol. 8, no. 6, pp. 361–362, 2010, [Online]. Available: http://sdqweb.ipd.kit.edu/publications/pdfs/HeKoRe2013-ICAC-Elasticity.pdf

[6] C. Vazquez, R. Krishnan, and E. John, "Time series forecasting of cloud data center workloads for dynamic resource provisioning," *J. Wirel. Mob. Networks, Ubiquitous Comput. Dependable Appl.*, vol. 6, no. 3, pp. 87–110, 2015, doi: 10.22667/JOWUA.2015.09.31.087.

[7] L. Lu *et al.*, "Application-driven dynamic vertical scaling of virtual machines in resource pools," in *2014 IEEE Network Operations and Management Symposium (NOMS)*, IEEE, May 2014, pp. 1–9. doi: 10.1109/NOMS.2014.6838238.

[8] Y. Al-dhuraibi, F. Paraiso, N. Djarallah, and P. Merle, "Autonomic Vertical Elasticity of Docker Containers with ElasticDocker To cite this version : Autonomic Vertical Elasticity of Docker Containers with ELASTICDOCKER," *Proc. 10th IEEE Int. Conf. Cloud Comput. IEEE CLOUD 2017*, 2017.

[9] R. Bruno *et al.*, "Dynamic vertical memory scalability for OpenJDK cloud applications," in *Proceedings of the 2018 ACM SIGPLAN International Symposium on Memory Management - ISMM 2018*, New York, New York, USA: ACM Press, 2018, pp. 59–70. doi: 10.1145/3210563.3210567.

[10] Y. Al-Dhuraibi, F. Zalila, N. Djarallah, and P. Merle, "Coordinating vertical elasticity of both containers and virtual machines," in *CLOSER 2018 - Proceedings of the 8th International Conference on Cloud Computing and Services Science*, 2018, pp. 322–329. doi: 10.5220/0006652403220329.

[11] S. Farokhi, P. Jamshidi, E. Bayuh Lakew, I. Brandic, and E. Elmroth, "A hybrid cloud controller for vertical memory elasticity: A control-theoretic approach," *Futur. Gener. Comput. Syst.*, vol. 65, pp. 57–72, Dec. 2016, doi: 10.1016/j.future.2016.05.028.

[12] G. Moltó, M. Caballer, and C. De Alfonso, "Automatic memory-based vertical elasticity and oversubscription on cloud platforms," *Futur. Gener. Comput. Syst.*, vol. 56, no. October, pp. 1–10, Mar. 2016, doi: 10.1016/j.future.2015.10.002.

[13] S. Shekhar, H. Abdel-Aziz, A. Bhattacharjee, A. Gokhale, and X. Koutsoukos, "Performance Interference-Aware Vertical Elasticity for Cloud-Hosted Latency-Sensitive Applications," in *2018 IEEE 11th International Conference on Cloud Computing (CLOUD)*, IEEE, Jul. 2018, pp. 82–89. doi: 10.1109/CLOUD.2018.00018.

[14] E. B. Lakew, A. V. Papadopoulos, M. Maggio, C. Klein, and E. Elmroth, "KPI-Agnostic Control for Fine-Grained Vertical Elasticity," in *Proceedings - 2017 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, CCGRID 2017*, IEEE, May 2017, pp. 589–598. doi: 10.1109/CCGRID.2017.71.

[15] A. P. Barzu, M. Carabas, and N. Tapus, "Scalability of a Web Server: How Does Vertical Scalability Improve the Performance of a Server?," in *Proceedings - 2017 21st International Conference on Control Systems and Computer, CSCS 2017*, IEEE, May 2017, pp. 115–122. doi: 10.1109/CSCS.2017.22.

[16] C. Ruiz, H. A. Duran-Limon, and N. Parlavantzas, "An RLS Memory-based Mechanism for the Automatic Adaptation of VMs on Cloud Environments," in *Proceedings of the 2017 Workshop on Adaptive Resource Management and Scheduling for Cloud Computing - ARMS-CC '17*, New York, New York, USA: ACM Press, 2017, pp. 17–23. doi: 10.1145/3110355.3110358.

[17] S. Farokhi, E. B. Lakew, C. Klein, I. Brandic, and E. Elmroth, "Coordinating CPU and Memory Elasticity Controllers to Meet Service Response Time Constraints," in *2015 International Conference on Cloud and Autonomic Computing*, IEEE, Sep. 2015, pp. 69–80. doi: 10.1109/ICCAC.2015.20.

[18] W. Dawoud, I. Takouna, and C. Meinel, "Elastic VM for Cloud Resources Provisioning Optimization," in *Communications in Computer and Information Science*, 2011, pp. 431–445. doi: 10.1007/978-3-642-22709-7_43.

[19] N. Roy, A. Dubey, and A. Gokhale, "Efficient autoscaling in the cloud using predictive models for workload forecasting," in *Proceedings - 2011 IEEE 4th International Conference on Cloud Computing, CLOUD 2011*, 2011, pp. 500–507. doi: 10.1109/CLOUD.2011.42.

[20] D. Perez-Palacin, R. Mirandola, and R. Calinescu, "Synthesis of Adaptation Plans for Cloud Infrastructure with Hybrid Cost Models," in *2014 40th EUROMICRO Conference on Software Engineering and Advanced Applications*, IEEE, Aug. 2014, pp. 443–450. doi: 10.1109/SEAA.2014.57.

[21] S. K. Moghaddam, R. Buyya, and K. Ramamohanarao, "ACAS: An anomaly-based cause aware auto-scaling framework for clouds," *J. Parallel Distrib. Comput.*, vol. 126, pp. 107–120, Apr. 2019, doi:

10.1016/j.jpdc.2018.12.002.

[22] A. Naskos *et al.*, "Dependable Horizontal Scaling Based on Probabilistic Model Checking," in *2015 15th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, IEEE, May 2015, pp. 31–40. doi: 10.1109/CCGrid.2015.91.

[23] M. R. López and J. Spillner, "Towards Quantifiable Boundaries for Elastic Horizontal Scaling of Microservices," in *Companion Proceedings of the10th International Conference on Utility and Cloud Computing - UCC '17 Companion*, New York, New York, USA: ACM Press, 2017, pp. 35–40. doi: 10.1145/3147234.3148111.

[24] A.-F. Antonescu and T. Braun, "Simulation of SLA-based VM-scaling algorithms for cloud-distributed applications," *Futur. Gener. Comput. Syst.*, vol. 54, pp. 260–273, Jan. 2016, doi: 10.1016/j.future.2015.01.015.

[25] A. Naskos, A. Gounaris, and P. Katsaros, "Cost-aware horizontal scaling of NoSQL databases using probabilistic model checking," *Cluster Comput.*, vol. 20, no. 3, pp. 2687–2701, Sep. 2017, doi: 10.1007/s10586-017-0816-5.

[26] V. Podolskiy, A. Jindal, and M. Gerndt, "IaaS Reactive Autoscaling Performance Challenges," in *IEEE International Conference on Cloud Computing, CLOUD*, 2018, pp. 954–957. doi: 10.1109/CLOUD.2018.00144.

[27] S. Dipietro, R. Buyya, and G. Casale, "PAX: Partition-aware autoscaling for the Cassandra NoSQL database," in *NOMS 2018 - 2018 IEEE/IFIP Network Operations and Management Symposium*, IEEE, Apr. 2018, pp. 1–9. doi: 10.1109/NOMS.2018.8406271.

[28] Y. Hu, B. Deng, F. Peng, and D. Wang, "Workload prediction for cloud computing elasticity mechanism," in *Proceedings of 2016 IEEE International Conference on Cloud Computing and Big Data Analysis, ICCCBDA 2016*, IEEE, Jul. 2016, pp. 244–249. doi: 10.1109/ICCCBDA.2016.7529565.

[29] R. N. Calheiros, E. Masoumi, R. Ranjan, and R. Buyya, "Workload Prediction Using ARIMA Model and Its Impact on Cloud Applications' QoS," *IEEE Trans. Cloud Comput.*, vol. 3, no. 4, pp. 449–458, Oct. 2015, doi: 10.1109/TCC.2014.2350475.

[30] A. Bauer, J. Grohmann, N. Herbst, and S. Kounev, "On the Value of Service Demand Estimation for Auto-scaling," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, IEEE, 2018, pp. 142–156. doi: 10.1007/978-3-319-74947-1_10.

[31] B. B. Bibal and D. Dharma, "HAS: Hybrid auto-scaler for resource scaling in cloud environment," *J. Parallel Distrib. Comput.*, vol. 120, pp. 1–15, Oct. 2018, doi: 10.1016/j.jpdc.2018.04.016.

[32] M. S. Aslanpour, M. Ghobaei-Arani, and A. Nadjaran Toosi, "Auto-scaling web applications in clouds: A cost-aware approach," *J. Netw. Comput. Appl.*, vol. 95, pp. 26–41, 2017, doi: 10.1016/j.jnca.2017.07.012.

[33] Y. Li and Y. Xia, "Auto-scaling web applications in hybrid cloud based on docker," in *Proceedings of 2016 5th International Conference on Computer Science and Network Technology, ICCSNT 2016*, IEEE, Dec. 2017, pp. 75–79. doi: 10.1109/ICCSNT.2016.8070122.

[34] B. Asgari, M. Ghobaei Arani, and S. Jabbehdari, "An Effiecient Approach for Resource Auto-Scaling in Cloud Environments," *Int. J. Electr. Comput. Eng.*, vol. 6, no. 5, p. 2415, Oct. 2016, doi: 10.11591/ijece.v6i5.10639.

[35] M. Tighe and M. Bauer, "Integrating cloud application autoscaling with dynamic VM allocation," in *IEEE/IFIP NOMS 2014 - IEEE/IFIP Network Operations and Management Symposium: Management in a Software Defined World*, 2014. doi: 10.1109/NOMS.2014.6838239.

[36] X. Tang and F. Zhang, "Quantifying Cloud Elasticity on Smart Devices with Container-based Autoscaling," *Casalicchio, E. Clust. Comput*, 2017.

[37] P. D. Kaur and I. Chana, "A resource elasticity framework for QoS-aware execution of cloud applications," *Futur. Gener. Comput. Syst.*, vol. 37, pp. 14–25, Jul. 2014, doi: 10.1016/j.future.2014.02.018.

[38] A. Barros, D. Grigori, N. C. Narendra, and H. K. Dam, *Service-oriented computing: 13th international*

conference, ICSOC 2015 Goa, India, November 16-19, 2015 proceedings, vol. 9435, no. July 2017. in Lecture Notes in Computer Science, vol. 9435. Berlin, Heidelberg: Springer Berlin Heidelberg, 2015. doi: 10.1007/978-3-662-48616-0.

[39] H. C. Lim, S. Babu, and J. S. Chase, "Automated control for elastic storage," in *Proceeding of the 7th International Conference on Autonomic Computing, ICAC '10 and Co-located Workshops*, New York, New York, USA: ACM Press, 2010, pp. 1–10. doi: 10.1145/1809049.1809051.

[40] A. A. D. P. Souza and M. A. S. Netto, "Using Application Data for SLA-Aware Auto-scaling in Cloud Environments," in *2015 IEEE 23rd International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems*, IEEE, Oct. 2015, pp. 252–255. doi: 10.1109/MASCOTS.2015.15.

[41] S. Sotiriadis, N. Bessis, C. Amza, and R. Buyya, "Elastic load balancing for dynamic virtual machine reconfiguration based on vertical and horizontal scaling," *IEEE Trans. Serv. Comput.*, vol. 12, no. 2, pp. 319–334, Mar. 2019, doi: 10.1109/TSC.2016.2634024.

[42] B. Xie, G. Sun, and G. Ma, "Prediction-based autoscaling for container-based PaaS system," in *Proceedings - 2017 IEEE 2nd International Conference on Automatic Control and Intelligent Systems, I2CACIS 2017*, IEEE, Oct. 2017, pp. 19–24. doi: 10.1109/I2CACIS.2017.8239026.

[43] O. Abu Oun and T. Kiss, "Job-queuing and auto-scaling in container-based cloud environments," in *CEUR Workshop Proceedings*, 2019.

[44] S. Taherizadeh and V. Stankovski, "Dynamic multi-level auto-scaling rules for containerized applications," *Comput. J.*, vol. 62, no. 2, pp. 174–197, 2019, doi: 10.1093/comjnl/bxy043.

[45] A. Khan, Xifeng Yan, Shu Tao, and N. Anerousis, "Workload characterization and prediction in the cloud: A multiple time series approach," in *2012 IEEE Network Operations and Management Symposium*, IEEE, Apr. 2012, pp. 1287–1294. doi: 10.1109/NOMS.2012.6212065.

[46] A. Bauer, N. Herbst, S. Spinner, A. Ali-Eldin, and S. Kounev, "Chameleon: A Hybrid, Proactive Auto-Scaling Mechanism on a Level-Playing Field," *IEEE Trans. Parallel Distrib. Syst.*, vol. 30, no. 4, pp. 800–813, Apr. 2019, doi: 10.1109/TPDS.2018.2870389.

[47] P. Singh, P. Gupta, and K. Jyoti, "TASM: technocrat ARIMA and SVR model for workload prediction of web applications in cloud," *Cluster Comput.*, vol. 22, no. 2, pp. 619–633, Jun. 2019, doi: 10.1007/s10586-018-2868-6.

[48] T. Ye, X. Guangtao, Q. Shiyou, and L. Minglu, "An Auto-Scaling Framework for Containerized Elastic Applications," in *Proceedings - 2017 3rd International Conference on Big Data Computing and Communications, BigCom 2017*, IEEE, Aug. 2017, pp. 422–430. doi: 10.1109/BIGCOM.2017.40.

[49] Y. OGAWA, G. HASEGAWA, and M. MURATA, "Prediction-Based Cloud Bursting Approach and Its Impact on Total Cost for Business-Critical Web Systems," *IEICE Trans. Commun.*, vol. E100.B, no. 11, pp. 2007–2016, 2017, doi: 10.1587/transcom.2016NNP0006.

[50] Y. Meng, R. Rao, X. Zhang, and P. Hong, "CRUPA: A container resource utilization prediction algorithm for auto-scaling based on time series analysis," in *2016 International Conference on Progress in Informatics and Computing (PIC)*, IEEE, Dec. 2016, pp. 468–472. doi: 10.1109/PIC.2016.7949546.

[51] D. Serrano *et al.*, "Towards QoS-Oriented SLA Guarantees for Online Cloud Services," in *2013 13th IEEE/ACM International Symposium on Cluster, Cloud, and Grid Computing*, IEEE, May 2013, pp. 50–57. doi: 10.1109/CCGrid.2013.66.

[52] R. Han, M. M. Ghanem, L. Guo, Y. Guo, and M. Osmond, "Enabling cost-aware and adaptive elasticity of multi-tier cloud applications," *Futur. Gener. Comput. Syst.*, vol. 32, no. 1, pp. 82–98, Mar. 2014, doi: 10.1016/j.future.2012.05.018.

[53] B. Urgaonkar, P. Shenoy, A. Chandra, and P. Goyal, "Dynamic Provisioning of Multi-tier Internet Applications The Case for A New Provisioning Technique," *Second Int. Conf. Auton. Comput.*, pp. 217–228, 2005.

[54] H. Arabnejad, C. Pahl, P. Jamshidi, and G. Estrada, "A Comparison of Reinforcement Learning Techniques for Fuzzy Cloud Auto-Scaling," in *2017 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID)*, IEEE, May 2017, pp. 64–73. doi:

10.1109/CCGRID.2017.15.

[55] J. F. Naomi and S. Roobini, "A Hybrid Auto scaling for Cloud applications using an Efficient Predictive technique in Private Cloud," *Indian J. Sci. Technol.*, vol. 12, no. 8, pp. 1–7, Feb. 2019, doi: 10.17485/ijst/2019/v12i8/141807.

[56] A. Najjar, X. Serpaggi, C. Gravier, and O. Boissier, "Multi-agent Negotiation for User-centric Elasticity Management in the Cloud," in *2013 IEEE/ACM 6th International Conference on Utility and Cloud Computing*, IEEE, Dec. 2013, pp. 357–362. doi: 10.1109/UCC.2013.104.

[57] B. An, V. Lesser, D. Irwin, and M. Zink, "Automated negotiation with decommitment for dynamic resource allocation in cloud computing," *Proc. Int. Jt. Conf. Auton. Agents Multiagent Syst. AAMAS*, vol. 2, no. Aamas, pp. 981–988, 2010.

[58] M. Siebenhaar, T. A. B. Nguyen, U. Lampe, D. Schuller, and R. Steinmetz, "Concurrent Negotiations in Cloud-Based Systems," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2012, pp. 17–31. doi: 10.1007/978-3-642-28675-9_2.

[59] R. Buyya, "Intercloud: Scaling of applications across multiple cloud computing environments," *10th Int. Conf. Algorithms Archit. Parallel Process.*, pp. 13–31, 2010, [Online]. Available: http://scholar.google.com.au/scholar?hl=en&q=intercloud+computing&as_sdt=0,5&as_ylo=2008&as_vis=0#4

[60] K. Keahey, P. Armstrong, J. Bresnahan, D. LaBissoniere, and P. Riteau, "Infrastructure outsourcing in multi-cloud environment," in *Proceedings of the 2012 workshop on Cloud services, federation, and the 8th open cirrus summit - FederatedClouds '12*, New York, New York, USA: ACM Press, 2012, p. 33. doi: 10.1145/2378975.2378984.

[61] M. C. de Abranches and P. Solis, "A mechanism of auto elasticity based on response times for cloud computer enviroments and autossimilar workload," in *2016 XLII Latin American Computing Conference (CLEI)*, IEEE, Oct. 2016, pp. 1–9. doi: 10.1109/CLEI.2016.7833403.

[62] Q. Zhang, L. T. Yang, Z. Yan, Z. Chen, and P. Li, "An Efficient Deep Learning Model to Predict Cloud Workload for Industry Informatics," *IEEE Trans. Ind. Informatics*, vol. 14, no. 7, pp. 3170–3178, Jul. 2018, doi: 10.1109/TII.2018.2808910.

[63] P. Tang, F. Li, W. Zhou, W. Hu, and L. Yang, "Efficient Auto-Scaling Approach in the Telco Cloud Using Self-Learning Algorithm," in *2015 IEEE Global Communications Conference (GLOBECOM)*, IEEE, Dec. 2014, pp. 1–6. doi: 10.1109/GLOCOM.2014.7417181.

[64] T. C. Chieu, A. Mohindra, A. A. Karve, and A. Segal, "Dynamic Scaling of Web Applications in a Virtualized Cloud Computing Environment," in *2009 IEEE International Conference on e-Business Engineering*, IEEE, 2009, pp. 281–286. doi: 10.1109/ICEBE.2009.45.

[65] J. Huang, C. Li, and J. Yu, "Resource prediction based on double exponential smoothing in cloud computing," in *2012 2nd International Conference on Consumer Electronics, Communications and Networks (CECNet)*, IEEE, Apr. 2012, pp. 2056–2060. doi: 10.1109/CECNet.2012.6201461.

[66] M. Wajahat, A. Gandhi, A. Karve, and A. Kochut, "Using machine learning for black-box autoscaling," in *2016 Seventh International Green and Sustainable Computing Conference (IGSC)*, IEEE, 2016, pp. 1–8. doi: 10.1109/IGCC.2016.7892598.