1. **Project title:** *An Event Driven and Light Weight Proactive Auto-Scaling Architecture for Cloud Application.*
2. **Type of research:** *Applied*
3. **Field of the research:** *ICT/CSE/Applied Science*
4. **Project background:**

The Internet business becomes more appealing with the introduction of cloud computing. The customers in a competitive market rent computing resources on a pay-per-use basis from cloud providers (CP). For example, Amazon provides virtual computers in their Elastic Compute Cloud (EC2) service. As of the fourth quarter of 2020, there are 4.66 billion active internet users (KEMP, 2020) in the world. It indicates a 321 million increased compared to 2019's fourth quarter (growth rate 7.4 percent). As of the vast user traffic, if we don't focus on the auto-scaling, it will cost not only huge power consumption but also poor user experience.

A lot of study has been done in the auto-scaling arena in recent years. The auto-scaling problem was addressed in a number of methods in these works. The majority of these researches focus on a single aspect of the subject. One field of study, for example, is devoted only to workload forecasting. These efforts are aimed at developing a method for accurately forecasting future workload for each service. Other studies aim to forecast future resource needs, while others focus on the decision making aspect of the auto-scaling problem in order to arrive at the best scaling solution. Such judgments take into account the service's present condition as well as its resource requirements.

Islam et al. conducted research in cloud and they proposed a proactive method. In their algorithm, they used an artificial neural network (Islam et al., 2012). The algorithm is capable of scaling based on resource utilization awareness. They measured the performance of neural networks. But they did not investigate their algorithm's efficiency in comparison with that of the reactive method.

A resource-aware solution was modeled in *CloudSim* by Huang *et al.* for a real-world application (Huang et al., 2012). However, they did not include pricing, which is regarded as one of the most significant application demands, in their study.

Mohamed et al. studied SLA-aware resource scaling using a reactive approach (Mohamed et al., 2014). However, in this study, the resource status was not taken into account. They were unable to study the criteria of Service Level Agreement (SLA) violations, costs, and resource use.

Espert and Garcia also did reactive technique research (Garc´ıa et al., 2014). In their research, they suggested a resource management architecture using a horizontal scaling algorithm. As a type of RT, their suggested technique benefited from the cloud's SLA awareness. They focused on cloud cost to assess their algorithm, and while calculating cost, they did not account for SLA violation penalty.

Using resource usage aware proactive methods, Qavami *et al.* presented an approach for predicting resources for application (Qavami et al., 2014). As part of their decision-making process, they use data from the state of resource usage. In this case, they calculated the cost without considering the penalty for SLA violations.

de Assun¸c˜ao *et al.* evaluated the SLA-focused work in form of user patience in his work on response time from applications (Assuncao et al., 2015). They employed a proactive strategy called weighted exponential smoothing in their approach. This strategy assisted in improving the research result by proactively keeping track of the response time users experience and the forecast of resource use.

Chana and Singh presented a proactive approach based on QoS constraints (Gill and Chana, 2015). The impact of the SLA breach penalty on the cost was not taken into account in this method. Their major research focus was heterogeneous resource supplies.

In a study, Caballer *et al.* examined the use of cloud computing. They provided a reactive resource management architecture in their research (Molt´o et al., 2016). This auto-scaling technique takes a resource conscious approach. In this design, the algorithm is given decision-making capabilities but is merely aware of resource (memory) usage. Furthermore, the recommended strategy was not contrasted with proactive alternatives. Again, they did not compare the approaches in terms of the criterion for SLA violation cost and SLA awareness.

Bankole and Ajila proposed a SLA-aware and resource-aware algorithm (Ajila and Bankole, 2016). However, they solely assessed the outcomes of time series-related criteria in their research and did not include cost or quality of service (QoS) factors.

Ghobaei-Arani *et al.* proposed a proactive method that uses resource utilization parameters for auto-scaling (Ghobaei-Arani et al., 2016). The use of linear regression to analyze the arrival rate of user requests is also useful in their study. Their method reduces costs and SLA violations, and they use Learning Automata to make scaling decisions.

Aslanpour and Dashti proposed a proactive auto-scaling algorithm for cloud application (Aslanpour and Dashti, 2017). In their research they tried to minimize the overall cloud cost while maintaining user satisfaction. But in their research they could have minimized the cost more if they had use a different approach.

In (Ye et al., 2017), T. Ye and X. Guangtao suggested a combined technique for anticipating the required resources in order to maintain the SLA even when the workload of the system varies substantially. In this study, the scaler unit determines required resources needed for the next time interval based on the projected value. In this scenario, the method used to forecast time series is ARMA. In addition, another quick provisioning algorithm is also used. The technique assumes that vertical scaling should be employed to enable rapid resource provisioning in the face of severe workload variances in order to fulfill SLA requirements. In this study, the seasonality of data is not taken into consideration. Another disadvantage of this method is that some of the threshold values must be established by a human operator. In other works, the ARMA model has been used to predict time series. Roy et al., for example, has used this method to forecast the workload of the service (Roy et al., 2011). The proposed method in that paper is less accurate than many existing predictive algorithms, and it performs poorly when it comes to remembering the workload from previous timesteps.

Martinez *et al.* provided a technique for learning the service's workload patterns, which are repeated on a regular or seasonal basis, and these patterns were utilized to estimate workload of the service (Mart´ınez et al., 2017). A datastore is used in this method to store all relevant workload patterns. In the datastore, some patterns are preloaded. Based on the manually acquired information, further patterns may be incorporated to the system (e.g., with the help of a human operator). Each pattern has a number of features that can be used to compare a real workload to previously identified patterns. During runtime, this function attempts to match the current workload pattern with a known pattern in the datastore. The authors utilize the pattern's form and related attributes to match the current data with one of the patterns stored in the datastore. The pattern is denormalized using the actual workload values seen in runtime to anticipate future workload. The actual benefit of this method is that it anticipates service demand based on seasonal and periodic trends, making it ideal for systems with predictable workloads. However, this method is inapplicable to many current systems with changing workloads, resulting in poor job performance.

In (Vel´asquez et al., 2013; Makridakis S, 2018), it is observed that the computational requirements of machine learning are far greater than statistical approaches and exponential smoothing has a better fit over seasonal data. But ML has a better accuracy in predicting data for unknown circumstances. One of our main focuses is to design a lightweight model. Hence, in our approach, the lightweight exponential smoothing will be used with improved adaptability and accuracy.

5. Rationale:

i) Relationship of the objectives to existing scientific knowledge on the subject:

The present cloud technologies enable application vendors to rent virtual computers. These virtual machines are hosted by cloud providers. This lowers the expense of managing computing resources for application providers. Because of the difference between the frequency with which end user requests arrive and the lack of awareness of the resources available, the expectation of the application providers from the cloud providers to do more than just host applications. As a result, the cloud provider wants to minimize under and over provisioning of resources.

Over-provisioning raises costs for the application providers while under-provisioning creates consumer frustration. One practical answer to this problem is to lower the cost of renting virtual machines as long as the machine can follow the quality of service (QoS) restrictions. This quality of service (QoS) constraint is covered by a service level agreement (SLA) contract for a cloud environment. After the QoS has been finalized, the service provider and client sign this SLA contract. A major challenge in the internet business is finding a balance between cost and performance (Jiang et al., 2017). An algorithm based on the flexibility of cloud resources can be developed to automatically scale up or down resources to meet this issue (Veni and Saira Bhanu, 2016).

Based on how they handle the decision-making parameter, scaling algorithms are divided into two categories. They are: i) Reactive ii) Proactive.

Reactive scaling is a rule-based (threshold-based) technique that calculates the penalty for violating a service level agreement and then scales resources in response to the violation. However, there are several drawbacks to this approach. For example, in Amazon’s EC2 service, an on-demand virtual machine takes 7 to 15 minutes to boot up (Islam et al., 2012a). As a consequence, when the threshold is exceeded, this scaling suspends application vendors while the virtual machine is still operational. Decision-making following a rule violation is linked to an SLA violation penalty for application providers (Qu et al., 2018). Some threshold violations may be caused by transient fluctuations, in which case scaling choices may not be required (Xiao et al., 2013).

The proactive method appears to be capable of covering up the flaws of the reactive approach by looking back in time and predicting future resource usages. This approach can forecast future resource requirements before crisis situations and SLA breaches arise. Furthermore, it can address the problem of delayed VM starting by anticipating the future.

Researchers, on the other hand, are generally attempting to demonstrate which scaling factors are more useful in scaling decision-making. In 2016, Aslanpour and Dashti addressed how to choose resource usage factors and reaction time (RT) (Aslanpour and Dashti, 2016). Nonetheless, it indicates that knowledge discovery and examination of each parameter’s history can help achieve effective auto-scaling. As a result, the aim of this research is to present an event based auto-scaling approach for discovering and analyzing information from the background of the decision-making parameter.

In order to reduce procurement and management costs of resources, the cloud environment is used by the service providers (APs) to host their applications. However, variation in the traffic load of the client applications, as well as the appealing auto-scaling capability of the cloud resource, has prompted application providers to seek ways to reduce the cost of renting services. The aim of this project is to develop a constructive auto-scaling mechanism powered by events in cloud systems fitted with heuristic predictors.

ii) Relevance of the proposed study to national priorities or regional priorities:

Because of the flexibility of cloud computing, many manual activities are no longer necessary and are replaced by automated processes. Auto-scaling reduces the need for continuing service monitoring to increase or decrease the scale, reducing maintenance costs and fines for SLA breaches for businesses. This research presents an 'Event Based Proactive Auto-Scaling' model to be used in cloud applications that minimizes the service level agreement violation and at the same time minimizes the renting cost for the application providers of Bangladesh and all over the world.

Therefore, the outcomes of the proposed project will contribute significantly to the auto-scaling approach for discovering and analyzing information from the background of the decision-making parameter to the areas of information science and network engineering. Therefore, it will also help to promote the level of researches of cloud computing researchers and enriched skilled faculties.

6. Objective(s) of the Project:

An event-based proactive method for cloud instances is proposed in this project. The main research objectives of this project are to:

- Do fundamental research in the area of cloud computing, especially, in the theory of auto scaling.
- Develop a forecasting algorithm that observes and predicts resource usage based on the decision-making parameters' past behavior.
- Design an event-driven auto-scaling architectural model based on MAPE (Monitor, Analysis, planning, Execution) loop.
- Predict resource usage by monitor SLA aware and Resource aware auto-scaling in cloud system

7. Methodology:

The core model will be designed using heuristic called triple exponential smoothing (TES). Firstly, TES can smooth trends and seasonality in time series (Billah et al., 2006) (Singh et al., 2019). Secondly, TES is lightweight which is one of the constraints and TES can also be highly adaptive by tuning its constants. Upper interval predictions will be used to keep surplus resources and Fibonacci weighted moving average of upper interval of predictions will be used to deal with traffic fluctuations.

The cloud providers provide IaaS, PaaS and SaaS services to their customers. The cloud providers would use proposed proactive auto scaling component and scale their service instances for better user experience. The figure 1 indicates where the proactive auto-scaling module can be used in cloud architecture.

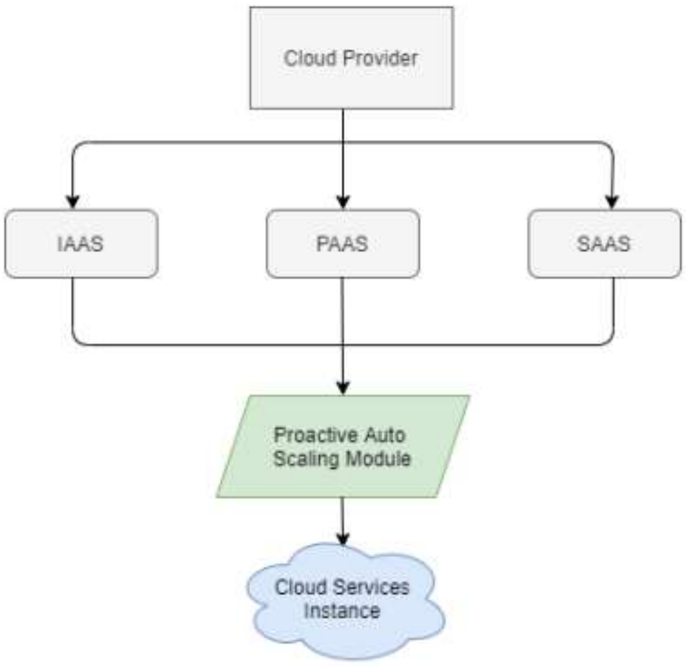

Figure 1: PASC Integration in Cloud Environment

To achieve the goal of this project we have to develop forecasting algorithm along with following components. The detail of each component are describe below.

i) Monitor Component:

Monitor component monitors prediction parameters from the system. These prediction parameters are CPU utilization, VMs Count, Throughput, Response Time, Delay Time, SLA violation count, SLA violation percentage, SLA violation time, Falied Cloudlets, and Workload. These prediction parameters will be collected from the system and added to the history database by this component.

ii) Analyzer Component:

Analyzer component collects recent data from the history database for three different windows. This component analyzes data of ten prediction parameters collected by the monitor component in the previous stage. This component uses an adaptive approach to tune the constants for triple exponential smoothing.

iii) Planner Component:

Planner component takes prediction event from Prediction Event Queue as input and use them to take scaling decisions accordingly. This component first calculates the SLA threshold and then the planner component decides scaling event (scale down or up) using this value

From the *PredictionEventQueue* queue the model gets the *predictionEvent* object that will be generated in the analyzer component. This *predictionEvent* contains information about SLA identifier. Then the proposed model collects the SLA for that *predictionEvent* using SLA identifier. This SLA contains all the information about the degree of service expected by a consumer from a cloud service provider. After collecting the SLA for the *predictionEvent*, the comparison between the SLA threshold with the prediction will be made. If the prediction is less than threshold, the proposed model calculates the number of scaled-down decision. After that the proposed model generates *decisionEvent* for each scale down decisions. Otherwise, the proposed model generates *decisionEvents* for each scale up decision in the same process. The generated *decisionEvents* events will be pushed to the *decisionEventQueue* to use as input in the later component.

iv) Normalizer Component:

Normalizer component takes value from *decisionEventQueue* as input and normalize them. At the first step, the proposed model pops the first decision event and then peeks the second decision event from the queue. In the second step, the model checks whether the first and the second decision events are contradictory or not. If they are contradictory decisions and the scaling time difference between them is less than *contradictoryDET* (contradictory Decision Event Time) then they practically eliminate each other. In this scenario, the component pops second event from the queue and does not push any event to the normalized decision event queue. In case both the decision events are not contradictory or the scaling time difference between them is greater than *contradictoryDET*, then the component pushes the first event to the normalized decision queue.

v) Executor Component:

The executor component checks the events from the *decisionEventQueue*. If the event is scale-up, this component performs scale up in the cloud service instance. Otherwise, the component first figures out which cloud instance will be taken down first. After that, the component scales down that selected cloud instance.

Therefore, the methodology of the proposed research comprises the following.

- Surveying articles containing the existing results on auto-scaling methods in cloud computing.
- Development of forecasting algorithmic methods to observes and predicts resource usage based on the decision-making parameters' past behavior.
- Development of algorithmic methods for generating Monitor Component, Analyzer component, Planner component, Normalizer component and Executor component.
- Establishing the exactness and the time-space complexity of these algorithms theoretically.
- Implementation of the algorithm into the computer by developing JAVA codes for computing numerical results.
- Involving our graduate students and/or consultants who have the related knowledge to assist us if needed.

References:

Ajila, S. and Bankole, A. (2016). Using machine learning algorithms for cloud client prediction models in a web vm resource provisioning environment. *Transactions on Machine Learning and Artificial Intelligence*, 4.

Aslanpour, M. S. and Dashti, S. (2017). Proactive auto-scaling algorithm (pasa) for cloud application. *International Journal of Grid and High Performance Computing*, 9:1–16.

Aslanpour, M. S. and Dashti, S. E. (2016). Sla-aware resource allocation for application service providers in the cloud. *In 2016 Second International Conference on Web Research (ICWR)*, pages 31–42.

Assuncao, M., Cardonha, C., Netto, M., and Cunha, R. (2015). Impact of user patience on auto-scaling resource capacity for cloud services. *Future Generation Computer Systems*, 55.

Billah, B., King, M., Snyder, R., and Koehler, A. (2006). Exponential smoothing model selection for forecasting. *International Journal of Forecasting*, 22:239–247.

Garc´ıa Garc´ıa, A., Blanquer Espert, I., and Hern´andez Garc´ıa, V. (2014). Sla-driven dynamic cloud resource management. *Future Gener. Comput. Syst.*, 31:1–11.

Ghobaei-Arani, M., Jabbehdari, S., and Pourmina, M. (2016). An autonomic approach for resource provisioning of cloud services. *Cluster Computing*, 19.

Gill, S. S. and Chana, I. (2015). Q-aware: Quality of service based cloud resource provisioning. *Computers Electrical Engineering*, 47.

Huang, J., Li, C., and Yu, J. (2012). Resource prediction based on double exponential smoothing in cloud computing. In *2012 2nd International Conference on Consumer Electronics, Communications and Networks (CECNet)*, pages 2056–2060.

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

KEMP, S. (2020). Digital 2020: *Global digital overview*.

Lorido-Botr´an, T., Miguel-Alonso, J., and Lozano, J. (2013). Comparison of auto-scaling techniques for cloud environments.

Makridakis S, Spiliotis E, A. V. (2018). Statistical and machine learning forecasting methods: Concerns and ways forward.

Mart´ınez, R. G., Li, Z., Lopes, A., and Rodrigues, L. (2017). Augure: Proactive reconfiguration of cloud applications using heterogeneous resources. In *2017 IEEE 16th International Symposium on Network Computing and Applications (NCA)*, pages 1–8. IEEE.

Mohamed, M., Amziani, M., Bela¨ıd, D., Tata, S., and Melliti, T. (2014). An autonomic approach to manage elasticity of business processes in the cloud. *Future Generation Computer Systems*, 50.

Molt´o, G., Caballer, M., and de Alfonso, C. (2016). Automatic memory-based vertical elasticity and oversubscription on cloud platforms. *Future Gener. Comput. Syst*., 56(C):1–10.

Qavami, H., Jamali, S., Akbari, M., and Javadi, B. (2014). Dynamic resource provisioning in cloud computing: A heuristic markovian approach. pages 102–111.

Qu, C., Calheiros, R. N., and Buyya, R. (2018). Auto-scaling web applications in clouds: A taxonomy and survey. *ACM Comput. Surv.*, 51(4).

Singh, K., Shastri, S., Bhadwal, A., Kour, P., Kumari, M., Sharma, A., and Mansotra, V. (2019). Implementation of exponential smoothing for forecasting time series data.

Vel´asquez, J., Zambrano, C., and Franco, C. (2013). A comparison of exponential smoothing and neural networks in time series prediction. *Dyna (Medellin, Colombia)*, 80:66–73.

Veni, T. and Saira Bhanu, M. (2016). Auto-scale: automatic scaling of virtualised resources using neuro-fuzzy reinforcement learning approach. *International Journal of Big Data Intelligence*, 3:145.

Xiao, Z., Song, W., and Chen, Q. (2013). Dynamic resource allocation using virtual machines for cloud computing environment. *IEEE Transactions on Parallel and Distributed Systems*, 24(6):1107–1117.

Ye, T., Guangtao, X., Shiyou, Q., and Minglu, L. (2017). An auto-scaling framework for containerized elastic applications. In *2017 3rd International Conference on Big Data Computing and Communications (BIGCOM)*, pages 422–430.

8. Expected output(s): Through this project, we expect the following outcomes.

- Obtaining some theoretical results related to Auto-Scaling Architecture in Cloud Application.
- Describing an event-driven auto-scaling architectural model based on MAPE (Monitor, Analysis, planning, Execution) loop.
- Developing a forecasting algorithm that observes and predicts resource usage based on the decision-making parameters' past behavior.
- Strengthening our skill in the applications areas of the Auto Scaling in Cloud Application and increasing the number of skilled researchers in the aforesaid areas.

9. Work Plan/ Activity Schedule: Our work plan comprises the following:

(1) Literature Survey
(2) Objective analysis
(3) Characterization of objective
(4) Mathematical Analysis and Architectural Design
(5) Algorithm design and development
(6) Performance Analysis
(7) Communication of the findings for report writing and publication

Activity Schedule:

Title of Activity	Timelines (12 Months) July 2022 - June 2023											
	1	2	3	4	5	6	7	8	9	10	11	12
Literature Survey	🟩	🟩	🟩									
Objective analysis			🟩	🟩	🟩							
Characterization of objective				🟩	🟩	🟩						
Mathematical Analysis and Architectural Design					🟩	🟩	🟩					
Algorithm design and development							🟩	🟩	🟩	🟩		
Performance Analysis									🟩	🟩	🟩	
Report writing and publication										🟩	🟩	🟩