Ingénierie des Systèmes d'Information
Vol. 30, No. 4, April, 2025, pp. 961-970
Journal homepage: http://iieta.org/journals/isi

Hybrid Workload Prediction for Improved Autoscaling in IaaS Clouds: An ARIMA-OLSTM Approach

Satya Nagamani Pothu*, Swathi Kailasam

Department of Computer Science and Engineering, Koneru Lakshmaiah Education Foundation, Guntur 522302, India

Corresponding Author Email: happysatyasai@gmail.com

https://doi.org/10.18280/isi.300413

Received: 5 February 2025
Revised: 15 April 2025
Accepted: 22 April 2025
Available online: 30 April 2025

Keywords:
auto scaling, workload prediction; scalability; infrastructure-as-a-service; resource utilization

ABSTRACT

Cloud computing's dynamic characteristics require precise prediction of workload and effective auto-scaling to optimize resource usage in Infrastructure-as-a-Service (IaaS) settings. To maximize auto-scaling, this research presents a robust hybrid workload prediction model that uses a hybrid pelican optimization algorithm (POA) for intelligent scaling decisions and Autoregressive Integrated Moving Average-Long Short-Term Memory (ARIMA-OLSTM) for accurate workload forecasting. ARIMA-OLSTM combines deep learning techniques with statistical methods. LSTM (optimized by RMSProp) learns non-linear, sequential information from ARIMA's residuals, while ARIMA represents the linear trends within historical workload sequences. The prediction accuracy is significantly increased by this two-step process. Resource scaling decisions are ideally determined during the planning stage by a Hybrid POA that is inspired by pelican hunting techniques and further improved using Lyrebird Optimization. Through constantly changing virtual machine parameters, it is strategically beneficial to find a balance between cost-effectiveness, system responsiveness, and SLA fulfillment. Extensive tests on realistic cloud workloads demonstrate that the suggested model outperforms existing models such as RHAS, GRASP, and ADA-RP, minimizing RMSE to 0.1513 and MAPE to 0.1557. Furthermore, compared to traditional methods, it maintains 50% less resource use and achieves a 70% reduction in reaction time, confirming the model's effectiveness, scalability, and prediction accuracy.

1. INTRODUCTION

Infrastructure-as-a-Service (IaaS) clouds, which provide scalable and flexible resources on demand, have modified the computing environment [1]. The growing dependence of companies on cloud infrastructures to guide their programs highlights the important need for powerful useful resource control structures. The largest one is Auto Scaling, which dynamically modifies the assets allocated in reaction to varying workload needs [2, 3]. The unpredictable nature of workloads makes it tough to obtain the most excellent Auto Scaling in IaaS clouds [4]. A hybrid workload prediction model that integrates proactive and reactive scaling methods is obtainable as a solution to this problem. By enhancing the effectiveness and responsiveness of auto-scaling methods, this model seeks to maximize aid utilization and minimize expenses.

Resource control in dynamic computing environments is being revolutionized through the hybrid workload prediction model for Auto Scaling in IaaS clouds, which creatively combines proactive and reactive scaling methodologies [5]. Proactive scaling allows for seamless adaptations to anticipated versions and maximizes aid efficiency. It is characterized by way of preemptive useful resource modifications based totally on predicted workload styles. Instead, reactive scaling ensures that overall performance isn't always disrupted by unplanned spikes or declines in demand by way of quickly adapting to unanticipated workload fluctuations in real time [6, 7]. The hybrid paradigm leverages the benefits of both proactive foresight and reactive model with the aid of skillfully fusing two techniques. The Auto Scaling system is enabled by this synergistic integration to anticipate changes in workload and modify efficiently, making sure the highest quality performance and aid allocation below a variety of operating situations [8]. The hybrid model creates a brand-new benchmark for agility and performance in cloud aid control by combining real-time responsiveness and predictive analytics dynamically. This permits organizations to navigate through the complexity of converting needs with unparalleled precision and resilience

The dynamic and unpredictable nature of cloud workloads won't be accurately captured via conventional workload prediction models, which frequently depend best on historic data or statistical forecasting strategies [9, 10]. Instead, the recommended hybrid model improves workload estimates' accuracy and dependability by combining the advantages of time-series analysis, predictive analytics, and machine-learning techniques. The hybrid model's ability to continually analyze historical data and adjust to changing workload patterns can result in more precise estimations that can be

obtained with greater confidence. Furthermore, the system can fast start reactive scaling sports in reaction to unexpected workload fluctuations due to the fact real-time monitoring and feedback mechanisms are covered [11]. This ensures the most efficient resource provisioning and overall performance balance

Therefore, an innovative step in cloud resource management has been made with the creation of a hybrid workload prediction model for Auto Scaling in IaaS clouds. This model provides a complete method to the issues presented by using workload variability and unpredictability by smoothly merging proactive foresight and reactive flexibility [12, 13]. In dynamic computing environments, cloud infrastructures may additionally allocate resources efficaciously, fulfill overall performance desires, and optimize cost efficiency because of the hybrid version's dynamic interplay of predictive analytics and actual-time responsiveness. Because of its adaptive studying abilities, workload forecasts can be constantly advanced, ensuing in particular and well-timed modifications to fulfill changing wishes [14]. In the continuously changing world of cloud computing, the hybrid workload prediction model, as a lighthouse of efficiency and innovation, attendants in a brand-new generation of efficient useful resource usage and operational resilience

The major contributions of the paper are as follows:

- The hybrid technique of the model, which combines the hybrid POA and the hybrid ARIMA with an optimized long short-term module (ARIMA-OLSTM), enables to maximization of useful resource utilization in cloud structures. Through precise forecasting of incoming demand and adaptive resource sizing, it ensures effective distribution of processing electricity and storage ability, resulting in monetary financial savings and more desirable provider durability
- The Monitoring, Analysis, Planning, Execution, and Knowledge phases of the version's framework facilitate better decision-making for IaaS carriers. It makes it feasible to balance commercial enterprise issues with Service Level Agreements (SLA) compliance using making well-timed and properly informed decisions about scaling up or down. This complements cloud-primarily based apps' standard overall performance and boosts user happiness
- The model shows the use of Mean-Absolute-Percentage-Error (MAPE) and Root-Mean-Square-Error (RMSE) to evaluate how correct workload estimates are. The version also evaluates reaction time and CPU utilization, presenting a thorough evaluation framework to affirm the efficacy of the prediction model and automobile-scaling alternatives in actual cloud environments.

The following sections are organized as follows: Section 2 explores relevant research and literature reviews, Section 3 introduces the proposed framework, Section 4 provides a detailed analysis of the observed results and discussions, and Section 5 offers the final assessment of this study.

2. LITERATURE REVIEW

This section reviewed some of the most recent research studies on reactive, and proactive-based workload prediction in IaaS CC settings.

A Robust Hybrid Auto-Scaler (RHAS) was proposed in 2021 for cloud-based web applications, utilizing threshold-based criteria, time series forecasting, and proactive/reactive auto-scaling strategies to dynamically allocate processing and storage resources [15]. The framework contains functions to safeguard user requests and responses to deal with safety concerns.

RHAS proved its effectiveness in maximizing performance and resource utilization by demonstrating a 14% cost reduction, notable improvements in response time, service level agreement (SLA) compliance, and consistent CPU utilization through experimentation with real-time web application workloads from NASA and ClarkNet.

FLAS is an auto-scaler for distributed systems that seamlessly integrates proactive and reactive scaling techniques [16]. To optimize scaling actions, FLAS leverages a reactive contingency system and predictive models for high-level metrics trends. When FLAS was implemented for a content-based publish-subscribe middleware, it was capable of minimizing instrumentation and obtaining performance necessities greater than 99% of the time, all while being adaptable to different programs.

An innovative proactive autoscaling technique was offered in 2023 [17] to improve the Quality of Service (QoS) of microservice installations in cloud environments. The method optimizes useful resource allocation via the use of a two-state machine-learning Random Forest (RF) version to estimate future CPU and memory usage values. The technique confirmed awesome discounts in end-to-end latency and resource consumption through validation with actual global workloads and deployment on a microservice prototype platform.

An approach for simulating auto-scaling mechanisms in cloud infrastructures utilizing stochastic Petri nets (SPN) and an adaptive search metaheuristic (GRASP) has been proposed [18]. The purpose of the technique is to locate the first-rate configurations to decrease costs and achieve service level agreements (SLAs), permitting higher operational management of cloud offerings.

An adaptive auto-scaling structure referred to as ADA-RP was presented in 2023 [19] to enhance useful resource provisioning in cloud computing environments. ADA-RP lowers prices and improves application overall performance by using dynamically automobile-scaling cloud resources in actual time based on expected workload calls using making use of beyond-time-series statistics.

A computational method for assessing the workload of microservices in cloud-native web programs was created in 2023 [20]. The method minimized scaling techniques and progressed useful resource allocation efficiency while upholding Quality of Service (QoS) requirements via a multi-criteria selection-making mechanism.

The research [21] performed a comparative study in 2024 between new methods based on control theory and queuing theory and autoscaling solutions provided by vast cloud providers. To shed mild on the effectiveness of diverse autoscaling strategies, the examine set out to evaluate their overall performance in terms of resource utilization optimization and violations of carrier-level agreements (SLAs).

BIAS Autoscaler introduces a novel approach leveraging burstable instances alongside standard instances for efficient queuing management in cloud-based microservice workloads [22]. After accomplishing an intensive trial on the Google

Cloud Platform, BIAS Autoscaler proved that it may save expenses by up to 25% and increase useful resource performance using 42% when compared to the usage of conventional instances solely.

3. PROPOSED METHODOLOGY

The proposed Robust Hybrid Workload Prediction Model aims to address the difficult conditions of workload prediction in cloud computing, particularly for autoscaling in web programs. By leveraging previous workload strains, this model forecasts future workload on physical machines, enabling more efficient resource allocation and capacity planning. The key is to accurately estimate the given attribute for horizontal scaling, especially by taking into account the expected workload. Through this technique, IaaS providers can enhance company sustainability, reduce operational expenses, and optimize resource utilization inside cloud data centers. Figure 1 suggests the overall architecture of the proposed technique.

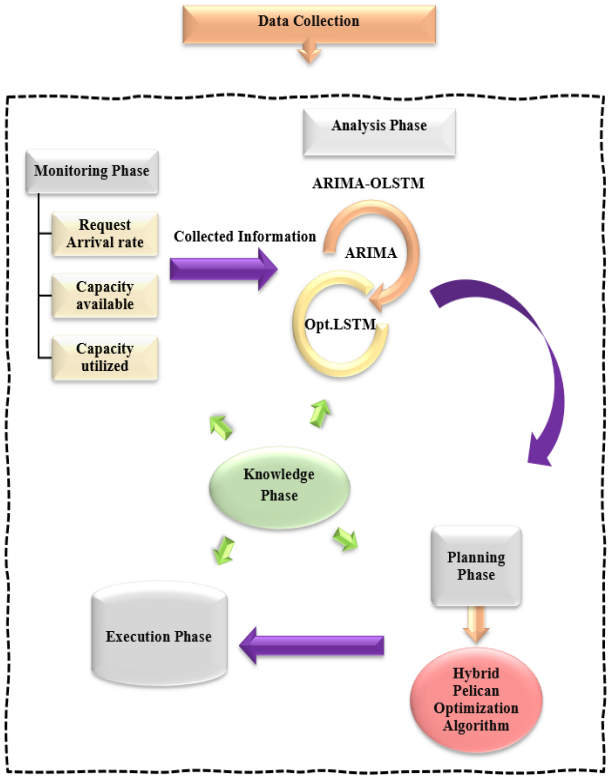

Figure 1. Overall architecture of the proposed methodology

3.1 Monitoring phase

Initially, this phase collects data often approximately the software and infrastructure stage parameters. The monitoring module records the request arrival rate, capacity available, and capacity utilized the use of the manipulated domain.

Request Arrival Rate: It is the pace at which tasks or requests are directed to the cloud. Monitoring this metric offers insights into workload intensity and demand fluctuations, supporting optimizing resource allocation and understanding performance dynamics inside cloud environments

Capacity Available: It denotes the entire resources like virtual machines, storage, and community bandwidth ready for deployment in the cloud infrastructure. Monitoring this metric gives visibility into resource availability, guiding decisions on scaling, and allocation, and making sure the maximum beneficial performance and responsiveness to various workloads in cloud environments.

Capacity Utilized: Capacity Utilized inside the usage of the manipulated area refers to the degree to which resources within the cloud infrastructure are currently being used. The control domain encompasses the management and monitoring mechanisms that oversee resource allocation and utilization. By monitoring capability utilization via this domain, cloud administrators can ensure efficient resource management, identify capability bottlenecks, and make knowledgeable selections regarding scaling, load balancing, and optimization techniques to maintain system performance and stability.

3.2 Analysis phase

In this phase, a hybrid method combining reactive and proactive strategies is delivered. Specifically, the ARIMA-OLSTM technique is hired. Additionally, reading CPU utilization and reaction time, it forecasts the high workload of one minute for the next scaling period. This integrated approach complements workload forecasting accuracy and allows powerful useful resource allocation in real-time cloud environments

One of the most broadly used linear regression strategies for stationary time collection forecasting is the Autoregressive integrated shifting common version (ARIMA) model. The forecasting model's structure is represented through the parameters $a, d,$ and m, which stand for auto-regression $Auto_{reg}(a)$, moving average Mov_{avg} (m), and differencing degree d. The version is expressed as ARIMA (a, m, d). The following is a description of the scientific formulation for ARIMA (a, m, d) in Eq. (1)

$$\left(1 - \sum_{i=1}^{a} \varphi_i \, l^i\right)(1 - l)^d x_t = \left(1 + \sum_{i=1}^{m} \theta_i \, l^i\right)\varepsilon_t \qquad (1)$$

where, ε_t are error terms, φ_i are the model's autoregressive part's [23] parameters, and θ_i are the Mov_{avg} part's parameters. l stands for the lag operator in this equation.

For accurate time series forecasting, Box and Jenkins proposed a three-step method for building an ARIMA model. Model identification is the first phase, which includes using differencing to make the series stable and analyzing ACF and PACF plots to determine ARIMA terms. The second phase, parameter estimation, finds the optimal model order by applying criteria like AIC and BIC. Diagnostic checking is the final phase, in which residuals are examined to verify the sufficiency of the model. This methodical technique strikes a balance between model accuracy and simplicity to deliver effective workload prediction in cloud systems. The LSTM model then receives the ARIMA residuals as input.

LSTM is a type of recurrent neural network (RNN) that works well with sequential data, like time series, since it can retain input memory over time. With their huge memory capacity, LSTMs combat the vanishing gradient problem, which hinders learning over lengthy sequences, which typically affects standard RNNs. They can learn from stimuli that are widely apart and develop long-term dependence. Three crucial gates that regulate data flow in a network forget, input, and output are used by LSTMs to do this.

Forget gate: Controls conditionally what data to remove from the block, from which the following is derived in Eq. (2).

$$f_t = \sigma\big(w_f[h_{t-1}, x_t] + b_f\big) \tag{2}$$

Input gate: Selects values to be inserted conditionally to update the memory state.

$$i_t = \sigma(w_i[h_{t-1}, x_t] + b_i) \tag{3}$$

Output gate: Determine output conditionally using the input and block memory.

$$o_t = \sigma(w_o[h_{t-1}, x_t] + b_o) \tag{4}$$

These three analog gates operate in the 0–1 range and are based on the sigmoid function. To drive the values to be between -1 and 1, a tanh function is used to calculate the input features at every time *t* using input x_t, prior hidden state h_{t-1} as follows in Eq. (5)

$$\tilde{c}_t = tanh(w_c[h_{t-1}, x_t] + b_c) \tag{5}$$

The adjusted input characteristics along with the partial decay of the previous memory cell contribute to the modification of the memory cell, resulting in Eq. (6).

$$c_t = f_t c_{t-1} + i_t \tilde{c}_t \tag{6}$$

In the end, memory c_t and output gate o_t compute the hidden output state h.

$$h_t = o_t * tanh\,(c_t) \tag{7}$$

In Eqs. (2)-(7), matrices weight matrices are w_c, w_f, w_i and w_o; the bias vectors are b_c, b_f, b_i, and b_o.

Using RMSProp to optimize the LSTM model significantly boosts performance by dynamically adjusting each parameter's learning rate using the moving average of squared gradients. The complex architecture of LSTM makes this technique very helpful since it allows precise parameter adjustment, prevents vanishing gradients, and offers reliable, effective training on sequential data. The RMSProp method updates the weight matrices w_c, w_f, w_i and w_o, and bias vectors b_c, b_f, b_i, and b_o. The Adaptive Gradient technique serves as the foundation for RMSProp, a learning rate adaptation technique that reduces the computational expense of neural network training. It works particularly well at resolving vanishing gradients in RNNs, notably LSTMs. In deep or complex LSTM structures, disappearing or exploding gradients can still occur. Techniques like gradient clipping, learning rate adjustment, decay adjustment, and appropriate selection of batch sizes are used to combat this. Regularization methods that increase generalization and avoid overfitting include L2 regularization and dropout. Better performance on sequential tasks and better control of gradient difficulties are guaranteed by proper hyperparameter tuning of both RMSProp and LSTM parameters. Initialize the collected gradient, Squared gradient accumulation for every parameter in Eq. (8).

$$E_t = 0 \tag{8}$$

Repeated until most iterations or convergence, determine the objective function's gradient approximately the parameters in Eq. (9)

$$g_t = \nabla_\theta J(\theta_t) \tag{9}$$

Revise the squared gradients' exponentially weighted average in Eq. (10) and replace the parameters in Eq. (11)

$$E_t = \gamma E_{t-1} + (1-\gamma) g_t^2 \tag{10}$$

$$\theta_{t-1} = \theta_t - \alpha \frac{g_t}{\sqrt{E_t - \epsilon}} \tag{11}$$

where, θ is an initial parameter, the learning rate is α, the decay factor is γ, g_t is the loss function's gradient at time t concerning the parameters, E_t is the average of the squared gradients weighted exponentially, a small constant called ϵ keeps division by zero from happening.

Algorithm 1. Pseudocode for ARIMA-OLSTM

Input: Time series data (*x*), ARIMA operators (*a*, *m*, *d*), learning rate (α), decay rate (γ), ϵ
Output: Trained ARIMA coefficients (φ, θ), Trained LSTM parameters (the weights w_c, w_f, w_i and w_o; and the bias vectors b_c, b_f, b_i, and b_o)
Initialize ARIMA-OLSTM parameters and hyperparameters.
Initialize RMSProp parameters (α, γ, ϵ).
ARIMA-OLSTM Hybrid Model Development
Prepare the time series data and do some preprocessing
Create an ARIMA model by applying the Box-Jenkins technique
Utilizing stationarity and ACF/PACF analysis, identify the type of model.
Utilise AIC/BIC metrics to estimate parameters (Eq. (1)).
Use residual analysis to validate the model.
Obtain the ARIMA residual time series for the LSTM input.
Training LSTM Models with RMSProp
Set the cumulative gradient E_t (Eq. (8)) to zero for each parameter.
Continue until convergence is reached.
Determine the objective function's gradient, g_t, using (Eq. (9))
Update the squared gradients' exponentially weighted average, or E_t (Eq. (10))
Use the RMSProp update rule (Eq. (11)) to update the model's parameters.
Rule for RMSProp Updates (Eq. (11))
def RMSProp_Update(θ_t, g_t, E_t)

$$E_t = \gamma E_{t-1} + (1-\gamma) g_t^2$$
$$\theta_{t-1} = \theta_t - \alpha \frac{g_t}{\sqrt{E_t - \epsilon}}$$

return θ_t, E_t

3.3 Planning phase

The analysis phase assesses the existing situation, and this makes decisions about scaling to balance advantage and SLA compliance. The hybrid POA, which integrates cooperative foraging behaviors with optimization algorithms to direct effective resource allocation and decision-making in complicated situations, is used in this decision-making process.

3.3.1 Hybrid POA and lyrebird optimization algorithm

The huge pelican uses a big pouch in its gullet to catch and eat targets. It also has a long beak. This species, which inhabits groups of several hundred pelicans, enjoys socializing and living in groups. Pelicans are large birds with a height of 1.06 to 1.83m, a wingspan of 0.5 to 3m, and a weight range of 2.75 to 15kg. They mainly eat fish, but also frogs, crustaceans, and

turtles when necessary. Pelicans cooperate in hunting, and diving from heights of 10-20 m or lower. They use their wings to corral fish to shallow waters for easier catching. Their hunting process involves efficient water removal before swallowing the fish. This behavior showcases their intelligence and hunting skills. The strategy of the proposed POA is inspired by the hunting strategy of pelicans.

Mathematical Model: POA is a population-based algorithm with pelicans as population followers proposing candidate solutions. Each member suggests values for variables based on their hunt space position. Primarily, followers are erratically prepared within problem bounds using Eq. (12).

$$p_{i,j} = L_j + r \cdot (U_j - L_j), \text{ i=1,2…, n, j=1, 2, …, m} \quad (12)$$

Value of variables, denoted by $p_{i,j}$ in ith candidate solution. n represents the population size, m is the number of variables, r is a random number between 0 and 1, L_j is the lower bound of jth variable, and U_j is the upper bound of jth variable. A population matrix in Eq. (13), for pelicans in the POA, is used to identify population members. Candidate solutions are represented by rows, and problem variable values are represented by columns.

$$P = \begin{bmatrix} P_1 \\ \vdots \\ P_i \\ \vdots \\ P_n \end{bmatrix}_{n\times m} = \begin{bmatrix} p_{1,1} & \cdots & p_{1,j} & \cdots & p_{1,m} \\ \vdots & \ddots & \vdots & \iddots & \vdots \\ p_{i,1} & \cdots & p_{i,j} & \cdots & p_{i,m} \\ \vdots & \iddots & \vdots & \ddots & \vdots \\ p_{n,1} & \cdots & p_{n,j} & \cdots & p_{n,m} \end{bmatrix}_{n\times m} \quad (13)$$

where, the pelican population matrix is P and P_i represents ith pelican. One possible solution for the mentioned problem is to make every person in the population of the proposed POA a pelican. Consequently, each of the potential solutions can be utilized to assess the objective function of the specified problem. Eq. (14) uses a vector known as the objective function vector to derive values obtained for the objective function.

$$Fn = \begin{bmatrix} Fn_1 \\ \vdots \\ Fn_i \\ \vdots \\ Fn_n \end{bmatrix}_{n\times m} = \begin{bmatrix} Fn(P_1) \\ \vdots \\ Fn(P_i) \\ \vdots \\ Fn(P_n) \end{bmatrix}_{n\times 1} \quad (14)$$

where, the ith candidate solution's objective function value is Fn_i and Fn is the objective function vector.

Phase 1: Approaching the target

The pelicans have to locate their prey and then move in its direction during the first step. Modeling this pelican technique enables search space scanning and the exploration capabilities of the proposed POA in identifying different search space locations. The fact that the prey's location is randomly generated inside the search space is essential to POA. As a result, POA has more exploration capacity while precisely searching the problem-solving domain. Eq. (15) provides a mathematical simulation of the above-mentioned thoughts and the pelican's method of finding prey.

$$p_{i,j}^{peli} = \begin{cases} p_{i,j} + r \cdot (prey_j - k \cdot p_{i,j}), & Fn_y < Fn_i; \\ p_{i,j} + r \cdot (p_{i,j} - prey_i), & else, \end{cases} \quad (15)$$

where, k is an arbitrary number between one and two, Prey is located in jth dimension, $p_{i,j}^{peli}$ is the ith pelican's new rank in jth dimension based on phase 1, and Fn_y is its objective function value. A number, parameter k, has a random value of either 1 or 2. For every member and iteration, a random parameter is chosen. This parameter's value of two causes a member to be further displaced and might yield them into previously unexplored areas of search space. Consequently, parameter k affects POA's ability to discover and precisely scan search space. If the pelican's new position improves the goal function's value, the new position is accepted. This kind of updating, known as efficient updating, keeps the algorithm from going to suboptimal places. Eq. (16) is used to model this process.

$$P_i = \begin{cases} P_i^{pos_1}, & Fn_i^{Peli_1} < Fn_i; \\ P_i, & else \end{cases} \quad (16)$$

where, $P_i^{pos_1}$ is the objective function value founded on phase 1 and $Fn_i^{Peli_1}$ is the new position of ith pelican.

Phase 2: Flying above the water's surface

During the second phase, the pelicans expand their wings to drive the fish higher and gather the food into their throat pouch once they reach the water's surface. In the area they are attacking, pelicans can catch more fish by using this approach. The recommended POA converges to more opportune places inside the hunting zone as a result of modeling this pelican behavior. This strategy improves POA's local search capability and exploitation possibilities. Mathematically speaking, the method needs to consider the points surrounding the pelican position to converge to an ideal solution. Eq. (17) provides a mathematical simulation of pelican hunting behavior.

$$p_{i,j}^{pos_2} = p_{i,j} + H \cdot \left(1 - \frac{itr}{max_{itr}}\right) \cdot (2 \cdot r - 1) \cdot p_{i,j} \quad (17)$$

where, H is a constant equivalent to 0.2, $H \cdot \left(1 - \frac{itr}{max_{itr}}\right)$ is the locality range of $p_{i,j}$, itr is the iteration clock, and the maximum number of iterations is max_{itr}. Based on phase 2, $p_{i,j}^{pos_2}$ represents the new position of ith pelican in jth dimension. The coefficient $H \cdot \left(1 - \frac{itr}{max_{itr}}\right)$ represents the radius of the population members' local neighborhoods to search near each other to converge on a better answer. This coefficient can be effectively applied to the POA exploitation power to approximate the optimal global solution. More space is considered around each member in the initial iterations due to the high value of this coefficient. The neighborhood radii of each member decrease as the procedure replicates more, due to a decrease in coefficient. For the POA to approach the global (and even precisely global) optimal solutions given the utilization notion, this allows us to scan the region encircling each individual in the population using increasingly smaller and more accurate steps. In certain situations, the coordination of agents within the algorithm may result in overhead and complexity, which could affect its scalability and efficiency. This is particularly true for difficult or high-dimensional optimization problems. Thus, the improvement for POA is enhanced by the lyrebird optimization algorithm (LOA).

The population member's location in the search space is altered throughout this stage of LOA by the lyrebird's modeled strategy of hiding in its immediate safe region. The potential application of LOA in local searches is demonstrated by the lyrebird's positional changes as it moves in little steps to find

a decent hiding location and accurately checks its surroundings. Using Eq. (18), each LOA member's new position is found in the LOA design based on the lyrebird's migration model toward the closest suitable place for concealment. The related member's previous position is replaced if, by Eq. (19), this new placement increases the value of the objective function.

$$x_{i,j}^{pos} = x_{i,j} + \left(1 - 2r_{i,j}\right).\frac{U_j - L_j}{itr} \tag{18}$$

$$P_i = \begin{cases} P_i^{pos_2}, & Fn_i^{pos_2} \leq Fn_i \\ P_i, & else, \end{cases} \tag{19}$$

In this case, $P_i^{pos_2}$ is a new position determined for ith lyrebird using suggested LOA's concealing method; $P_i^{pos_2}$ is its jth dimension; $Fn_i^{pos_2}$ is the value of its objective function; $r_{i,j}$ are random values from interval [0, 1]; and the iteration counter is itr.

Algorithm 2: Pseudocode for H-PLA

Initialize Parameter P →variable, Fn → Function
Phase 1: Approaching the target
def phase_1_pelican_approach (P, Fn, $prey$, max_{itr})
  for i in range (max_{itr})
    for pelican in $peli$
      r=random. uniform (0, 1)
 k=random. choice ([1, 2])
      for j in range(len(pelican)):
        if $Fn_y < Fn_i$

$$p_{i,j} + r \cdot \left(prey_j - k \cdot p_{i,j}\right)$$

        else

$$p_{i,j} + r \cdot \left(p_{i,j} - prey_i\right)$$

  return P
Phase 2: Flying above the water's surface
def phase_2_pelican_flying (P, max_{itr}):
H=0.2
  for i in range (max_{itr}):
    for pelican in P
r=random. uniform (0, 1)
      for j in range(len(pelican))

$$p_{i,j}^{pos_2} = p_{i,j} + H \cdot \left(1 - \frac{itr}{max_{itr}}\right) \cdot (2 \cdot r - 1) \cdot p_{i,j}$$

$$P_i = \begin{cases} P_i^{pos_2}, & Fn_i^{pos_2} \leq Fn_i \\ P_i, & else, \end{cases}$$

return P
Hybrid Pelican Optimization Algorithm (POA)
def hybrid_POA(P , Fn , $prey$, $max_{itr}Phase\ 1$, $max_{itr}Phase\ 2$)

$P_{phase1} = P_{phase1}^{peli}$ (P, Fn, $prey$, $max_{itr}Phase\ 1$)
$P_{phase2} = P_{phase2}^{peli}$ (P, Fn, $prey$, $max_{itr}Phase\ 2$)
return P_{phase2}

SLA compliance and workload for the VM will be balanced, and resource scaling will play a major role in this process during the planning phase. The POA, which draws inspiration from the cooperative hunting behaviors of pelicans, will be employed in this phase. Before evaluating each configuration in terms of resource utilization and reaction time, an objective function was used to set the CPU, memory, and storage configurations for each virtual machine at random within the constraints of the task. Pelicans (VM configurations), fly over the search space in the first phase and adjust their placements to achieve the best possible resource utilization. The new configuration is approved if it has a better objective function. The algorithm's second step fine-tunes the configurations by taking the local search space into account and making minor tweaks to increase efficiency. To guarantee that the system operates close to optimal conditions, the LOA further fine-tunes the setups. In cloud systems, this will result in efficient resource allocation and SLA compliance.

3.4 Execution phase

The Execution Phase is based on the planner's interpretations, which are critical to the system's subsequent stages. There comes a point when a definitive choice is made on whether to scale up, scale down, or maintain the status of things, followed by a formal request to the CP for approval. The default executor will select computers at random from the resource pool available for execution during scale-up or scale-down. As part of this phase's validation, a detailed examination will be performed to ensure that the limit of on-demand virtual machines is not exceeded before making a scale-up decision. If the limit is exceeded, the request for scale-up will be rejected, this is critical in adhering to set limits. Furthermore, once the on-demand has reached zero, all subsequent scaling requests will be deemed unnecessary and will not be fulfilled. In contrast, to execute dynamic resource management in the cloud, the algorithm is implemented in this phase using the techniques outlined below.

It uses algorithms such as the POA to make intelligent resource allocation decisions. The POA will balance the exploration and exploitation phases to obtain the best virtual machines and their utilization in a way that allows for optimal resource allocation while respecting the limits and constraints of the cloud environment, avoiding unwanted scaling, and ensuring system efficiency.

3.5 Knowledge phase

This phase is centered on the knowledge capture, preservation, and application of lessons learned during the process loop to individuals. The preceding stage's responsibilities included collecting raw data during monitoring and completing evaluations. This process may yield findings, patterns, best practices, or, in some cases, identified errors. This knowledge must be organized in a way that makes it easily accessible to the people. This data can be stored on a collaborative platform, in project management tools, or in a centralized knowledge management system. The knowledge base should be searchable and well-structured, with efficient retrieval achieved by usable naming conventions, tagging, and categorization.

In a CC setting, this entails applying algorithms to the data available. Some of the primary functions are performed by Hybrid Pelican Optimisation Algorithm and Long Short-Term Memory networks. POA, for example, can help with resource allocation optimization because pelicans prioritize seeking new resources over exploiting established ones. Instead, LSTM networks aid in data processing and sequential prediction, with the vanishing gradient issue well addressed by their long-term memory capability.

At this stage, more collaboration is needed to help algorithms develop strength in decision support and overall efficacy in the cloud environment. These allow it to learn from previous data, adapt to changing conditions over time, and

make the best use of resources to guarantee the system performs optimally, all while adhering to service level agreements.

4. RESULT AND DISCUSSION

The configuration and simulation settings which include a range of scenarios meant to assess and analyse the outcomes are explained in the sections that follow. The suggested approach is compared against state-of-the-art methods such as RHAS [15], GRASP [18], and ADA-RP [19] and proposed. Evaluation metrics including Root-Mean-Square Error (RMSE) and Mean-Absolute-Percentage-Error (MAPE) also evaluated the CPU utilization and response time.

The suggested approach has been applied to the usage of Python 3.10 on the Google Colab coding platform and simulated on an Intel Core i3 Processor strolling Windows 10 with 8 GB of RAM. The configuration and simulation settings which consist of more than a few scenarios supposed to assess and analyse the results are defined in the sections that follow. The cautioned approach is compared to new techniques consisting of RHAS [15], GRASP [18], and ADA-RP [19] proposed. Evaluation metrics consisting of RMSE and MAPE additionally evaluated the CPU utilization and response time

4.1 Performance metrics

Evaluation metrics along with MAPE and RMSE additionally evaluated the CPU utilization and response time.

- **RMSE:** The RMSE between the discovered and actual values is the size of the variation between the 2.

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N} \|y(i) - \hat{y}(i)\|^2}{N}} \quad (20)$$

where, i is the variable, N denotes the non-statistics lacking point, $y(i)$ is the actual remark time series, and $\hat{y}(i)$ is the envisioned time series.

- **MAPE:** A statistical forecasting technique's prediction accuracy is gauged by way of the MAPE.

$$MAPE = \frac{1}{N}\sum_{i=1}^{n}\left|\frac{y_e(i) - y_a(i)}{y_a(i)}\right| * 100 \quad (21)$$

where, $y_e(i)$ is the real cost, $y_a(i)$ is the forecast value.

4.2 Performance analysis

It provides an overall performance analysis that sets the suggested model against current ones in terms of RMSE, MAPE, CPU utilization, and response time.

Table 1 is a comparison of the performance of four models (RHAS, GRASP, ADA-RP, and Proposed) with different VM machine configurations. With an increase in VM resources from 10 to 50, all models improve in RMSE, MAPE, and response time. For RHAS, RMSE goes down from 0.2738 to 0.2498, MAPE from 0.2914 to 0.2575, and prediction accuracy improves from 97.26% to 97.53%. GRASP indicates RMSE from 0.3281 to 0.2818, MAPE from 0.3024 to 0.2710, and prediction accuracy from 96.67% to 96.8%. The RMSE of ADA-RP reduces from 0.3016 to 0.2815, MAPE from 0.2678 to 0.2254, and prediction accuracy rises from 96.74% to 96.85%. The Proposed Model performs better than all the rest, with RMSE reducing from 0.2078 to 0.1513, MAPE from 0.1816 to 0.1556, and prediction accuracy increasing from 98.05% to 98.45%.

Figure 2 shows the RMSE values, which show accurate predictions, varied from 0.273874 to 0.251098. The MAPE values demonstrated remarkable prediction accuracy, ranging from 0.181694 to 0.291473, with the lowest value at 40 virtual machines. With 40 virtual machines, the CPU utilization reached a peak of 9.35% as machine size rose. The trade-off between accuracy and reaction time in auto-scaling systems is highlighted by the fact that response times increased with VMs, taking 148.54 seconds with 50VMs.

Figure 3 shows MAPE values exhibit a range of 0.27104 to 0.30245, indicating that 50 virtual machines yielded the best results. As the number of virtual machines rises, utilization numbers climb from 21.57% to 108.21%, demonstrating a trade-off between accuracy and resource utilization. In general, more virtual machines increase accuracy but also use more resources.

Table 1. Overall performance analysis based on existing models

Model	VM Machine	RMSE	MAPE	Utilization	Response Time	Prediction Accuracy (%)
	10	0.2738	0.2914	0.0188	20.45	97.26
	20	0.2641	0.2574	0.021966	41.38	97.36
RHAS [15]	30	0.2557	0.2334	0.020516	78.34	97.44
	40	0.2510	0.1816	0.0935	124.41	97.52
	50	0.2498	0.2575	0.01256	148.54	97.53
	10	0.3281	0.3024	0.02424	21.57	96.67
	20	0.3110	0.2976	0.01474	42.42	96.69
GRASP [18]	30	0.3004	0.2916	0.02096	75.28	96.7
	40	0.2981	0.2875	0.04096	98.15	96.72
	50	0.2818	0.2710	0.06096	108.21	96.8
	10	0.3016	0.2678	0.03542	19.42	96.74
	20	0.2990	0.2543	0.056487	38.52	96.75
ADA-RP [19]	30	0.2904	0.2447	0.02465	61.42	96.8
	40	0.2892	0.2389	0.025463	85.47	96.82
	50	0.2815	0.2254	0.036214	98.24	96.85
	10	0.2078	0.1816	0.003376	12.41	98.05
	20	0.1988	0.1743	0.004606	24.23	98.11
Proposed	30	0.1853	0.1712	0.0012462	30.52	98.21
	40	0.1626	0.1654	0.0010335	41.06	98.34
	50	0.1513	0.1556	0.009344	50.36	98.45

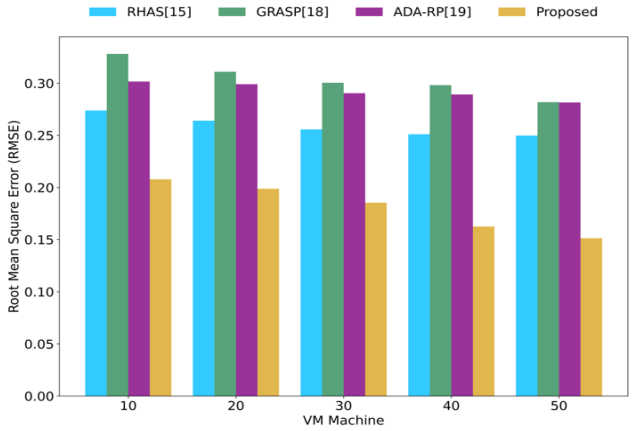

Figure 2. Graphical representation of the RMSE analysis

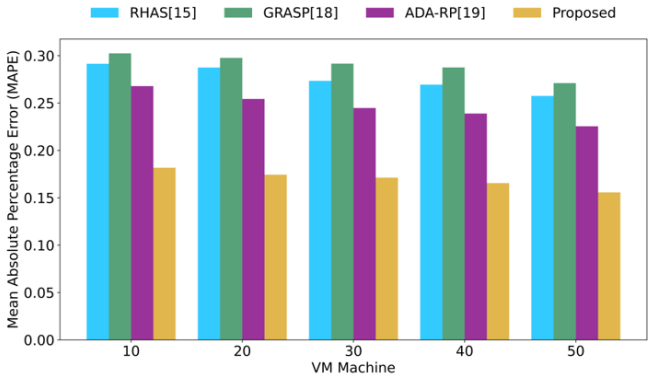

Figure 3. Graphical representation of the MAPE analysis

Figure 4 shows the algorithm's accuracy in estimating workload was indicated by the utilization time, which varied from 0.281581 to 0.301633. The algorithm's performance was illustrated in terms of percentage error by MAPE values, which varied from 0.22545 to 0.26789. The utilization numbers, which show the effective utilization of resources, ranged from 0.02465 to 0.056487. The response time ranged from 19.42 to 98.24, demonstrating the adaptability of the algorithm to varying demand scenarios.

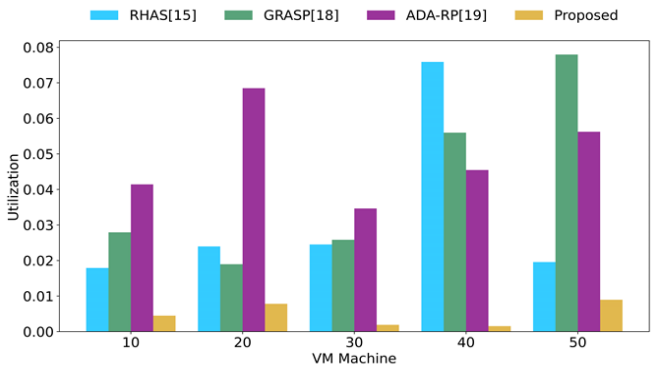

Figure 4. Graphical representation of the resource utilization analysis

In comparison to the RHAS model, Figure 5 demonstrates the suggested approach shows considerable gains in workload prediction accuracy across a range of virtual machine configurations. It is regularly found that the suggested model produces better accurate workload projections. The MAPE values, which range from 0.174367 to 0.009344, likewise show excellent precision. Notably, the MAPE stays low at 0.0010335, indicating small inaccuracy in percentage terms, while the RMSE lowers to 0.162614 for 40 virtual machines, indicating precise predictions. With a CPU utilization range from 0.003376 to 0.009344, it is evident that resources are being used efficiently. With 10VMs, reaction times grow progressively to 50.36 seconds for 50 VMs, indicating the anticipated trade-off between prediction accuracy and system response time in auto-scaling systems. Figure 6 shows the visual representation of the performance analysis for the prediction accuracy results.

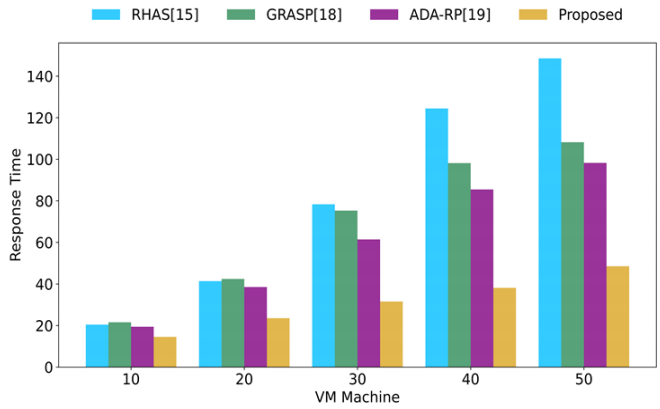

Figure 5. Graphical representation of the response time analysis

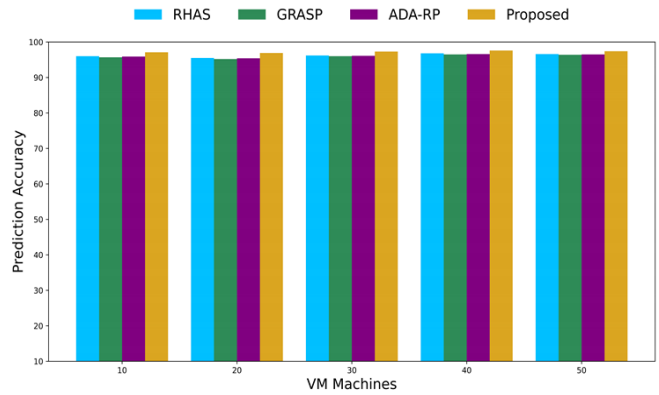

Figure 6. Graphical representation of the prediction accuracy analysis

The ANOVA test findings demonstrate that RMSE and MAPE differ significantly between groups, with incredibly low p-values (6.47E-09 and 4.66E-06, respectively) as shown in Table 2 and Figure 7. This suggests that the models or conditions assessed have a considerable impact on these measurements. Instead, there are no discernible variations in Response Time (p-value=2.12E-01) or Utilization (p-value=7.97E-02), indicating that these variables do not significantly differ between the groups and ANOVA distribution plot is also shown in the Figure 8.

Table 2. Performance analysis for the ANOVA test

Metric	F-Statistic	P-Value
RMSE	59.82962	6.47E-09
MAPE	23.10992	4.66E-06
Utilization	2.71018	7.97E-02
Response Time	1.675569	2.12E-01

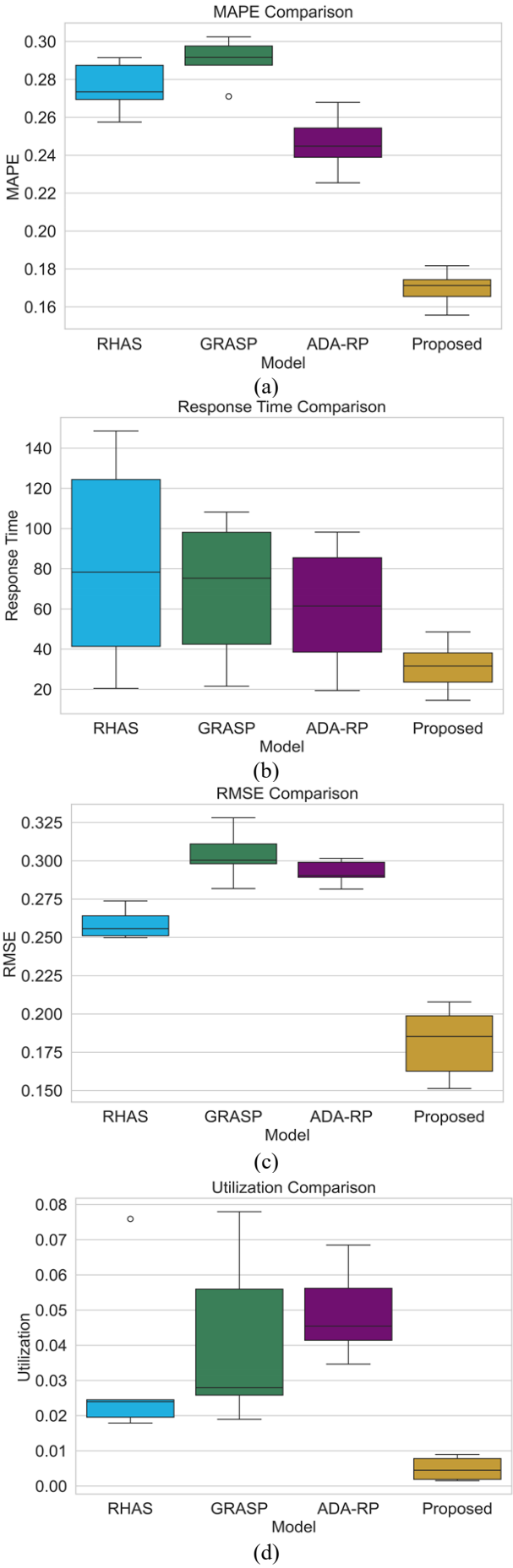

Figure 7. Graphical representation of the boxplot comparison. (a) MAPE, (b) Response time, (c) RMSE, and (d) Utilization

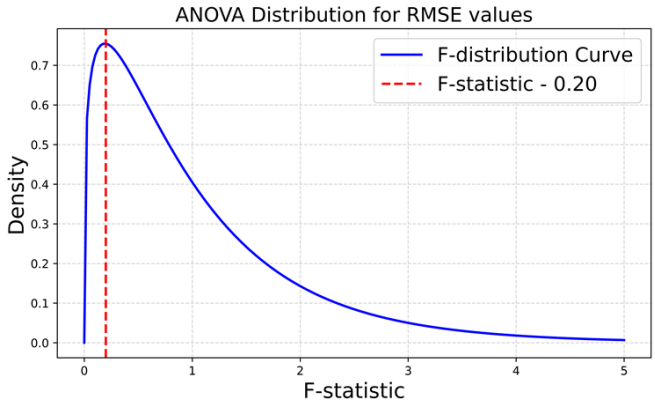

Figure 8. ANOVA test results for RMSE

5. CONCLUSION

For auto-scaling in web applications, a robust hybrid workload prediction model is developed using a comprehensive architecture that comprises phases for monitoring, analysis, planning, execution, and knowledge. With this approach, cloud computing companies can increase service sustainability, reduce operational costs, and better manage infrastructure resources. Time series forecasting, hybrid analysis techniques, and optimization algorithms such as H-PLA and ARIMA-OLSTM enable accurate workload prediction and dynamic auto-scaling decisions. This helps to maximize resource utilization, promote scalability, and improve compliance with SLAs while considering profit trade-offs. The assessment measures provide a robust validation framework for the prediction model's accuracy and the effectiveness of auto-scaling choices. The analysis of 50 node findings showed effective resource management, with a MAPE of 0.151318, RMSE of 0.155692, utilization time of 0.009344, and reaction time of 50.36 units. Future research will examine the application of the three novel models in a range of cloud contexts to assess the models' efficacy and presentation. Future studies might use Transformer-based models to identify temporal patterns more accurately, extend the model to edge-cloud hybrid systems, use reinforcement learning to make decisions in real-time, and evaluate performance across large, diverse datasets and workload profiles.

REFERENCES

[1] Dass, A.K., Parida, A., Panigrahi, S., Moharana, S.K. (2023). Virtualization in cloud computing: Transforming infrastructure and enhancing efficiency. Research and Applications: Emerging Technologies, 5(3): 26-40. https://doi.org/10.5281/zenodo.10300506

[2] Dittakavi, R.S.S. (2021). An extensive exploration of techniques for resource and cost management in contemporary cloud computing environments. Applied Research in Artificial Intelligence and Cloud Computing, 4(1): 45-61.

[3] Park, J., Jeong, J. (2023). An autoscaling system based on predicting the demand for resources and responding to failure in forecasting. Sensors, 23(23): 9436. https://doi.org/10.3390/s23239436

[4] Chouliaras, S. (2023). Adaptive resource provisioning in

cloud computing environments. Doctoral Dissertation, Birkbeck, University of London. https://doi.org/10.18743/PUB.00051213

[5] Santos, J., Wauters, T., Volckaert, B., De Turck, F. (2021). Towards low-Latency service delivery in a continuum of virtual resources: State-of-the-art and research directions. IEEE Communications Surveys & Tutorials, 23(4): 2557-2589. https://doi.org/10.1109/COMST.2021.3095358

[6] Tärneberg, W., Skarin, P. (2023). Constructive dissonance in the cloud: Adaptive out-of-phase scheduling for periodic tasks. In 2023 IEEE 12th International Conference on Cloud Networking (CloudNet), Hoboken, NJ, USA, pp. 103-111. https://doi.org/10.1109/CloudNet59005.2023.10490059

[7] Taha, M.B., Sanjalawe, Y., Al-Daraiseh, A., Fraihat, S., Al-E'mari, S.R. (2024). Proactive auto-Scaling for service function chains in cloud computing based on deep learning. IEEE Access, 12: 38575-38593. https://doi.org/10.1109/ACCESS.2024.3375772

[8] Katal, A., Sethi, V., Lamba, S. (2021). Virtual machine scaling in autonomic cloud resource management. Autonomic Computing in Cloud Resource Management in Industry 4.0. Springer, Cham, 301-323. https://doi.org/10.1007/978-3-030-71756-8_17

[9] Al-Sayed, M.M. (2022). Workload time series cumulative prediction mechanism for cloud resources using neural machine translation technique. Journal of Grid Computing, 20(2): 16. https://doi.org/10.1007/s10723-022-09607-0

[10] Alqahtani, D. (2023). Leveraging sparse auto-Encoding and dynamic learning rate for efficient cloud workloads prediction. IEEE Access, 11: 64586-64599. https://doi.org/10.1109/ACCESS.2023.3289884

[11] Pulle, R., Anand, G., Kumar, S. (2023). Monitoring performance computing environments and autoscaling using AI. International Research Journal of Modernization in Engineering Technology and Science, 5(5): 8934-8942. https://www.doi.org/10.56726/IRJMETS40883

[12] SILVA, P.R.P.D. (2019). A hybrid strategy for auto-Scaling of VMs: An approach based on time series and thresholds. Master's Thesis, Universidade Federal de Pernambuco.

[13] Chatzipanagiotou, D. (2024). Prediction-Based resource allocation in a rolling-Horizon framework considering activity prioritization.

[14] CODE, A.I.A. (2023) Dynamic autonomic systems: Augmenting infrastructure as code with machine learning for proactive and predictive scaling in complex it environments. Journal ID, 9339: 1263.

[15] Singh, P., Kaur, A., Gupta, P., Gill, S.S., Jyoti, K. (2021). RHAS: Robust hybrid auto-Scaling for web applications in cloud computing. Cluster Computing, 24(2): 717-737. https://doi.org/10.1007/s10586-020-03148-5

[16] Rampérez, V., Soriano, J., Lizcano, D., Lara, J.A. (2021). FLAS: A combination of proactive and reactive auto-Scaling architecture for distributed services. Future Generation Computer Systems, 118: 56-72. https://doi.org/10.1016/j.future.2020.12.025

[17] Al Qassem, L.M., Stouraitis, T., Damiani, E., Elfadel, I.A.M. (2023). Proactive random-forest autoscaler for microservice resource allocation. IEEE Access, 11: 2570-2585. https://doi.org/10.1109/ACCESS.2023.3234021

[18] Fé, I., Matos, R., Dantas, J., Melo, C., Nguyen, T.A., Min, D., Choi, E., Silva, F.A., Maciel, P.R.M. (2022). Performance-cost trade-off in auto-scaling mechanisms for cloud computing. Sensors, 22(3): 1221. https://doi.org/10.3390/s22031221

[19] Chouliaras, S., Sotiriadis, S. (2023). An adaptive auto-scaling framework for cloud resource provisioning. Future Generation Computer Systems, 148: 173-183. https://doi.org/10.1016/j.future.2023.05.017

[20] ZargarAzad, M., Ashtiani, M. (2023). An auto-Scaling approach for microservices in cloud computing environments. Journal of Grid Computing, 21(4): 73. https://doi.org/10.1007/s10723-023-09713-7

[21] Quattrocchi, G., Incerto, E., Pinciroli, R., Trubiani, C., Baresi, L. (2024). Autoscaling solutions for cloud applications under dynamic workloads. IEEE Transactions on Services Computing, 17(3): 804-820. https://doi.org/10.1109/TSC.2024.3354062

[22] Dantas, J., Khazaei, H., Litoiu, M. (2021). Bias autoscaler: Leveraging burstable instances for cost-effective autoscaling on cloud systems. In Proceedings of the Seventh International Workshop on Serverless Computing (WoSC7) 2021, pp. 9-16. https://doi.org/10.1145/3493651.3493667

[23] Fan, D., Sun, H., Yao, J., Zhang, K., Yan, X., Sun, Z. (2021). Well production forecasting based on ARIMA-LSTM model considering manual operations. Energy, 220: 119708. https://doi.org/10.1016/j.energy.2020.119708