IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 5 May 2024, accepted 27 June 2024, date of publication 2 July 2024, date of current version 19 July 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3421956

RESEARCH ARTICLE

RLPRAF: Reinforcement Learning-Based Proactive Resource Allocation Framework for Resource Provisioning in Cloud Environment

REENA PANWAR AND M. SUPRIYA
Department of Computer Science and Engineering, Amrita School of Computing, Amrita Vishwa Vidyapeetham, Bengaluru 560035, India
Corresponding author: Reena Panwar (p_reena@blr.amrita.edu)

ABSTRACT Recent developments in cloud technology enable one to dynamically deploy heterogeneous resources as and when needed. This dynamic nature of the incoming workload causes fluctuations in the cloud environment, which is currently addressed using traditional reactive scaling techniques. Simple reactive approaches affect elastic system performance either by over-provisioning resources which significantly increases the cost, or by under-provisioning, which leads to starvation. Hence automated resource provisioning becomes an effective method to deal with such workload fluctuations. The aforementioned problems can also be resolved by using intelligent resource provisioning techniques by dynamically assigning required resources while adapting to the environment. In this paper, a reinforcement learning-based proactive resource allocation framework (RLPRAF) is proposed. This framework simultaneously learns the environment and distributes the resources. The proposed work presents a paradigm for the optimal allocation of resources by merging the notions of automatic computation, linear regression, and reinforcement learning. When tested with real-time workloads, the proposed RLPRAF method surpasses previous auto-scaling algorithms considering CPU usage, response time, and throughput. Finally, a set of tests demonstrate that the suggested strategy lowers overall expense by 30% and SLA violation by 77.7%. Furthermore, it converges at an optimum timing and demonstrates that it is feasible for a wide range of real-world service-based cloud applications.

INDEX TERMS Resource allocation, resource provisioning, autonomic computing systems, machine learning, reinforcement learning, virtual machines.

I. INTRODUCTION

Cloud computing is the edge of distributed computing and the platform of the future for hosting service-based and scientific-based cloud applications. Cloud applications use various on-demand services hosted by providers to carry out a range of functions. Major providers such as Microsoft Azure and Amazon EC2 use multi-layer auto-scaling to ensure effective resource allocation to address this and fit in with a subscription-based pricing model.

Cloud computing, with its pricing structure, offer tremendous processing capabilities and flexible resource allocation. To enable on-demand processing and storage for high-performance, highly available applications, many cloud data centres feature a large number of computing nodes. It becomes difficult to predict the exact resources required when service requests fluctuate quickly, which can result in problems like resource over- or under-provisioning. Over-provisioning of resources leads to resource wastage while under-provisioning with fewer resources leads to revenue loss and SLA violation. Therefore, focusing on dynamic resource provisioning is essential for effective resource prediction in order to overcome these obstacles.

The associate editor coordinating the review of this manuscript and approving it for publication was Ayaz Ahmad.

To ensure that Service Level Agreements (SLA) are met without incurring extra expenses or causing service delays, an efficient elasticity mechanism must precisely predict the resources needed based on the present workload. This indicates that to maximize resource predictions and guarantee SLA compliance, research gaps in dynamic resource provision must be addressed. In order to prevent resource over- or under-provisioning, it is essential to make well-informed decisions when compiling services. An effective and expandable infrastructure is crucial for using cloud computing service-based apps. Proactive resource allocation should be a part of this architecture, guaranteeing that resources are allotted ahead of time to avoid data loss. This proactive strategy enhances customer satisfaction, efficiency, and scalability, providing a dependable and smooth experience for cloud-based and service-based applications.

Thus, a dynamic resource provisioning system is needed to provide cloud customers with a collection of various tools available in the field of computing that can be utilized to effectively manage job processes and organize data storage [1]. One of the motivations of this paper is to study the problem of resource provisioning using dynamic algorithms in a cloud-computing environment. When a consumer requests resources in the cloud, Effective allocation is essential for cloud users to guarantee that their performance requirements are constantly satisfied [2]. There is an imbalance in the back-end burden caused by traditional server architectures' inability to adaptively assign computer resources for changing data requests. Processing problems and the possibility of data loss result from this imbalance. Therefore, a robust, scalable processing and storage infrastructure, that ensures a proactive and autonomous virtual resource allocation mechanism is needed.

Resource provisioning, which is a crucial aspect of resource management, is essential in determining the necessary infrastructure resources (capacity) needed to support a set of applications. This plays a vital role in ensuring that there are sufficient resources available to run the applications [3]. Two crucial processes make up the cloud provisioning process: First, VM Provisioning, which entails launching one or more virtual machines (VMs) on virtual servers in either a private or public cloud computing platform, several mapping requirements, such as memory and storage to the primary cloud, are considered when selecting a physical server for web hosting virtual machines in a cloud. The second is application service provisioning, which involves scheduling and mapping incoming requests to the services hosted in virtual machines (VMs) within a VM cluster.

In this proposed work, we mainly concentrate on the second one, given a set of virtual machines in a decentralized and load-balanced manner; incoming requests are distributed across various services [4]. With differing dynamic requirements of cloud services from the user's side, improving the availability of services to cloud users and mapping incoming requests to the services hosted in virtual machines within a cluster of VMs is a critical task for cloud providers. Hence, resource provisioning becomes a continual issue faced while working with cloud-based technologies, which impedes the use of cloud-based services offered by the cloud provider [5]. The goal of resource allocation strategies is cost reduction; dynamic allocation techniques enhance resource utilization while lowering virtual machine usage [3], [6].

The motivation of current research is to maximize the profitability and efficiency of cloud services, which raises concerns regarding cost, provisioning, resource selection, and computational efficiency. Therefore, efficient resource provisioning algorithms that satisfy performance requirements and benefit providers and customers alike are needed to answer these problems. Regression or prediction models based on time series analysis can be used to cloud resource utilization data to forecast future requirements. The selection process for models, including possible combinations, is still governed by accuracy evaluation in research.

In this work, we present a hybrid resource provisioning strategy for cloud applications that combines the ideas of reinforcement learning (Q-RL) with autonomous computing. The suggested work aims at minimizing both the under-provisioning and over-provisioning of resources for cloud users and, in turn, reducing the total cost of provisioning the resources for a given period. Our method provides the best possible resource provisioning framework, reducing SLA violations and enhancing response times by anticipating and dynamically adjusting computing resources according to the kind of application. This work focuses on the dynamic resource allocation strategy and, in particular, a reinforcement cloud resource provisioning method. This proposed work stems from issues with uncertainty regarding both demand and pricing that can be a factor in cloud computing environments. The model considers virtual machine load, request arrival rates, and rejection percentages for dynamic resource provisioning and follows IBM's suggested MAPE-K loop, which effectively scales resources to optimize performance while avoiding over- or under-provisioning. This approach, which is rather unique in the literature, combines proactive and reactive elements [7].

Here, emphasis is put on Actor-Critic techniques based on off-policy reinforcement learning that use the Experience Replay (ER) method to minimize the bias and variation and maximize the sample efficiency. Likewise, the evaluation results are used to calculate the surplus shared resources of the providers and the suggested approach will be used to offer its ideal value to cloud service providers [2]. Demand uncertainty emerges from the cloud consumer side for a required resource to be provisioned and, in turn, to make an optimal decision during runtime. In addition, cloud provider price uncertainty is considered to modify the balance between on-demand and oversubscribed prices. Numerous studies and simulations have been conducted, and the findings imply that the proposed work can lower total costs even in uncertainty. This paper's mathematical formulation, which can be combined as presented in the subsequent sections,

is where it makes its substantial contribution. The research contributions of the proposed work are:

- A framework (RLPRAF) for proactive resource provisioning using the MAPE-K loop has been proposed, which integrates reinforcement learning with autonomic computing and is based on fine-grained elastic resource supply under changing workloads. It offers resource provisioning that is both reactive and proactive for different cloud applications.
- Proposed a forecasting system for the dynamic workload changes that aid in resource scheduling by forecasting resource needs and scaling out delays.
- Proposed resource provisioning mechanisms for the cloud application services of the IaaS provider for each MAPE-K loop phase. It guarantees both autonomic and proactive resource provisioning for cloud applications.
- Analyzed the proposed model for different VM load conditions, and the results of response time (average) and a number of violations of the service level agreement have been presented to facilitate informed decision-making. The proposed model uses reinforcement learning, also known as Q-learning, which generates decisions by replicating the decision-making process and outlining the possibilities that have the most significant cumulative rewards in resource provisioning.
- Analyzed the performance of the proposed system using real-time web server traces from ClarkNet and Google.

This work also presents the results from the experiment on different traces taken to show that the reinforcement learning technique, compared to the existing methods, help resources to have better response times, fewer SLA violations, fewer hours of virtual machine utilization, and lower costs. The suggested framework is classified as a service-based cloud application among resource provisioning techniques because it allows the server to efficiently use all the resources given, even under changing workload conditions.

The rest of the document is structured as follows: A discussion of related research, as well as an overview of the methods and techniques employed for provisioning, is presented in Sections II and III, while Sections IV and V describe the proposed solution and present the simulation and experimental results obtained for a range of real-time traces. The paper is concluded in Section VI.

II. RELATED WORK

This section describes the summary of the existing research on resource provisioning and presents in two categorizes - one based on autonomic computing and other on reinforcement learning.

A. RESOURCE PROVISIONING BASED ON AUTONOMIC COMPUTING

According to IBM, autonomic computing systems are autonomous and capable of producing optimized system performance with limited work for system administrators. The MAPE-K (IBM) paradigm has been suggested by Huebscher et al. to manage autonomic computing with a variety of self-management, self-configuration, self-optimization, self-healing, and self-protecting properties [5], [8]. The performance of the resource management system, which is based on services, can be enhanced by predicting future resource needs in a way that satisfies the QoS standards specified in the SLA [9].

Using a service-level agreement, the dynamic resource provisioning and monitoring system (DRPM), as suggested in [7] and [10], maintains the resources and meets the QoS requirements of the customers. A hybrid resource provisioning approach is proposed by Etemadi et al. [11], mainly for SaaS applications, that lowers costs, time factors, and SLA breaches. To reduce latency and virtualization overhead, Wang et al. [12] propose an autonomous virtual application provisioning system for big data centres. In [13], an auto-scaling method for cloud workload prediction combines the auto-regressive moving average (ARMA) and linear regression models to anticipate resource workloads at a reasonable cost. A framework for e-healthcare apps was created by Bhardwaj and Sharma [14], which improved resource usage, decreased response times, and minimized rejected requests. For parallel scientific programmes, an extension in [15] offered an elastic controller with fuzzy logic that resulted in better resource usage and quicker completion times. Zhong et al. [16] provides a thorough comparison of these models that covers measurements, approaches, and policy types. Shakarami et al. [17] presented a survey for data replication scheme among different existing cloud computing solutions in the form of a classification approach.

Our study uses real-world workload logs from ClarkNet and Google Traces from the literature to find the models that could estimate cloud resource usage. The effectiveness of machine learning algorithms has also been evaluated.

B. RESOURCE PROVISIONING BASED ON REINFORCEMENT LEARNING

This section presents the survey on reinforcement learning-based resource provisioning techniques. As per Siar et al., [18], an agent uses Q-learning that combines Reinforcement learning and fuzzy intelligence techniques for efficiency maximization in scheduling tasks. A new cooperative Q-learning approach is presented in this work to encourage collaboration between multiple agents to improve efficiency in a specified environment. Moazeni et al. [19] proposed a dynamic resource allocation strategy using an adaptive multi-objective teaching-learning based optimization (AMO-TLBO) algorithm that helps to minimize cost and maximize utilization using well-balanced load across virtual machine. Fan et al. [20] proposed an efficient and highly secure blockchain assisted authentication scheme using a combined off-chain and on-chain approach.

Q-learning has also been used by Xu et al. [21] to design systems for vertical scaling. To manage the directed acyclic graph (DAG) structures' graph-based job

scheduling issue and shorten the overall DAG execution time, Orhean et al. [10] employed state-action-reward-state-action (SARSA), which is a model-free reinforcement learning algorithm that shares similarities with Q-learning. Our proposal in this paper focuses on implementing the MAPE-K loop using Q-learning for optimal decision-making.

Literature also finds the use of deep reinforcement learning (DRL) for resource provisioning in cloud applications [5]. In the DRL framework developed by Bao et al. [22] for batch processing, an artificial neural network (ANN) model has been trained using the actor-critic RL technique.

Ghobaei-Arani et al. [23], utilizing the MAPE-K loop, provide a hybrid resource provisioning approach with a focus on the planner phase for forecasting future requests. The results demonstrate that the suggested technique lowers cost, time elements, and SLA violations. This approach, however, is limited by the assumption that the requests are for SaaS cloud apps [24], while our proposed model focuses on the interaction between SaaS, PaaS, and IaaS layers. Bhardwaj and Sharma [25] develop an elastic resource provisioning framework for autonomic computing phases in e-health applications. The monitor and analyzer stages focus on using a queuing load prediction model. Results demonstrate improved resource use with quick response and completion times. When forecasting the resources used in the future, fuzzy logic aids decision-making, thus resulting in optimal resource use and rapid completion [10], [24].

The incorporation of Markov Decision Process (MDP), Q-learning, and edge-cloud technologies in the framework of microservice coordination is further extended by Wang et al. [26] where the sequential decision system that underlies the process of microservice coordination is assumed and formulated as an MDP design. To analyze the function invocation patterns and decide on the best function container scaling in advance, Agarwal et al. [27] provide a Q-learning agent that models system states using metrics such as available function containers, CPU utilization per container, and success/failure rates [23]. The work concludes that the model-free RL algorithms do not require prior knowledge of the input function because of their nature. However, these solutions only consider specific workloads, and hence performance cannot be assured under varying workloads, while our work focuses on prior knowledge for effective decision-making of virtual machine utilization using Q-learning in the planner phase of the MAPE-K loop.

The actor-critic approach is used by Qiu et al. [28] to scale the essential microservices identified by Support Vector Machine (SVM). This horizontal scaling approach alleviates SLA violations by analyzing three essential aspects: the SLO maintenance ratio, workload variations, and request composition. Ghobaei-Arani et al. [23] chose the Q-learning method where the learning agent repeatedly observes the current state of a controlled system (workload, VM count, and performance SLA), performs a task, and then changes to a new state. To avoid a prolonged period of exploitation and exploration, the authors introduce a convergence speedup phase to accelerate learning at regular intervals. Dezhabad and Sharifian [29] propose the GARLAS approach, which combines genetic algorithm, reinforcement learning, and queuing theory to determine the ideal number of active firewalls based on incoming traffic intensity at any given time. The integration of RL and genetic algorithms enables the system to learn and adapt to changes in workload and network conditions, making it a robust and scalable solution for firewall management in dynamic and unpredictable environments; however, the system focuses on parameter response time alone, while our model focuses on different parameters like average response time and average load at IaaS level. Horovitz and Arian [30] present a Q-learning technique for horizontal scaling that includes initialization steps, smoothing, and action monotonicity. For continuous actions in specified states, the method uses an action space methodology.

In their study, Wei et al. [31] propose a resource allocation method for SaaS providers operating in a dynamic and stochastic cloud environment. The method is based on the Q-learning adjustment algorithm (QAA) and aims to assist providers in making optimal resource allocation decisions [32]. This work aims to cut down rental costs as much as possible while offering enough processing capacity to meet client requests [31].

The model-free method-based works ([29]; [30]) not only need the space to reserve the action R(s, a), it also needs the effort and information to update it. When multiple characteristics or dimensions are used, it can lead to a combinatorial explosion of states, making them challenging to manage effectively. Furthermore, the number of states can increase exponentially, depending on the potential values of the specified variables, especially when scaling the system. This perspective reveals that one of the primary limitations of RL is the dimensionality problem, also known as the state space dimension problem. Hence several solutions have been looked into to mitigate its effects. The combination of RL and function approximation, a supervised learning generalization is one among them. One example is the non-linear approximation of R(s, a), as demonstrated by deep neural network theories (Cheng et al. [40], Tong et al. [41]). However, the field of cloud auto-scaling has not yet seen widespread adoption of RL with function approximation [42].

Table 1 summarizes the above-mentioned models based on the objective parameters, the optimization goal, the benefits, and the shortcomings of the methods under investigation. Based on their research objectives, the researcher can choose the best strategy.

III. BACKGROUND

This section provides an introduction and contextual information necessary for implementing the proposed system.

A. AUTONOMIC COMPUTING ARCHITECTURE

The autonomous processing system is the IBM-introduced element for autonomous computing. This system is controlled

TABLE 1. Autonomic provisioning - a comparative survey.

Reference	Objective Parameters	Evaluation Tool	Advantages	Limitations	Dataset Used
Ghobaei et al. [23]	CPU utilization,Response time	CloudSim	Decreased resource cost,Improves resource utilization	Limited to SaaS only	Clarknet and NASA
Tushar et al. [25]	Response Time, Finishing time, AverageVM load, Rejection percentage [7]	CloudSim	Effective Resource utilization, Proactive approach [7]	Finite number of resources for scaling	Clarknet
Xu et al. [21]	CPU utilization, Resource usage Cost and SLA constraint	Apache Hadoop (YARN)	Reduction in resource cost through optimization, SLA confirmation	Limited workload scenarios	Testbed
Zhang et al. [33]	CPU Utilization, Response time	Kubernetes	SLA violation reduction	Inaccuracy due to cold starts	NASA and FIFA
Orhean et al. [10]	Minimize the time required to execute tasks while ensuring SLAs	Work flowSim	Improves response time	Limited scalability	TestBed
Agarwal et al. [27]	Request arrival rate,SLA	Kubernetes	Reduced cold-start overhead	Efficiency of the training model	Azure traces (HTTP)
Rossi et al. [34]	CPU utilization, Response Time	Docker Swarm	Reduction of resource consumption, Improved training speed, Penalty rate	Simplicity of application models	Amazon EC2
Wang et al. [26]	To decrease service delays and lower migration costs.	MATLAB	Optimized performance	Lack of consideration on load balancing	Shanghai Telecom's dataset and Shanghai Taxi Track
Dezhabad et al. [29]	CPU Utilization,Response time,SLA	MATLAB	Reduces response time, Better resource utilization	Dimensionality problem	Calgary, ClarkNet, NASA and Saskatchewan
Arian et al. [30]	CPU utilization,SLA	Kubernetes	Improved response time	Dimensionality problem,Threshold-based approach	NodeCellar webworkload
Hu et al. [35]	Execution time, cost, Rental-Time	Simulation Engine	Effective resource utilization, Reduction in Cost, Based on deadline analysis	Not an autonomic predictive based approach	Alibaba Cloud
Ghobaei-Arani et al. [36]	Response time, Number of VMs, Cost	Cloudsim	Stronger prediction model, Reduces the cost for rental resources, Better response Time	Limited amount of resources	Synthetic, RuneScape workload
Etemadi et al. [37]	Cost,CPU utilization,network usage,Delay violation	iFogSim	Reduction in cost,delay violation,network usage , Increases CPU utilization	Resource placmement problem	Synthetic ,Real workload
Nazeri et al. [38]	Energy consumption,SLA ,Response Time	CloudSim	Decreses energy consumption, response time	availably and reliability did not considered	NASA and Clarknet
Fan et al. [20]	Energy consumption,Delay	Matlab	Task allocation effectively , extensive simulation was performed	Decreses task offloading	Synthetic workload
Veira et al. [39]	,Cost,SLA,Execution time	Cloudsigma	trust and processing considered	Increased scheduling cost	Synthetic workload
Proposed Work	**AvgVM Load, Response time, Rejection percentage, No of VMS**	**CloudSim**	Reduced cost for VMs, Improved Response time	**Resources are limited for a small interval of time**	**Clarknet, Google traces**

by the human body's autonomic nervous system, which knows the essential interfaces and functional components for the control MAPE-K loop's analysis phase and its management component. It is critical to provide constant feedback on the system's events to persuade users to operate service-based cloud applications [7]. The autonomous processing system is presented in Figure 1 with managed elements, cloud applications, resources, and an autonomous manager with a control MAPE-K loop. Interfaces, sensors, or actuators make up the managed element [43].

The autonomic engine stores the sensor's data from the environment (also known as managed elements). Based on the autonomic manager's information, the actuator scales the resources up or down. The Monitor, Analyzer, Planner, and Executor with the Knowledge engine, or MAPE-K loop, is a critical component of autonomic processing. Here, the managed element and MAPE-K loop stand in for the various software and hardware resources, such as cloud services, operating systems, CPUs, cloud applications, storage, virtual machines, etc. During the monitoring phase, the information acquired by the sensors about response time, CPU consumption, and memory is checked and saved in the knowledge base for later processing [11]. The analysis phase analyses the data gathered and makes resource predictions for future use. During the planning phase, the resources are scaled up to determine the most effective allocation, which is then implemented during the MAPE-K loop's execution phase.

B. PROACTIVE PROVISIONING OF VIRTUAL RESOURCES

This section describes the proactive resource allocation mechanism. These mechanisms are divided into two categories: elastic infrastructure and elastic support, which are dealt in this work The task type formed from data nodes is the input parameter. The virtual resource needs for task T are determined by the task type. For instance, the cloud would allot more virtual CPU resources to the work if it is classified as CPU-intensive and deadline-critical. However, if the task is classified as I/O-intensive and mission-critical, the server's

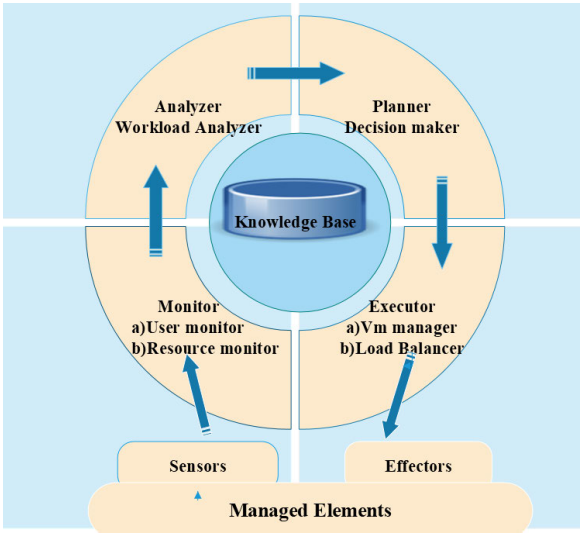

FIGURE 1. MAPE-K loop - architecture.

job is to just allocate task T to virtual I/O devices to fulfil its requirements.

C. USE OF REINFORCEMENT LEARNING

Reinforcement learning is a branch of artificial intelligence in which an agent keeps track of the condition of its surroundings and takes suitable optimal action through a trial and error method to determine the best solution to maximize the reward in a critical situation under dynamic circumstances. The agent finds the ideal system state at any given time based on historical data. Agents receive rewards based on their actions. The agent always wants to locate the biggest possible prize [44].

1) MARKOV DECISION PROCESS

A discrete-time stochastic process called a finite Markov Decision Process termed MDP, which has a finite set of states (s_t), a finite number of rewards (r_t), and a finite number of actions (a_t), can be used to formalize an RL problem. Only the finite MDP is taken into account in RL. Finding an optimal policy that can maximize the long-term expected reward is the main objective of the MDP. There are numerous methods for solving MDP, including policy iteration, Q-learning, value iteration, linear programming, etc. The issue that an agent is trying to resolve is the series of states (s_1, s_2, s_3,..., s_n). The agent acts and transfers it from one state to another. Each action must be performed at a specific time interval to maximize the reward.

2) MARKOV PROCESS

The Markov, a random probability distribution process, specifies the possible states in the process. Markov properties are defined as a present state's reliance on a past state S_t (overutilized, underutilized, or no operation)

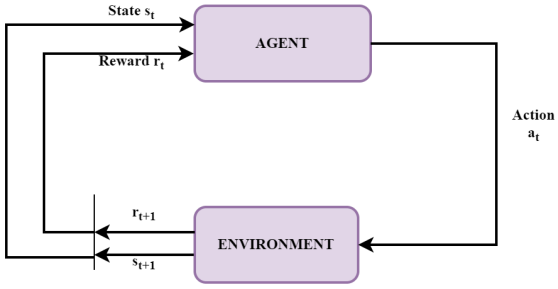

FIGURE 2. Interaction of a reinforcement learning agent with its surroundings.

as given in Eq. 1.

$$P[s_{t+1}|s_t] = P[s_{t+1}|s_1, s_2, \ldots.., s_n]. \tag{1}$$

The proposed planner phase of the autonomic MAPE-K loop, implements the RL-based agent, which takes action dynamically for the resources to be provisioned depending on the current status of the system's performance. Q-learning, a popular Bellman equation-based model-free reinforcement learning algorithm, has been used to scale microservices. The action-value function is implemented using Q-learning, which evaluates the reward of acting as a specific condition. Q-learning has the added advantage of being able to give the anticipated reward without using the environment's model.

The primary objective of Q-learning is to develop a policy that can instruct an agent on the behaviors to be performed to maximize reward-specific conditions. It is an off-policy RL that considers the optimal course of action in the given situation [23]. As a result, the algorithm's (Q-learning) state-action combination is as follows: $Q : S * A \rightarrow R$, where Q is initialized to this value dynamically.

Each time a new state is attained by an agent (s_{t+1}), which may be influenced by both the previous state s_t and the chosen action a_t, the Q-value table is updated sequentially. In Q-learning, the agent's objective is to maximize Q value. The agent chooses an action at A for a system at time t with state s_t as shown in Figure 2. The state changes from s_t to s_{t+1} upon the application of action a_t to the environment, and the environment returns an immediate reward r_{t+1} [23]. Bellman's equation can be used to update the Q-function each time, which results in the application of an active environment as determined by Eq. 2.

$$\underbrace{Q^{nw}(s_t, a_t)}_{\text{New Q-Value}} = Q(s_t, a_t) + \alpha [\underbrace{r_t}_{\text{Reward}} + \overbrace{\gamma \max Q'(s_{t+1}, a)}^{\substack{\text{Maximum predicted reward, given} \\ \text{new state and all possible actions}}} - Q(s_t, a_t)] \tag{2}$$

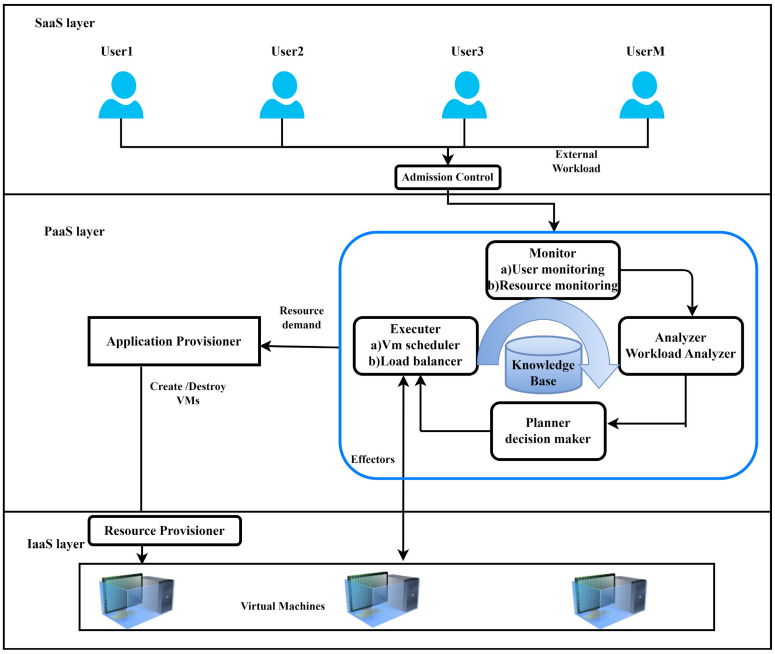

FIGURE 3. A framework for autonomous resource provisioning built on Mape-K control loop.

where α is the learning rate ($0 < \alpha \leq 1$) and r_t is the reward obtained while migrating from the state s_t to the state s_{t+1}. Here $Q^{nw}(s_t, a_t)$ is the consequence of three factors: first, (1)-α)$Q(s_t, a_t)$ is the current value, second, $\alpha\, r_t$ is the reward $r_t = r(s_t, a_t)$ to obtain if action a_t is taken when in state s_t (weighted by learning rate), third, the highest reward that may be earned from the state s_{t+1} is $\alpha\gamma \max_a Q(s_{t+1}, a)$, weighted by learning rate and discount factor. Here γ is the discount factor, which represents the influence of potential future benefits on our choice of action at this timestep and how our decision-making will change if we adopt a particular course of action. Section IV discusses the Bellman equation's application in detail.

IV. PROPOSED WORK

The proposed reinforcement learning approach (RLPRAF) described in this section, utilizes the MAPE-K loop and runs periodically for each cloud service offered by SaaS providers. This section includes an extensive analysis of the proposed MAPE-K algorithms for resource provisioning systems and discusses the problem formulation in detail.

A. RESOURCE PROVISIONING FRAMEWORK

Figure 3 depicts the proposed resource provisioning framework with the three main cloud layers: SaaS, PaaS, and IaaS. This is how the model operates: To use cloud services, a client must first submit requests to the admission control (i.e., the SaaS provider, which operates to increase profit) which subsequently forwards the request to the PaaS layer and store it in database. Second, the MAPE-K phase, which combines the reinforcement learning approach with linear regression modelling, acts as the brain for the provisioner's autonomous and effective decision-making and receives the request in the PaaS layer. Lastly, IaaS offers a huge quantity of VMs, enabling the expansion of numerous cloud services on a single VM. Through the MAPE-K loop, the PaaS layer maintains the resources for the cloud services offered by the SaaS layer.

The MAPE-K loop of the proposed solution is described in the subsections that follow.

1) MONITOR

Resource and user monitoring are the two parts that make up the monitoring phase. The task of resource monitoring entails gathering data on CPU utilization, typical VM load, and memory usage, whereas the task of user monitoring entails gathering data on workload requests made by each user, such as request arrival percentage, type, length, and dropped requests. These gathered data are kept in the repository for later use [7].

2) ANALYZER

The collected data during the monitor phase is processed during the analyzer phase. To guarantee the requested QoS level, the data gathered in the preceding phase is examined to see if any action is necessary.

3) PLANNER

To achieve an acceptable SLA with reduced cost, the planner phase decides on the number of distributed VMs needed among the total available services. The decision-maker for

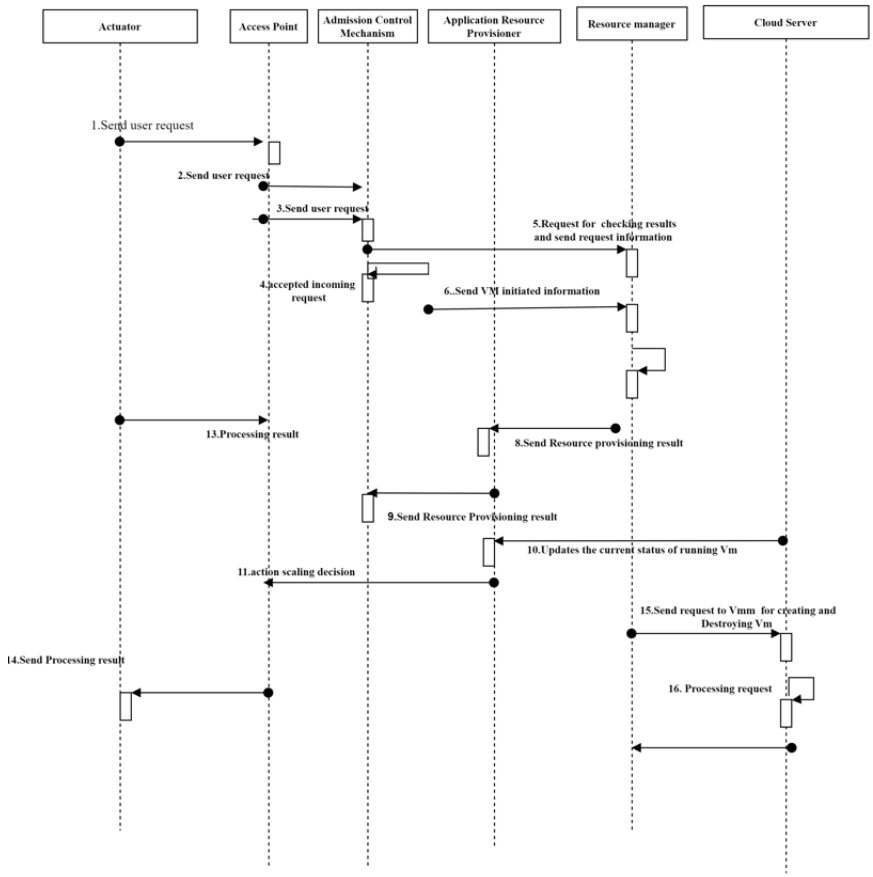

FIGURE 4. Sequence diagram for resource provisioning framework.

TABLE 2. Description of the notations.

User_Z	zth User
$\text{User}(\Delta t)$	Overall user count at interval (Δt)
Y	Number of requests made by all users combined [23]
Y_z	The total number of queries (request) received to zth users
D_z^r	rth request of zth users
Dl_z^r	deadline for rth requests for zth users
FT_z^r	Finish time of rth request of zth of users [7]
TmDelay_z^r	Amount of time required to complete the processing of a particular request in a system (Delay) for rth request of zth number of users
S	Total number of SaaS providers' cloud services [7]
S_i	ith cloud service provided by SaaS provider [25] [45]
$\text{VM}_i(\Delta t)$	Number of virtual machines supplied to cloud service S_i at tth time interval [44] [45] [25]
$LD_i(\Delta t)$	Total length of tasks/jobs on the VMi service queue at the specified time interval t proportional to the VM_i service rate [45].
$\text{LD}_{total}(\Delta t)$	Total load of all virtual machines at the selected time interval (Δt)
$LD_{average}$	Ratio of the number of active VMs to the overall load
WT	Waiting time for $User_m$'s task(s) to be dequeued from a queue (buffer) [25]
$\lambda_i(\Delta t)$	number of cloud service requests during the periodΔt
VM	Initializations of the total number of VMs
MakeSpan^r	Time taken to complete each user request Data^r
$\text{Rej}_i(\Delta t)$	The percentage of user requests that were declined Data^r
ResT^r	Time taken for a user to respond to a request Data^r
$\text{ResT}_{average}(\Delta t)$	Total response time to all users per interval as a ratio [25]
$\text{LD}_{Predicted}$	Predicted workload

the proposed approach uses reinforcement learning to allot the necessary VMs. This decision is made possible due to the presence of the analyzer phase.

4) EXECUTOR

A load balancer and VM manager are the components of the executor phase. The load balancer receives requests from the

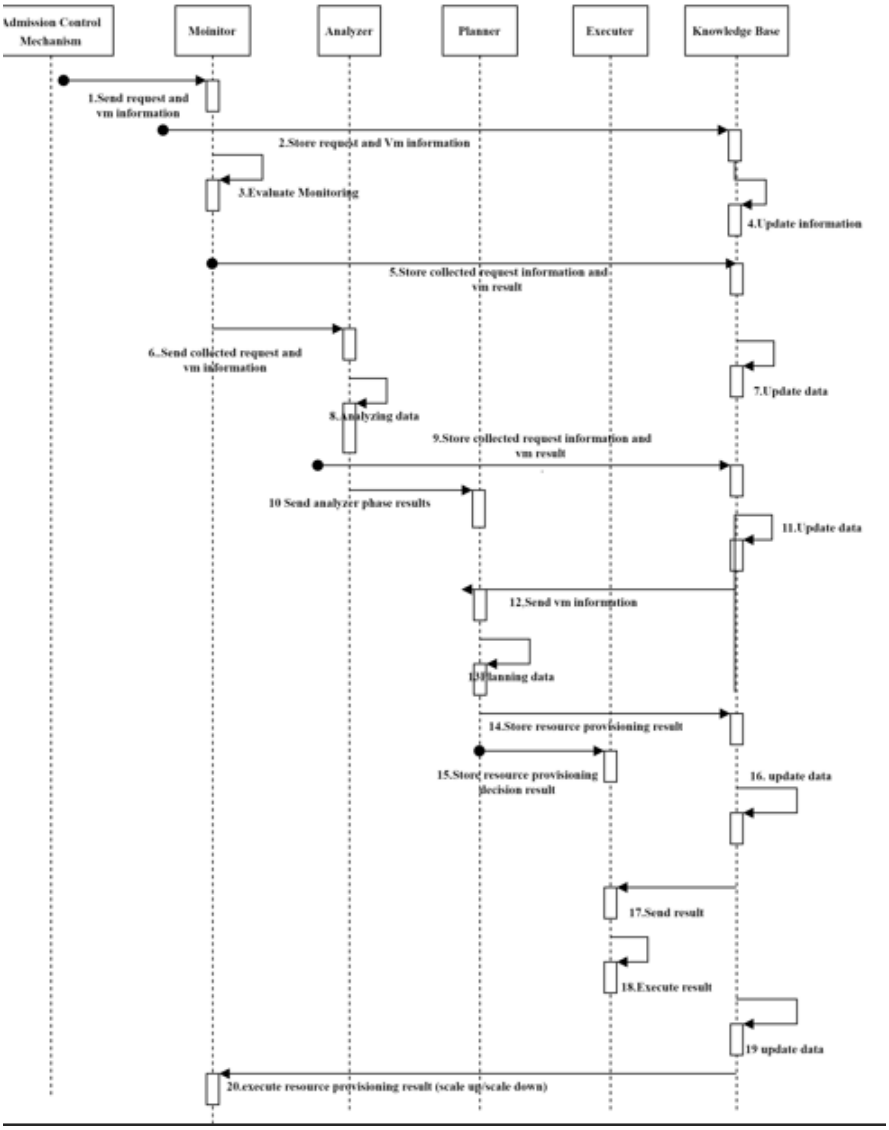

FIGURE 5. Sequence diagram for resource provisioning agent.

user to the accessible VMs in accordance with the shortest job first load-balancing policy. The VM Manager implements the choices made during the planning process (such as scaling operation, scale-out operation, or no-operation).

The sequence diagram explaining the workflow of the provisioning framework is presented in Figures 4 and 5.

B. PROBLEM FORMULATION

The sudden rise in user emergency service demands result in sharp and erratic surges of resource requests, which have a significant impact on resource allocation strategies for the applications. As an illustration of a use case, consider virtual machines that host online applications like e-commerce platforms, which are frequently N-tier and feature web servers that carry out business logic [45]. The number of incoming requests and the resources available to the VM determines the end-user response time. The rate at which requests arrive at the web server will typically vary, and it may experience a rise during peak hours. This will further increase the server's use of resources and put more strain on the physical node, which could impair the web server's response time.

Hence, necessary and quick decisions must be made in a cloud environment so that the response time and the SLA violations are minimal. The usage of Reinforcement learning allows for viable decision-making on the type and amount of VMs to be used (scaling), and resources ought to be allocated at any given moment (scheduling). We intend to create a method that reduces SLA violations by recognizing overloaded virtual machines and implementing a migration strategy to choose new target nodes for the VMs. Hence the primary goal of this paper is to provide resources for applications related to cloud-based services.

The proposed framework for service-based cloud applications in line with the notations presented in Table 2 is described in detail. This section also explains the performance measures needed to test the proposed methodology. Let Z denote the number of users ($User = Usr_1, Usr_2, Usr_3, \ldots, Usr_Z$), let D_z^r be the zth user's rth request and let each user Z have R_Z requests ($Req_Z = Req_Z^1, Req_Z^2, Req_Z^3, \ldots, Req_Z^{R_Z}$) produced for each type of service delivered by the SaaS provider. The maximum time a user must wait for the requested service (the SLA's response time requirement for User t) to respond is denoted by the variable Dl_z^r. Let S represent the total number of cloud services the SaaS provider offer, and let S_i stand for the cloud service with ID_i, ($S = S_1, S_2, S_3, \ldots, S_S$). Additionally, define $VM_i(t)$ as total virtual machines utilized by cloud services S_i at t and $Load_i(t)$ representing the load (number of requests) for each S_i at interval t. Assume V as the set of VMs denoted as $VM_1, VM_2, VM_3, \ldots, VM_x$ that IaaS provides to SaaS at specific intervals to execute different cloud services. Here, the load of a virtual machine can be determined as shown in Eq. 3.

$$LD[VM_i, \Delta(t)] = \frac{\sum_{i=1}^{CurrentCS}(CPEs_i * CL_i)}{PEs * MIPS} \tag{3}$$

where CL denotes the number of tasks running as a user request, CPEs mean the number of processors the program uses, and $CurrentCS$ indicates the total demand from the concurrently running VMs on the client side. PEs stand for the number of VM processing elements, and the computational capacity of every processing unit is denoted by $MIPS$ (Million Instructions Per Second) [7]. The overall load on the supply of VMs running at that specific period, as shown in Eq. 4, determines the average load on virtual machines in the cloud at a given time interval t [9].

$$LD_{average} = \frac{\sum_{i=1}^{ActiveVM} LD(VM_i)}{Activevirtualmachines(VM_i)} \tag{4}$$

where $ActiveVM$ is the total number of running virtual machines at a time interval Δt, and $LD(VM)$ denotes a full load of virtual machines at Δt. Response time is the computed period between the moment at which a request is made and the time at which the cloud client or user responds initially. Eq. 5 calculates the response time of a specific request generated by the user.

$$ResT = FNT - PAT - ART \tag{5}$$

where FNT denotes the user request's first response, PAT is the time the VM ran the request, and ART is the user request's arrival time in the cloud. The capacity of the overall system is determined using Eq. 6.

$$Cap_i = PE_{numi} \times PE_{mipsi} + VM_{bwi} \tag{6}$$

where PE_{numi} denote the total number of processing elements, PE_{mipsi} indicate the total number of processors in VM_i processing one million instructions per second, and VM_{bwi} denote the VM_i communication bandwidth. The average response time is calculated by dividing the overall response time ($ResT_z^r$) by the number of users or clients for each interval ($User(\Delta t)$), as presented in Eq. 7.

$$ResT_{average} = \frac{\sum_{i=0}^{User(\Delta t)} ResT_z^r}{User(\Delta t)} \tag{7}$$

Virtual machine initialization generally includes a delay factor known as Bootupdelay. Therefore, the delay is the time that elapses between the task's deadline and the actual response time for user requests Req_z^r, as shown by Eq. 8.

$$TmDelay_z^r = \begin{cases} FT_z^r - DL_z^r & \text{if } FT_z^r > DL_z^r \\ 0 & \text{if } Otherwise \end{cases} \tag{8}$$

Here, $TimeDelay_z^r$ denotes the time delay in the transaction. When the SLA's established terms and conditions, such as the service level objectives, aren't met by a SaaS provider, an SLA violation occurs (i.e. $SLAV(Req_z^r)$) If the delay duration exceeds zero, an SLA violation occurs; otherwise, no action is taken, as described in Eq. 9.

$$SLAV(Req_z^r) = \begin{cases} YES & \text{if } TmDelay_z^r > 0 \\ NO & \text{Otherwise} \end{cases} \tag{9}$$

The total Cost of virtual machines is calculated by the formula as follows in ($/hour$):

$$Cost_z^r = Cost_{memory} + Cost_{storage} + Cost_{VMhour} \tag{10}$$

Here $Cost_z^r$ denotes the Cost of virtual machines for rth request for zth user, $Cost_{memory}$ denotes the memory cost,$Cost_{storage}$ denotes the storage cost,and $Cost_{vmhour}$ denotes the virtual machine hour cost per time interval.

C. TIME COMPLEXITY OF THE PROPOSED ALGORITHM

Time complexity of an algorithm indicate the total steps required to solve a particular problem which is proportional to the length of the input. The arrival of user request play a major role in our proposed algorithm. The notations used in the time complexity analysis is listed in Table 3. It describes all the operations performed for each incoming request in a time interval Δt as given in Eq. 11.

$$T(n) = \Delta t.T_{req}.[T_{adc} + 3T_{sendr} + T_{mape} + 2T_{process} + T_{propagate}] \tag{11}$$

where T_{mape} is the time required to run the $mape()$ algorithm and is expressed by Eq. 12.

$$T_{mape} = T_m r + T_a s + T_p g + T_e n \tag{12}$$

The required time for execution of analysis and planner phases is represented by Eq. 13 and Eq. 14.

$$T_a = T_{la} \tag{13}$$

$$T_p = T_{p1} + T_{p2} + T_{p3} \tag{14}$$

TABLE 3. Notations for time complexity.

Notifications	Description	No. of iterations
T_{ur}	Time required for completion of user requests	-
T_{adc}	Time required for admission control	1
Δt	Time interval	-
$T_{process}$	Time required to process a request	1
T_{mape}	Time required for MAPE function	1
T_{sendr}	Required time to send a request	3
T_{mr}	Time for Monitor phase	-
T_{as}	Time for Analysis phase	-
T_{pg}	Time for Planning phase	-
T_{en}	Time for Execution phase	-
T_{la}	Time required for prediction	-
T_{bl}	Time required for Bayesian learning	-
$T_{propagate}$	Time required for propagation	-

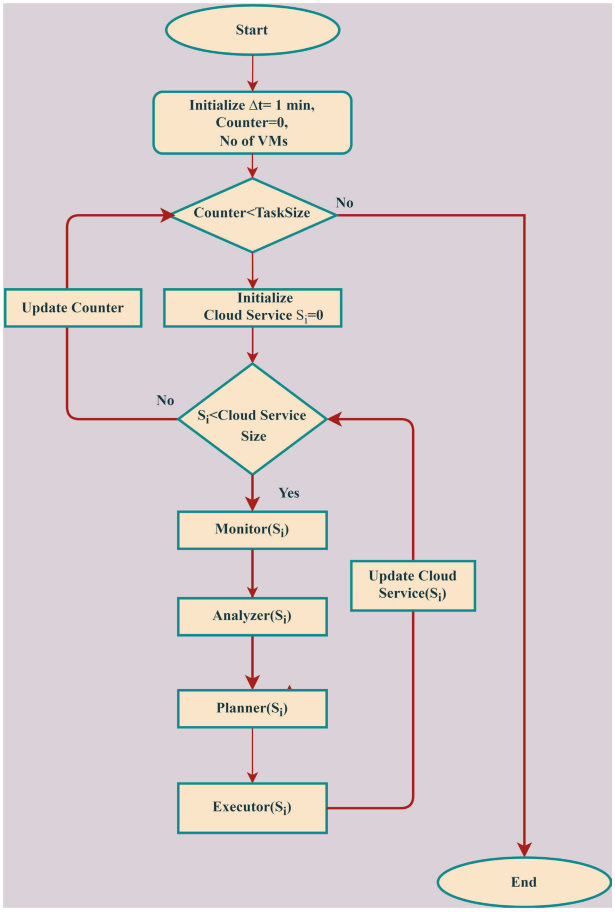

FIGURE 6. Flowchart for autonomic resource provisioning.

By combining equations Eqs.(12 to 14) in Eq.11, we get:

$$T(n) = \Delta t.T_{req}.[T_{adc} + 3T_{send} + T_{mr} + T_{as} + T_{pg} + T_{e}n + 2T_{process}] \quad (15)$$

Time required for serving user workload requests at a time interval Δt can be expressed as follows:

$$T_{req} = n \quad (16)$$

Here, all the values considered for analysis denote the maximum consuming time and is given as:

$$T(n) = \Delta t.n.[10.T_{max}] \quad (17)$$

Taking Δt as a constant, and T_{max} as the maximum required time for an operation, the time complexity of proposed algorithm with a linear running time is given as follows:

$$T(n) = (n) \quad (18)$$

D. PROPOSED ALGORITHM FOR RESOURCE PROVISIONING

This section presents autonomic resource provisioning strategies for different cloud services, with flowcharts presented in Figures 6 through 12. These algorithms are executed within each MAPE-K loop at regular intervals. The proposed MAPE-K algorithms continue to execute until all open requests in the cloud system are completed by time t. Real-time SaaS companies maintain a pre-defined list of cloud service providers to fulfill the constantly evolving needs of their customers [7], [24].

The suggested algorithm is responsible for the administration of all virtual machines allotted to the cloud service S_i. In the monitoring phase, the algorithm initializes the required VMs from an accessible resource pool, which may cause a minor pause, known as boot-up latency. To improve VM allocation, the MAPE-K loop proposed in this research employs the reinforcement learning approach and a linear regression model during the analysis phase. The algorithm optimizes VM allocation by considering multiple limitations, such as load, response time, makespan, and job rejection rate on the VM. This approach aims to improve the performance and efficiency of the cloud service by allocating resources more effectively and reducing latency in the system. The proposed technique is expected to enhance the quality of service and user experience for cloud-based applications [7], [24].

TABLE 4. Decision table for the proposed MDP - *Q*(*State*/*Action*).

States	Scaling			No-Operation			Scaleout		
	$Q(S/A)$	$R(S/A)$	$Q^{updated}(S/A)$	$Q(S/A)$	$R(S/A)$	$Q^{updated}(S/A)$	$Q(S/A)$	$R(S/A)$	$Q^{updated}(S/A)$
Over – Utilization1001[23$\rightarrow$]	0	0.77	0.811	0	0.22	0.188	0	0.11	0.001
Normal – Utilization1001[23$\rightarrow$]	0	0.11	0.544	0	0.88	0.451	0	0.11	0.005
Under – Utilization1001[23$\rightarrow$]	0	0.11	0.528	0	0.22	0.426	0	0.77	0.046

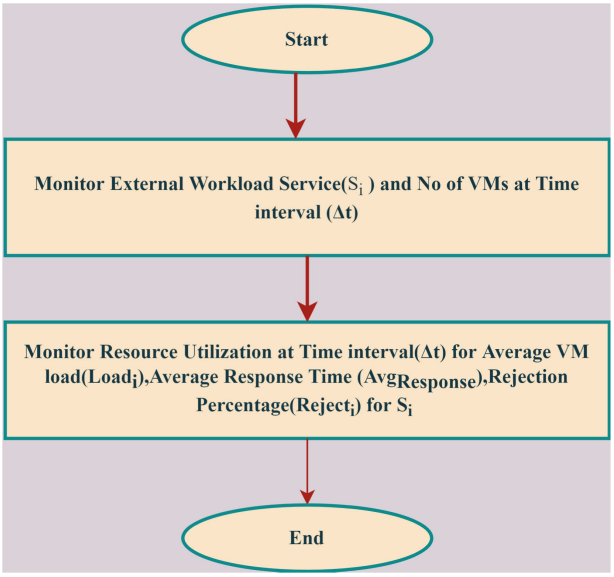

FIGURE 7. Flowchart for monitoring phase (Si).

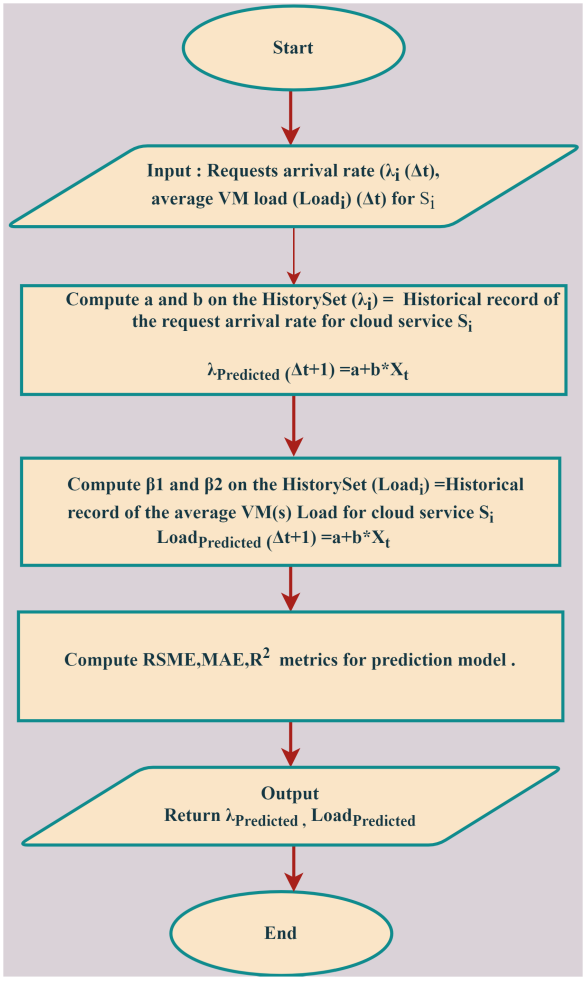

FIGURE 8. Flowchart for analysis phase (Si).

1) MONITORING PHASE

The gathering of metrics during the monitoring phase is performed in various modules as presented in Figure 7. Within the user monitoring module, every SaaS service offered by the provider S_i maintains a log of the number of virtual machines assigned by the IaaS provider for operating the service at specified intervals which is denoted by $Num_i(t)$ for every Δt. This information is collected through the user monitoring module, and helps to analyze the resource utilization patterns of the cloud services and optimize their performance and cost-effectiveness. The data on the number of VMs allocated to each service can also be used for capacity planning, load balancing, and resource allocation in the cloud environment [7]. It also keeps track of the total requests sent to the cloud service S_i during the Δt^{th} interval, indicated by $W_i(t)$. The module for resource monitoring gathers a load of virtual machines $Load_i(t)$, assigned to each cloud service (S_i) every Δt^{th} interval. The knowledge base stores the gathered data for the upcoming processing.

2) ANALYSIS PHASE

This phase of the MAPE-K loop is extremely important because it determines how the resources will be used in the future based on the average virtual machine load $VMLoad_i$ and request arrival rate λ_i. The model based on linear regression (LRM), as shown in Eq. 19, is used to forecast the resource use at the next interval of time $(t + 1)$.

$$Z_{t+1} = a + b * P_t \tag{19}$$

where t stands for the sample observation's index and Z_{t+1} denotes the workload at that particular moment $(t + 1)$. The time when the sample was taken is represented by the variable T_t. Eqs. 20 and 21 provide the computation of the y-intercept and slope, denoted as q and r [7]. The number of observed samples is represented by n in this case.

$$q = \frac{(\sum Z)(\sum T^2)(\sum T)(\sum TZ)}{n(\sum T^2) - (\sum T)^2} \tag{20}$$

$$r = \frac{n(\sum TZ) - (\sum T)(\sum Z)}{n(\sum T^2) - (\sum T)^2} \tag{21}$$

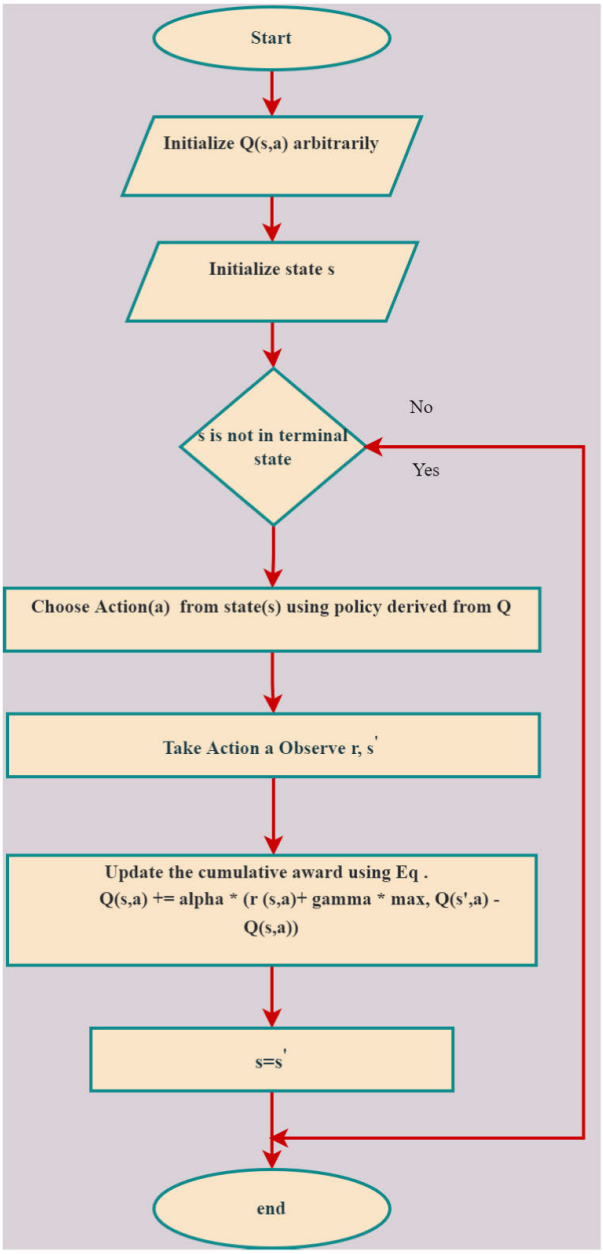

FIGURE 9. Flowchart for reinforcement learning (S_i).

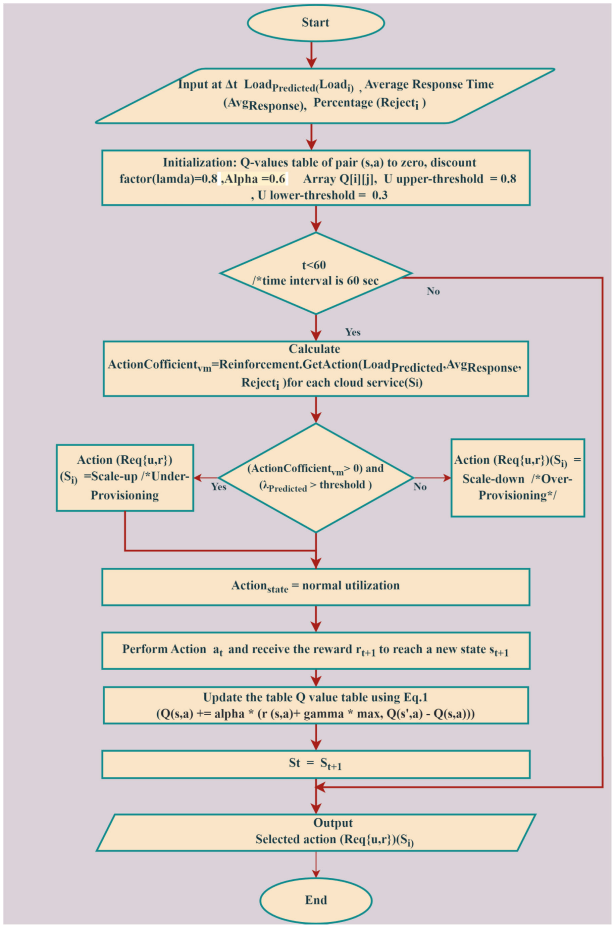

FIGURE 10. Flowchart for Planning Phase (S_i).

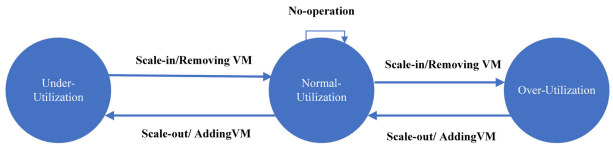

FIGURE 11. Flowchart for proposed Markov decision process with three states.

3) PLANNING PHASE

An RL-based approach has been proposed to decide the most appropriate line of action for selecting the size and placement of virtual machines. Typically, an MDP can be used to model RL-based techniques. An MDP can often be represented by the number of finite states and their transitions [23]. To model the system, we characterize the system states as an MDP, as seen in Figure 11; Its three states are normal utilization, underutilization, and overutilization [23]. These states demonstrate compliance with the performance criteria for resource supply, that the resource provisioning is excessive, and that the resource provisioning is insufficient.

Table 4 shows the proposed MDP's decision table for choosing one of the actions (such as scaling operation, scale-out operation, or no-operation) based on the current system state. Here, to determine the action-state pairings that result in higher rewards in the resource provisioning system, each pair can be modeled as a function of the likelihood $Reward = Q(s, a)$; the reward is referred to as a Q-value [23].

As described our research presents an RL algorithm termed Q-learning to identify the action-state pairings in the resource provisioning system that produce larger rewards. Q-learning is a straightforward RL algorithm that looks for the optimum course of action given the present state. Three simple steps make up the algorithm's operation, as presented below: (1) An action is started by the agent in a state, which is followed by a reward. (2) The agent has three options for the following

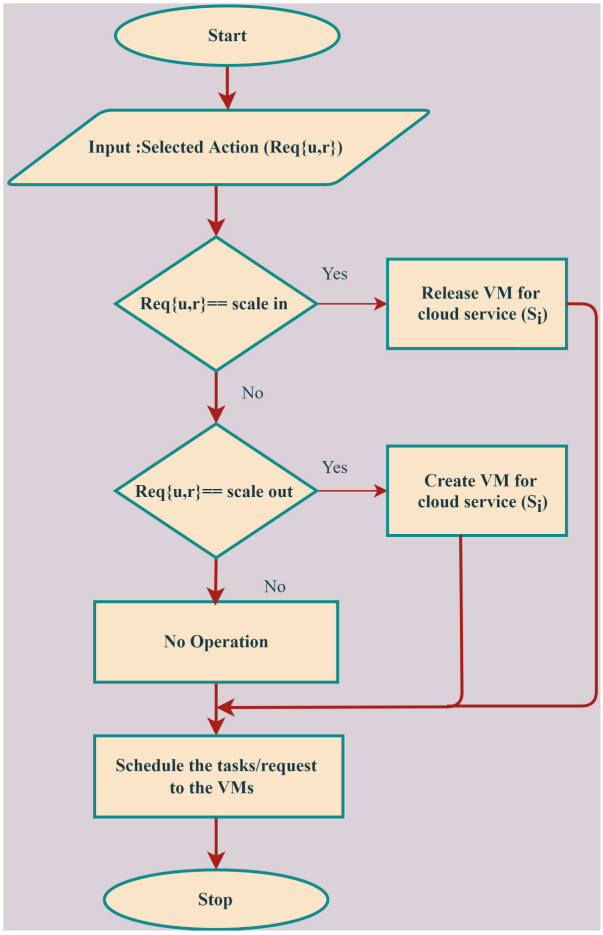

FIGURE 12. Flowchart for execution phase (Si).

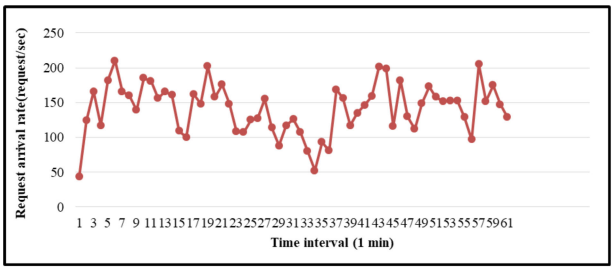

FIGURE 13. Workload patterns - ClarkNet.

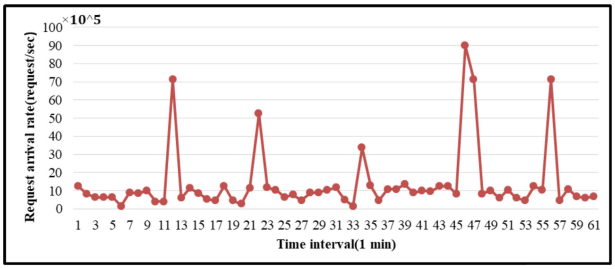

FIGURE 14. Workload patterns - Google cluster.

action: (a) take a random action; (b) update the Q-values; or (c) use the Q-table to choose the action with the highest value (i.e., Q[State, Action]). (3) Depending on the Q-value, the agent employs a reference table called a Q[State, Action] (Table 4) to decide the optimal course of action based on the values of $actionCofficient_{VM}$ and $\lambda_{Predicted}$ defined in the flowchart presented in Figure 10 [46]. The system status can be deduced from the predicted workload and the average response time [47]. The system is underutilized if the predicted workload and the average response times are below the lower threshold. The system is considered overutilized if the predicted workload or average response time utilization exceeds the upper limit. Otherwise, regular utilization of the system is considered. Using the reinforcement learning technique suggested in the planning phase, it is possible to choose the best options for the prediction provided by a VM utilizing the inputs from the prior stages. The projected workload and the typical response time are used to make predictions for VM scale-up, scale-down, or no operation using this proposed reinforcement learning model [9].

Table 4 is created based on the description given below:

1) Create the Q table from zero: The Q table is started with zeros in every cell. The activity is described in the columns, while the rows list the states. The dimensions of the rewards table and the Q-table are identical, yet they serve quite different purposes.
2) Update the cloud service's predicted workload and the average response time: The present state (i.e., s_t) is determined based on calculating the action coefficient at the time t. Based on the state, an action (from Table 4) is selected and implemented in the environment. After taking action, the RL agent receives feedback from the environment indicating the value of the (state, action) pair, denoted as r(s_t, a_t), which may result in a transition to the next state, represented by s_{t+1}. In this proposed MDP, the transition matrix is defined as the reward function (R(s, a)), as shown in Table 4 [23].
3) Determine the immediate reward: By acting on the current state for the initialized parameters, determine the instant reward. It is preferable if the virtual machines are idle rather than busy.
4) Updation of Q table: The Q-table is modified based on the estimated reward received from taking action. The updated values in the Q-table that reflect the reward received for an action are referred to as Q-values [44]. It consists of a mixture of state and action. This describes a specific state-action combination's quality. The higher Q-values are a requirement for better prizes. Finally, the best course of action for cloud service S_i in the next time interval is calculated by searching the next state in the updated Q-value as Table 4 [23] can be seen.
5) Ordering of jobs: According to the current scheduling policies, the tasks can be completed in any sequence, with the shortest assignment being completed first. Each task is assigned a processor, and each task is executed on allocated virtual machines.

TABLE 5. Parameters of virtual machine and parameters for cloudlets.

CPU (MIPS)	RAM	Size	Bw	VMM	VM Price ($ Per Hour)	PEs (No	Length	FileSize	OutputSize
1000	1.5 GB	1000 MB	1000 Mbps	Xen	4.02	1	Size of Task (MI)	300 kb	300 kb

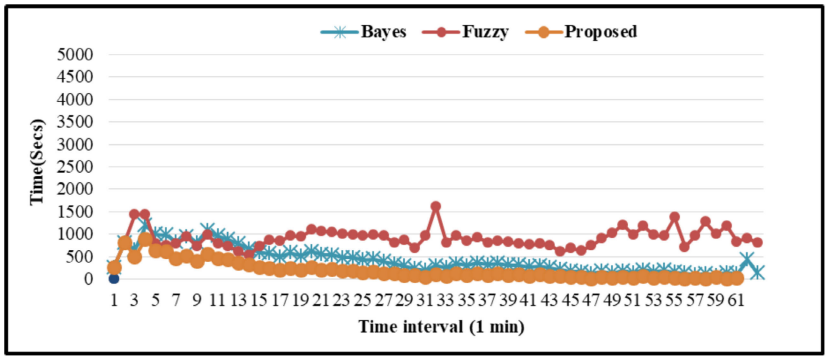

FIGURE 15. Virtual machine usage hour for ClarkNet dataset.

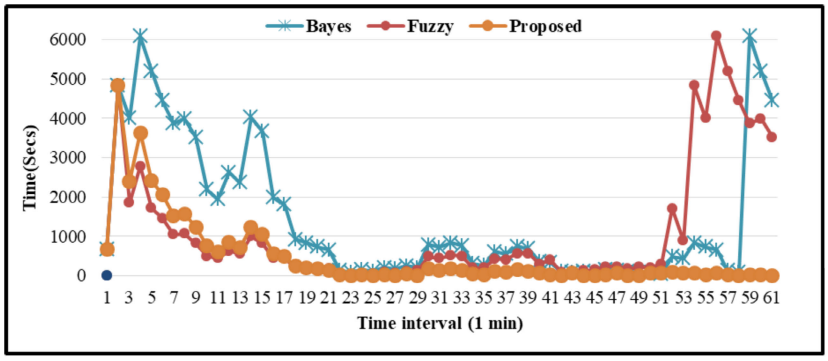

FIGURE 16. Virtual machine usage hour for Google dataset.

TABLE 6. Settings for cloudlet service.

Service Type)	Version	VM Description	No of Users
Cloud Service 1	Standard	Small	150
Cloud Service 2	Professional	Medium	250
Cloud Service 3	Enterprise	Large	350

The detailing of the above calculations is depicted in the flowcharts presented in Figures 9 and 10.

4) EXECUTION PHASE

The decision to switch the virtual machine on or off is made during the execution stage. Virtual machine management (VMM), as defined in Figure 12, produces a new VM for each cloud service S_i, if necessary, for effective resource provisioning and releases the inactive VM if it is not used or has low CPU use. VM load balancer thereby aids in scheduling tasks for a cloud service. S_i applies the best-fit strategy [7].

V. EXPERIMENTAL SETUP AND PERFORMANCE EVALUATION OF THE PROPOSED APPROACH

A framework for modelling and simulating cloud computing services, Cloudsim simulator, was used to develop the suggested strategy. [48].

We assume that an IaaS provider offers various virtual machines. Table 5 lists the configuration information of the VM used in this work with its cost and capacity.

Table 6 presents the cloudlet specification utilized in this simulation study. Our suggested approach has been tested using service-based applications to see how it performs compared to existing results.

The real-time workload traces generated by ClarkNet [49] and Google [50] were used to test the proposed method for several specified performance metrics stated in earlier sections. The trace encompasses information about every job that was submitted, scheduling decisions made, and resource utilization data for the jobs that were executed in those clusters. The patterns of the genuine load variations over time recorded from well-known websites of the identified workloads are represented in Figures 13 and 14.

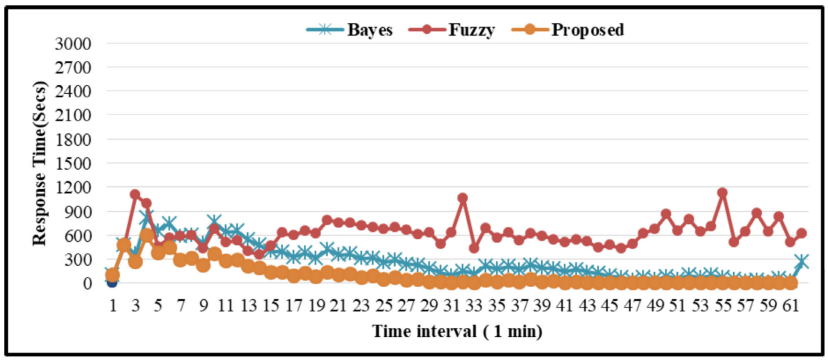

FIGURE 17. Average response time for ClarkNet dataset.

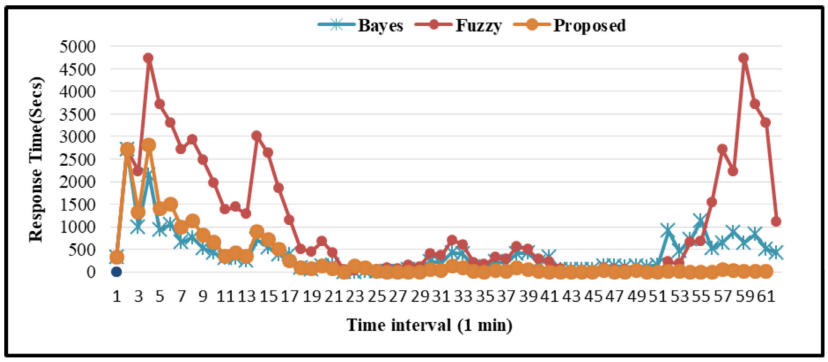

FIGURE 18. Result of the proposed model of Average response time for Google dataset.

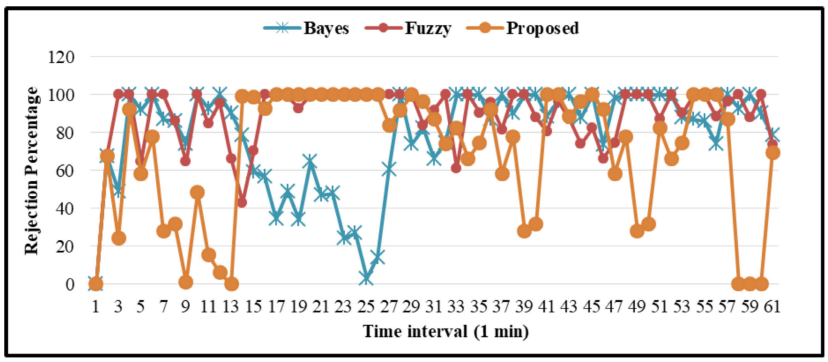

FIGURE 19. Average rejection percentage for ClarkNet dataset.

The ClarkNet traces and Google traces are trustworthy to be employed in a real cloud platform as they showcase a dynamic behavior that is relevant to the study. One-minute-long intervals are taken into account while calculating the workload. Leaving aside the 1-minute boot-up delay, the proposed reinforcement learning model based on the MAPE-K loop has been compared in terms of QoS parameters, namely processing time, average time for response, and rejection ratio [7]. Table 6 defines ''Edition'' as the version of the cloud service that is made available to users (standard, professional, and enterprise, for example). ''No of User'' refer to the highest quantity of users that are permitted to utilise the cloud service, limited by the type of cloud service edition. Higher editions with more users (like professional) are better suited for virtual machines with greater capabilities (like ''Extra large''). Additionally, 1-min intervals are used to calculate the time intervals. Consequently, there are 288 time intervals in a day. The time simulation used in this lasts for 1 hours. Furthermore, we consider time-shared scheduling to be the standard scheduling policy.

Figures 15 and 16 represent the comparison of the virtual machine (hours) for the proposed approach and fuzzy-based MAPE-K approach for ClarkNet and Google cluster traces

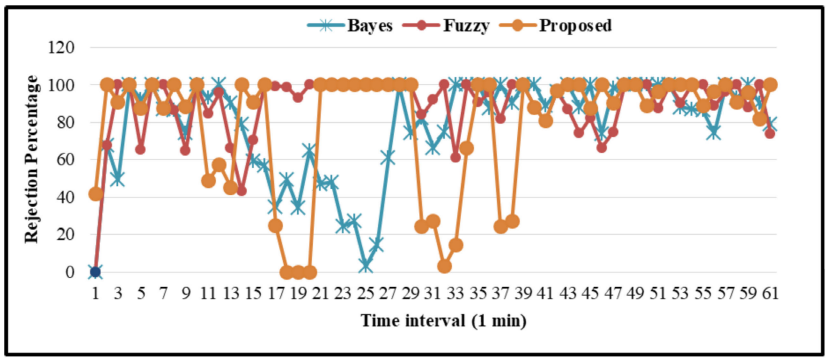

FIGURE 20. Average rejection percentage for Google dataset.

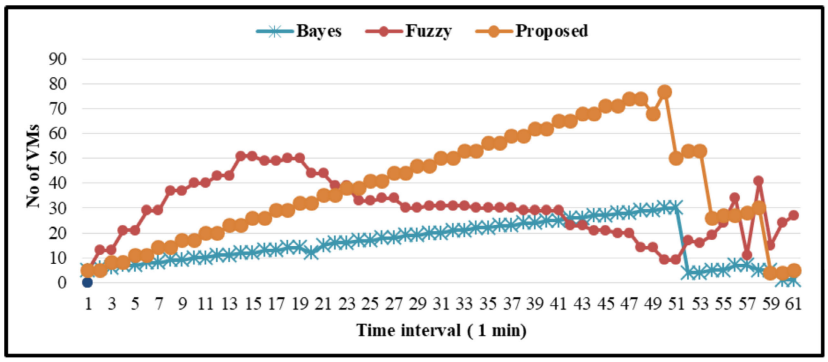

FIGURE 21. Average number of VMs for Claknet dataset.

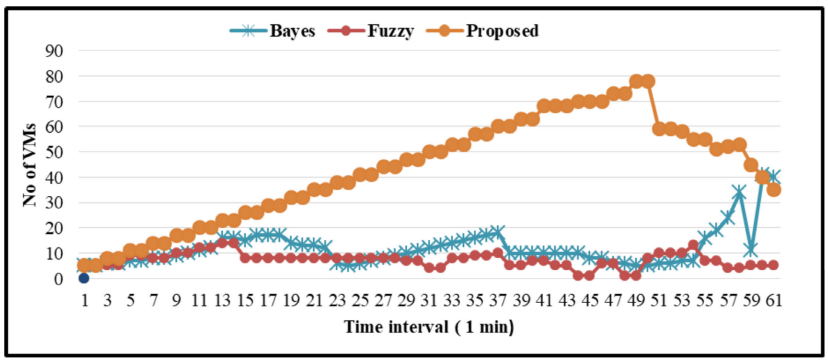

FIGURE 22. Average number of VMs for Google dataset.

at continuous time intervals t [51]. The virtual machine (avg) hours for both approaches have been listed in Table 7. Results indicate that the proposed approach outperforms the comparative approach concerning task response time. The response time of the task waiting in the queue gets reduced due to the early finishing time of the task by the VM.

Response time is a significant component in virtual machine provisioning since it lowers the provisioning expense. Figures 17 and 18 compare the traces concerning the response time (avg), which is also listed in Table 7. The waiting time of task during execution generally influences response time. According to [7], utilizing reinforcement learning as the decision-making mechanism leads to superior outcomes compared to the current implementations on fuzzy logic and Bayes.

SLA violations, another critical parameter for resource provisioning [7], can occur due to excessive resource utilization. It impedes the timely provision of cloud services in response to fresh inbound requests. Figures 19 and 20 compare the percentage of rejection of workload traces at each Δt interval. For ease of comparison, Table 7 presents the average percentage of SLA violations. This demonstrates that adjusting the rejection ratio will not resolve the SLA violation problem [25].

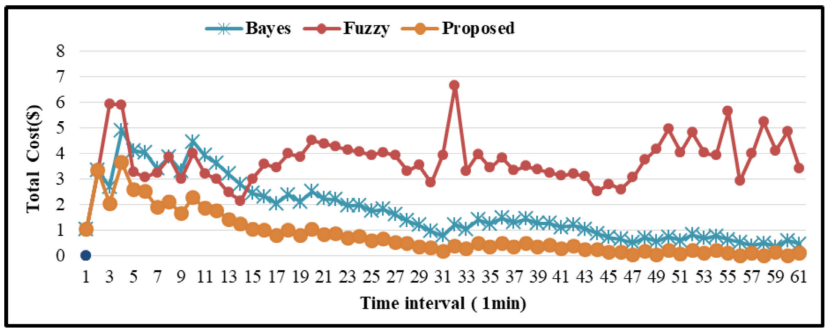

FIGURE 23. ClarkNet workload (Total Cost).

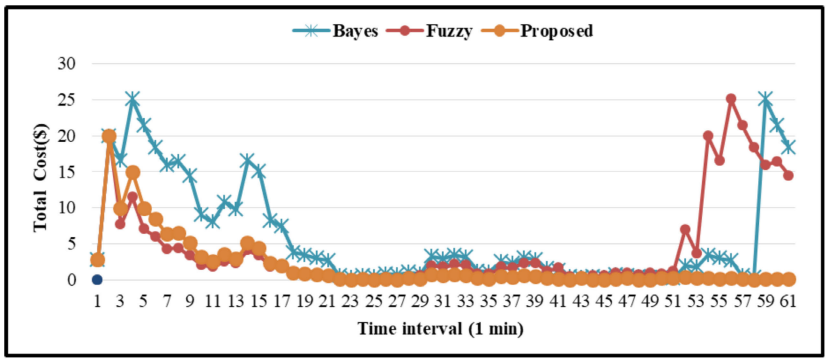

FIGURE 24. Google workload (Total Cost).

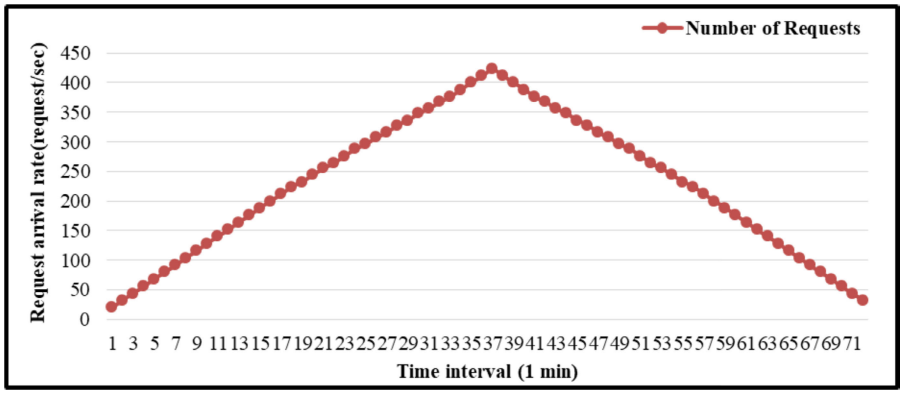

FIGURE 25. Workload patterns: smooth.

Figures 21 and 22 showcase the VMs assigned to both traces, with the consolidated list shown in Table 7. It could be observed that the proposed approach distributes the right amount of VMs at each period (i.e.) allocates about 80 VMs to deal with the incoming demand. As mentioned above, After the 36th interval, the suggested approach ensures the stability of the system, leading to reduced SLA violations and improved response time and job finish times. The number of VMs (avg) allocated throughout the test process is displayed in Table 8 which indicates that, despite utilizing more virtual machines, the proposed approach proves to be more efficient and effective in terms of virtual machine hours and total cost [25]. Thus the proposed technique reduces the SLA violations by 7.4% and increases the allocated VMs by 32%. Figures 23 and 24 present the total cost of allocating VMs for both approaches. As can be seen from Table 7, the suggested strategy uses an average of more virtual machines than the other approaches, supporting the idea that, in comparison to the other approaches, it produces the lowest average virtual machine hours even with the use of additional VMs. The suggested approach uses linear regression in the analysis phase and reinforcement learning as the decision-maker in the planning phase to accomplish these cost savings. Hence, the average use is lower in comparison, as shown in Table 7.

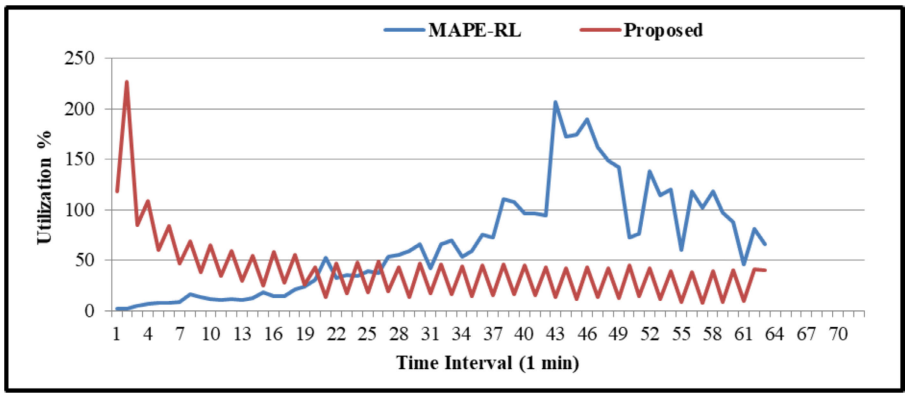

FIGURE 26. CPU utilization for smooth dataset.

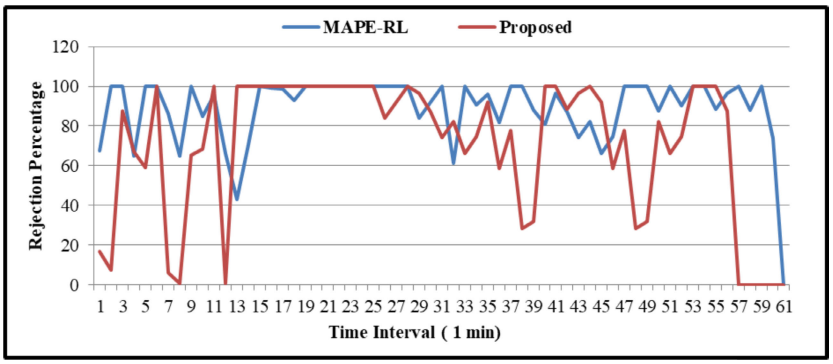

FIGURE 27. Average rejection percentage for smooth dataset.

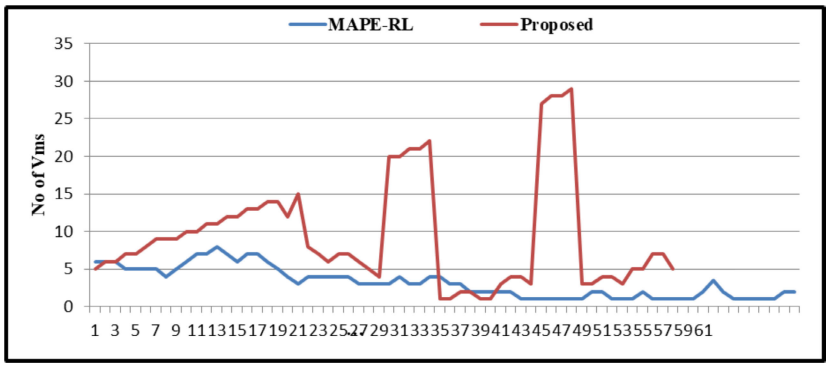

FIGURE 28. Average No of VMs for smooth dataset.

The following are the observations from the results: It can be seen that the proposed strategy uses virtual machines more efficiently than the fuzzy-based approach when comparing the average virtual machine hour listed in Table 7. This is because the jobs in the queue respond more quickly due to the implementation of the RL approach. The proposed approach reduces task waiting times because of the change made during the planning phase. The proposed strategy has a faster finish time than the existing approach thus showing a nominal improvement in the average time. The average rejection percentage is the third factor compared (to SLA violations). Again, it is clear that the suggestion of a probability-based decision maker [7] at the planning phase decreases SLA violations since it prevents unnecessary delays caused by numerous requests for cloud services, resulting in better service. Even though the difference is small, it will be more evident if it is tested on continuous data requests in real time. The suggested approach uses a minimum amount of VMs due to the observed decrease in SLA violations and improved response and finishing times. As a result of

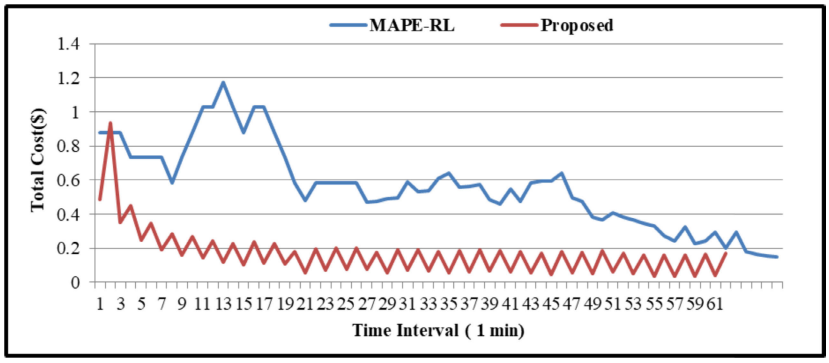

FIGURE 29. Total cost of VMs for smooth dataset.

TABLE 7. An analysis of the differences in the workloads between ClarkNet and Google traces.

Parameter - Workload	Fuzzy [25]	Bayesian [7]	Proposed
Hours of Virtual Machine (Avg Percentage) - ClarkNet	9.06	4.23	1.89
Hours of Virtual Machine (Avg) - Google	14.77	11.11	4.88
Time to Response (Avg Percentage) - ClarkNet	6.99	2.3	9.94
Time to Response (Avg Percentage) - Google	12.55	5.19	3.05
Percentage of Rejection (Avg) - ClarkNet	0.887	0.788	0.69
Percentage of Rejection (Avg)- Google	0.788	0.644	0.7721
Number of VMs (Avg Percentage) - ClarkNet	29.93	15.50	38.8
Number of VMs (Avg Percentage)- Google	11.75	7.33	43.01
Cost (Avg Percentage) - ClarkNet	3.72	1.74	0.77
Cost (Avg Percentage)- Google	5.99	4.5	1.98

TABLE 8. An analysis of the differences in the workloads between Smooth traces.

Parameter - Workload	MAPE-RL [23]	Proposed
Utilization (Avg Percentage) - Smooth	60.04	40.69
Percentage of SLA Violation (Avg) - Smooth	0.90	0.37
Number of VMs (Avg Percentage) - Smooth	3.42	9.43
Cost (Avg Percentage) - Smooth	0.600	0.167

the use of probabilistic reasoning, the suggested solution provides the required number of VMs at each interval. The chart shows that, in comparison to the fuzzy technique, the typical number of virtual machines required for execution was roughly reduced to 50%.

The proposed model's calculation cost was optimized by reducing the above-mentioned parameters. This demonstrates that the MAPE-K loop's use of reinforcement learning and linear regression beats fuzzy-based and Bayesian learning approaches in decision-making. Unlike the probabilistic approach, which captures partial knowledge and allows for learning and performance under uncertain future conditions, fuzzy logic captures partial truth in the data [7].

TABLE 9. Comparison of accuracy metrics.

Workload	Metrics	Fuzzy [25]	Proposed
ClarkNet	MSE	1980.21	1303.4
	RMSE	40.38	39.4
	R^2	0.789	0.053
Google Cluster	MSE	700.56	303.55
	RMSE	32.37	34.26
	R^2	0.556	0.244
Smooth	MSE	300.56	603.55
	RMSE	32.37	24.26
	R^2	0.656	0.1344

The dataset used for the proposed approach is Smooth workload (smooth variation of workload) which is obtained by normal distribution to model the incoming request arrival rate [37]. We evaluated the performance of the proposed approach under Smooth workload in Table 8 CPU utilizations of the two techniques for the Smooth workload at each interval are displayed in Figure 26 indicating that the proposed results outperforms other MAPE-RL approaches. The results show that the MAPE-RL wastes more resources in workload for the majority of intervals, whereas the suggested technique can utilize resources more optimally. There are times when the CPU utilization is higher than 100%. This is because the SaaS Provider is unable to handle all of the incoming requests for cloud services at that particular time, which results in under-provisioning (also known as over-utilization), which is an SLA violation. In terms of Utilization (40.69% vs. 60.04%), Percentage of SLA Violation (0.37% vs. 0.90%), Number of VMs (9.43% vs. 3.42%), and Cost (0.167% vs. 0.600%), the suggested solution performs better than MAPE-RL. These findings imply that the suggested approach performs better across a range of parameters, demonstrating its efficacy in workload management with smoother traces.

Figures 26 to 29 presents the result obtained on executing the proposed model on the Smooth dataset. In conclusion, the suggested strategy performs better for both real-time traces at all time intervals than other existing approaches. The suggested solution combines autonomic computing, reinforcement learning, and linear regression models, whereas the existing system is an elasticity-based resource provisioning architecture that incorporates fuzzy-based autonomic computing and linear regression models. The proposed (RLPRAF) solution employs an action-state strategy, whereas the current system employs fuzzy logic to enhance decision-making in uncertain scenarios. The suggested methodology addresses the undesired conditions of resource over and under-provisioning while outperforming the current method regarding response time [7].

The suggested regression model was tested for its Mean absolute error (MSE), Root Mean Squared Error (RMSE), and R-Squared (R^2) or Coefficient of determination metrics, with the findings provided in Table 9. Lower accuracy measures, except R^2, indicate higher regression model accuracy. When compared to the previous technique, the findings show that the suggested model has higher accuracy in terms of MSE and RMSE, as well as being more fit. However, the proposed model fails in terms of R^2. According to the literature, RMSE is the preferred method for comparing the accuracy of regression models and is commonly employed for this purpose. The reason for this is because RMSE measures how well a regression model can predict the absolute value of a response variable, but R-squared measures how well the predictor variables can explain the variation in the response variable. Because the objective of this work is prediction, we can disregard the R-squared error.

VI. CONCLUSION

This study proposes RLPRAF, a reinforcement learning-based proactive resource allocation framework that outperforms the existing fuzzy and Bayesian learning models, indicating that the suggested model holds great promise for forecasting cloud resource demands for cloud service providers. The method's effectiveness is evaluated by examining various parameters on the Google and ClarkNet traces. Based on reinforcement learning in the planning phase, this research presents an optimal dynamic solution to the problem of allocating resources effectively. To be more effective in conditions of dynamic workload, a solution has been presented as an algorithm to run at regular intervals. The suggested approach has been found to hold higher resource usage, shorter response times, and hardly any SLA violations. According to the findings, cloud service providers can use this model to distribute VM resources in advance based on workload estimations. In addition, the proposed model achieves optimal results when compared to the existing MAPERL approach. In the future, the blend of fuzzy logic and optimization methods could be experimented to yield better results.

REFERENCES

[1] D. Zhou, H. Chen, K. Shang, G. Cheng, J. Zhang, and H. Hu, "Cushion : A proactive resource provisioning method to mitigate SLO violations for containerized microservices," *IET Commun.*, vol. 16, no. 17, pp. 2105–2122, Oct. 2022. [Online]. Available: https://ietresearch.onlinelibrary.wiley.com/doi/abs/10.1049/cmu2.12464

[2] B. P. Dinachali, S. Jabbehdari, and H. H. S. Javadi, "A cost-aware approach for cloud federation formation," *Trans. Emerg. Telecommun. Technol.*, vol. 33, no. 12, p. e4631, Dec. 2022.

[3] C. Mommessin, R. Yang, N. V. Shakhlevich, X. Sun, S. Kumar, J. Xiao, and J. Xu, "Affinity-aware resource provisioning for long-running applications in shared clusters," *J. Parallel Distrib. Comput.*, vol. 177, pp. 1–16, Jul. 2023.

[4] R. Ranjan, L. Zhao, X. Wu, A. Liu, A. Quiroz, and M. Parashar, "Peer-to-peer cloud provisioning: Service discovery and load-balancing," in *Cloud Computing: Principles, Systems and Applications* (Computer Communications and Networks (CCN)). Springer, 2010, pp. 195–217.

[5] M. A. N. Saif, S. K. Niranjan, and H. D. E. Al-ariki, "Efficient autonomic and elastic resource management techniques in cloud environment: Taxonomy and analysis," *Wireless Netw.*, vol. 27, no. 4, pp. 2829–2866, May 2021.

[6] S. A. Khan, M. Abdullah, W. Iqbal, M. A. Butt, F. Bukhari, and S.-U. Hassan, "Automatic migration-enabled dynamic resource management for containerized workload," *IEEE Syst. J*, vol. 17, no. 2, pp. 2378–2389, Jan. 2022.

[7] R. Panwar and M. Supriya, "Dynamic resource provisioning for service-based cloud applications: A Bayesian learning approach," *J. Parallel Distrib. Comput.*, vol. 168, pp. 90–107, Oct. 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0743731522001307

[8] J. Pitt, *Self-Organising Multi-Agent Systems: Algorithmic Foundations of Cyber-Anarcho-Socialism*. Singapore: World Scientific, 2021.

[9] R. Panwar and M. Supriya, "Autonomic resource allocation frameworks for service-based cloud applications: A survey," in *Proc. Int. Conf. Comput., Commun., Intell. Syst. (ICCCIS)*, Oct. 2019, pp. 214–219.

[10] A. I. Orhean, F. Pop, and I. Raicu, "New scheduling approach using reinforcement learning for heterogeneous distributed systems," *J. Parallel Distrib. Comput.*, vol. 117, pp. 292–302, Jul. 2018.

[11] M. Etemadi, M. Ghobaei-Arani, and A. Shahidinejad, "Resource provisioning for IoT services in the fog computing environment: An autonomic approach," *Comput. Commun.*, vol. 161, pp. 109–131, Sep. 2020.

[12] X. Wang, Z. Du, Y. Chen, S. Li, D. Lan, G. Wang, and Y. Chen, "An autonomic provisioning framework for outsourcing data center based on virtual appliances," *Cluster Comput.*, vol. 11, no. 3, pp. 229–245, Sep. 2008.

[13] J. Yang, C. Liu, Y. Shang, B. Cheng, Z. Mao, C. Liu, L. Niu, and J. Chen, "A cost-aware auto-scaling approach using the workload prediction in service clouds," *Inf. Syst. Frontiers*, vol. 16, no. 1, pp. 7–18, Mar. 2014, doi: 10.1007/s10796-013-9459-0.

[14] T. Bhardwaj and S. Chander Sharma, "An efficient elasticity mechanism for server-based pervasive healthcare applications in cloud environment," in *Proc. IEEE 19th Int. Conf. High Perform. Comput. Commun. Workshops (HPCCWS)*, Dec. 2017, pp. 66–69.

[15] T. Bhardwaj and S. C. Sharma, "Fuzzy logic-based elasticity controller for autonomic resource provisioning in parallel scientific applications: A cloud computing perspective," *Comput. Electr. Eng.*, vol. 70, pp. 1049–1073, Aug. 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0045790617327180

[16] Z. Zhong, M. Xu, M. A. Rodriguez, C. Xu, and R. Buyya, "Machine learning-based orchestration of containers: A taxonomy and future directions," *ACM Comput. Surv.*, vol. 54, no. 10, pp. 1–35, Jan. 2022.

[17] A. Shakarami, M. Ghobaei-Arani, A. Shahidinejad, M. Masdari, and H. Shakarami, "Data replication schemes in cloud computing: A survey," *Cluster Comput.*, vol. 24, no. 3, pp. 2545–2579, Sep. 2021.

[18] H. Siar, S. H. Nabavi, and S. Shahaboddin, "Static task scheduling in cooperative distributed systems based on soft computing techniques," *Austral. J. Basic Appl. Sci.*, vol. 4, no. 6, pp. 1518–1526, 2010.

[19] A. Moazeni, R. Khorsand, and M. Ramezanpour, "Dynamic resource allocation using an adaptive multi-objective teaching-learning based optimization algorithm in cloud," *IEEE Access*, vol. 11, pp. 23407–23419, 2023.

[20] W. Fan, X. Liu, H. Yuan, N. Li, and Y. Liu, "Time-slotted task offloading and resource allocation for cloud-edge-end cooperative computing networks," *IEEE Trans. Mobile Comput*, vol. 23, no. 8, pp. 8225–8241, Jan. 2024.

[21] Y. Xu, J. Yao, H.-A. Jacobsen, and H. Guan, "Cost-efficient negotiation over multiple resources with reinforcement learning," in *Proc. IEEE/ACM 25th Int. Symp. Quality Service (IWQoS)*, Jun. 2017, pp. 1–6.

[22] Y. Bao, Y. Peng, and C. Wu, "Deep learning-based job placement in distributed machine learning clusters," in *Proc. IEEE INFOCOM Conf. Comput. Commun.*, Apr. 2019, pp. 505–513.

[23] M. Ghobaei-Arani, S. Jabbehdari, and M. A. Pourmina, "An autonomic resource provisioning approach for service-based cloud applications: A hybrid approach," *Future Gener. Comput. Syst.*, vol. 78, pp. 191–210, Jan. 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167739X17302327

[24] R. Buyya, C. Vecchiola, and S. T. Selvi, *Mastering Cloud Computing: Foundations and Applications Programming*. Newnes, 2013.

[25] T. Bhardwaj and S. C. Sharma, "An autonomic resource provisioning framework for efficient data collection in cloudlet-enabled wireless body area networks: A fuzzy-based proactive approach," *Soft Comput.*, vol. 23, no. 20, pp. 10361–10383, Oct. 2019, doi: 10.1007/s00500-018-3587-x.

[26] S. Wang, Y. Guo, N. Zhang, P. Yang, A. Zhou, and X. Shen, "Delay-aware microservice coordination in mobile edge computing: A reinforcement learning approach," *IEEE Trans. Mobile Comput.*, vol. 20, no. 3, pp. 939–951, Mar. 2021.

[27] S. Agarwal, M. A. Rodriguez, and R. Buyya, "A reinforcement learning approach to reduce serverless function cold start frequency," in *Proc. IEEE/ACM 21st Int. Symp. Cluster, Cloud Internet Comput. (CCGrid)*, May 2021, pp. 797–803.

[28] H. Qiu, S. S. Banerjee, S. Jha, Z. T. Kalbarczyk, and R. K. Iyer, "FIRM: An intelligent fine-grained resource management framework for SLO-oriented microservices," in *Proc. 14th USENIX Symp. Oper. Syst. Design Implement. (OSDI)*, 2020, pp. 805–825. [Online]. Available: https://www.usenix.org/conference/osdi20/presentation/qiu

[29] N. Dezhabad and S. Sharifian, "Learning-based dynamic scalable load-balanced firewall as a service in network function-virtualized cloud computing environments," *J. Supercomput.*, vol. 74, no. 7, pp. 3329–3358, Jul. 2018.

[30] S. Horovitz and Y. Arian, "Efficient cloud auto-scaling with SLA objective using Q-learning," in *Proc. IEEE 6th Int. Conf. Future Internet Things Cloud (FiCloud)*, Aug. 2018, pp. 85–92.

[31] Y. Wei, D. Kudenko, S. Liu, L. Pan, L. Wu, and X. Meng, "A reinforcement learning based auto-scaling approach for SaaS providers in dynamic cloud environment," *Math. Problems Eng.*, vol. 2019, pp. 1–11, Feb. 2019.

[32] Y. Garí, D. A. Monge, E. Pacini, C. Mateos, and C. García Garino, "Reinforcement learning-based application autoscaling in the cloud: A survey," *Eng. Appl. Artif. Intell.*, vol. 102, Jun. 2021, Art. no. 104288. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0952197621001354

[33] S. Zhang, T. Wu, M. Pan, C. Zhang, and Y. Yu, "A-SARSA: A predictive container auto-scaling algorithm based on reinforcement learning," in *Proc. IEEE Int. Conf. Web Services (ICWS)*, Oct. 2020, pp. 489–497.

[34] F. Rossi, M. Nardelli, and V. Cardellini, "Horizontal and vertical scaling of container-based applications using reinforcement learning," in *Proc. IEEE 12th Int. Conf. Cloud Comput. (CLOUD)*, Jul. 2019, pp. 329–338.

[35] J. Hu, K. Li, C. Liu, J. Chen, and K. Li, "Coalition formation for deadline-constrained resource procurement in cloud computing," *J. Parallel Distrib. Comput.*, vol. 149, pp. 1–12, Mar. 2021.

[36] M. Ghobaei-Arani, R. Khorsand, and M. Ramezanpour, "An autonomous resource provisioning framework for massively multiplayer online games in cloud environment," *J. Netw. Comput. Appl.*, vol. 142, pp. 76–97, Sep. 2019.

[37] M. Etemadi, M. Ghobaei-Arani, and A. Shahidinejad, "A cost-efficient auto-scaling mechanism for IoT applications in fog computing environment: A deep learning-based approach," *Cluster Comput.*, vol. 24, no. 4, pp. 3277–3292, Dec. 2021.

[38] M. Nazeri and R. Khorsand, "Energy aware resource provisioning for multi-criteria scheduling in cloud computing," *Cybern. Syst.*, vol. 24, pp. 1–30, May 2022.

[39] C. C. A. Vieira, L. F. Bittencourt, T. A. L. Genez, M. L. M. Peixoto, and E. R. M. Madeira, "RAaaS: Resource allocation as a service in multiple cloud providers," *J. Netw. Comput. Appl.*, vol. 221, Jan. 2024, Art. no. 103790.

[40] M. Cheng, J. Li, and S. Nazarian, "DRL-cloud: Deep reinforcement learning-based resource provisioning and task scheduling for cloud service providers," in *Proc. 23rd Asia South Pacific Design Autom. Conf. (ASP-DAC)*, Jan. 2018, pp. 129–134.

[41] Z. Tong, H. Chen, X. Deng, K. Li, and K. Li, "A scheduling scheme in the cloud computing environment using deep Q-learning," *Inf. Sci.*, vol. 512, pp. 1170–1191, Feb. 2020.

[42] S. Anilkumar, J. Jithish, and S. Sankaran, "Towards efficient resource provisioning in vehicular networks," in *Proc. Int. Conf. Technological Advancements Power Energy*, Dec. 2017, pp. 1–4.

[43] R. N. Calheiros, R. Ranjan, and R. Buyya, "Virtual machine provisioning based on analytical performance and QoS in cloud computing environments," in *Proc. Int. Conf. Parallel Process.*, Sep. 2011, pp. 295–304.

[44] A. Kaur, P. Singh, R. Singh Batth, and C. Peng Lim, "Deep-Q learning-based heterogeneous earliest finish time scheduling algorithm for scientific workflows in cloud," *Softw., Pract. Exper.*, vol. 52, no. 3, pp. 689–709, Mar. 2022.

[45] A. R. Hummaida, N. W. Paton, and R. Sakellariou, "Scalable virtual machine migration using reinforcement learning," *J. Grid Comput.*, vol. 20, no. 2, pp. 1–26, Jun. 2022.

[46] S. Otabek and J. Choi, "Twitter attribute classification with Q-learning on bitcoin price prediction," *IEEE Access*, vol. 10, pp. 96136–96148, 2022.

[47] A. A. Khaleq and I. Ra, "Intelligent autoscaling of microservices in the cloud for real-time applications," *IEEE Access*, vol. 9, pp. 35464–35476, 2021.

[48] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. F. De Rose, and R. Buyya, "CloudSim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," *Softw., Pract. Exper.*, vol. 41, no. 1, pp. 23–50, Jan. 2011.

[49] *Intelligent Autoscaling of Microservices in the Cloud for Real-Time Applications*. Accessed: 2021. [Online]. Available: http://ita.ee.lbl.gov/html/contrib/ClarkNet-HTTP.html

[50] *CloudSim: A Toolkit for Modeling and Simulation of Cloud Computing Environments and Evaluation of Resource Provisioning Algorithms*. Accessed: 2011. [Online]. Available: https://github.com/google/cluster-data

[51] T. Deepika, P. Prakash, and N. M. Dhanya, "Efficient resource prediction model for small and medium scale cloud data centers," *J. Intell. Fuzzy Syst.*, vol. 39, no. 3, pp. 4731–4747, Oct. 2020.

REENA PANWAR received the B.Tech. degree in information technology from the Bharat Institute of Technology, AKTU, in 2011, and the M.Tech. degree in computer science and engineering from Galgotias College of Engineering and Technology, AKTU, in 2015. She is currently pursuing the Ph.D. degree in computer science and engineering from the Amrita School of Computing, Bengaluru, Karnataka, India. She has three years of teaching experience. Her current research interests include distributed systems and cloud computing.

M. SUPRIYA received the Ph.D. degree in CSE from Amrita Vishwa Vidyapeetham, Bengaluru, India, in 2017. She is currently an Associate Professor with the Department of Computer Science and Engineering, Amrita Vishwa Vidyapeetham. She has more than 24 years of teaching experience, among which 14 years contribute to research. She has more than 60 Scopus-indexed publications in peer-reviewed journals and conferences. Her research interests include parallel/distributed computing, cloud computing, fog, and edge computing. She is an active ACM Professional member and leads various ACM club activities at the college.