International Journal of
INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING
ISSN:2147-6799 www.ijisae.org Original Research Paper

Pricing Model - Aware Task Scheduling in Cloud paradigm enhanced with DRL on MAPE Framework

Shruthi P S[*1] , Dr. D. R. Umesh [2]

Submitted: 26/01/2024 **Revised:** 04/03/2024 **Accepted:** 12/03/2024

Abstract: Cloud computing being a multifarious technology offers greater services to the end users, on a serious note several challenges need to be addressed to date even though numerous researches have been conducted in this field. Among the major issues like security, sustainability, availability, and many more, efficient resource allocation according to the requirement is important. Efficient resource allocation has become a critical issue in cloud computing because over-provisioning increases financial risk both at the provider's site and end-user; under-provisioning increases the service latency it may violate service level agreements eventually providers lose their customers. Hence many research works are going on to come up with an optimal solution for resource allocation in the cloud paradigm in different ways, like load balancing, workflow scheduling, container positioning, QOS parameter-based scheduling, etc. Our work addresses the task secluding by using the Deep Reinforcement technique corresponding with MAPE architecture, where the process of task scheduling is evolved over the various stages of MAPE architecture combining the VM pricing models (reserve, on-demand, and spot). We implemented our technique on the BitBrains dataset which consists of traces of 1750 VM. The results are discussed on variants of Reinforcement techniques and concluded the Reinforcement combining with neural network i.e DeepReinforcement technique with VM pricing model (DRL with PM) shows better results compared to other techniques. Our work's throughput is compared to those of others who achieved promising results, validating that our approach to task scheduling yields superior outcomes.

Keywords: *Reinforcement learning, Deep Reinforcement learning, MAPE architecture, VM Pricing model, Q learning*.

1. Introduction

Cloud computing, particularly the Infrastructure as a Service (IaaS) model, offers reliability, availability, and scalability, facilitating the execution of diverse applications through a pay-as-you-go model. However, it introduces challenges in resource management due to factors like application heterogeneity, resource contention, and varying workload patterns. Successful resource management solutions aim to maintain user satisfaction by dynamically adjusting infrastructure to meet changing demands, but breaches the existing and policy created Service Level Agreements (SLAs) that also offers Quality of Service (QoS) can occur without a thorough understanding of environment dynamics and their impact on system performance.

Cloud workloads, especially those involving end-users, are dynamic and uncertain due to user actions and environmental factors. Predicting future workloads is challenging, requiring consideration of factors like user behaviour and application specifics. Adaptable decision-making mechanisms are essential for dynamically scaling resources to ensure Quality of Service (QoS) satisfaction amidst diverse performance-related issues. Auto-scaling strategies aim to optimize objectives like execution time, cost-effectiveness, and adherence to Service Level Agreements (SLAs), facing challenges in scaling and scheduling optimization.

1. Scaling Phase: In the scaling stage, the goal is to dynamically adapt the quantity and types of cloud resources, such as virtual machines (VMs), based on the changing demands of the application. This involves acquiring or releasing resources to align with application requirements. Both the number and types of resources need to be optimized.

2. Scheduling Phase: The scheduling stage deals with the assignment of individual application tasks to the acquired resources. This task allocation process aims to maximize resource utilization and overall system efficiency.

Autoscaling strategies must continuously refresh their information to make informed decisions, taking into account the following factors:

Application displays changing responsibility designs at various stages. Models used to appraise task spans are innately flawed.

Cloud foundations are set apart by fluctuating execution levels.

Autoscaling strategies, depicted in Figure 1, require continuous monitoring of both infrastructure and applications. This ongoing observation helps mitigate disparities between estimated task information and actual execution progress, enabling the scaling and scheduling decisions during runtime. These strategies operate on a periodic execution pattern, making adjustments to instance numbers for each VM type and price model within each update interval, while efficiently allocating tasks to available VMs.

1 PES College of Engineering, Mandya, Karnataka,-571401, INDIA
ORCID ID : 0009-0000-6002-4838
2 PES College of Engineering, Mandya, Karnataka,-571401, INDIA
ORCID ID : 0000-0002-6813-3876
** Corresponding Author Email: shruthips@pesce.ac.in*

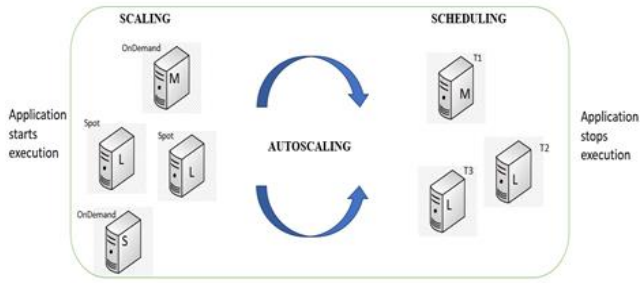

Fig. 1. Autoscaling process that incorporates the scaling and planning subprocesses

The subproblems of autoscaling, namely task scheduling and scaling, are both NP-hard due to their computational complexity. The inherent variability in cloud performance adds uncertainty to application execution. Our primary goal is to enhance cloud performance through autoscaling, with a focus on employing Reinforcement Learning (RL) techniques for task scheduling in this paper. Scaling considerations will be addressed in future research. Various methods, such as Dynamic programming, Likelihood calculations, Heuristic strategies, Meta-Heuristic calculations, Hybrid Algorithms, and ML techniques, have been investigated to address issues.

1.1. Reinforcement Learning

Reinforcement Learning (RL), [2] referenced in works is a learning process that aims 1) to generate trial-and-error in dynamic environment with recursive and irritative process 2) optimal action that meets that state of art irrelatively. The RL technique is driven discussion process that is been modelled referring to MDP but it manages with compromising any number of states and actions namely S and A respectively and a reinforcement signal, typically referred to as reward. A reward function is represented as R: S × A → R and a state transition function T: S × A → π(S) [3]. Here, π(S) represents a probability distribution over the set of states. The primary goal of the RL agent is to find an ideal approach, meant as 'π,' which directs the specialist in choosing activities that boost the drawn out aggregate award [4].

As seen in Fig. 2, when the framework is at time 't' with a state st from the state space S, the specialist pursues a choice by choosing an action at from the activity space A. Following the application of action at to the environment, the state changes from st to st+1 and the climate gives a quick award rt+1 [5].

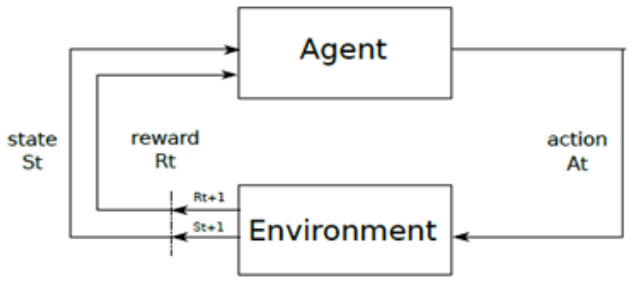

Fig. 2. Relationship between an RL agent and Environment

There have been broad conversations with respect to the use of ML in the space of distributed computing or cloud scheduling [4]. As of recent, profound support learning (DRL), an innovative approach that falls within the realm of machine learning, has gained prominence for addressing cloud computing scheduling challenges. To handle continuous state spaces and continuous time-varying environments, one approach is to use nonhomogeneous Markov processes. However, using them often involves solving complex differential equations with variable coefficients, which can be challenging and computationally expensive.

Figure 3 show the model of RL framework with neural network i.e DRL. DRL leverages the strengths of deep neural networks and reinforcement learning, demonstrating notable advantages in various scenarios, particularly within the intricate landscape of cloud computing [5]. Our research is among the first to address both scheduling and scaling simultaneously for autoscaling in the cloud environment using reinforcement learning (RL) techniques.

is conceptualized as a control loop where learning occurs gradually through trial and error, driven by feedback, enhancing the system's autonomy and adaptability to changes. This adaptive quality is crucial in uncertain environments where prior knowledge is vague or incomplete. At each decision or action, there's potential to modify the state of art in the environment and rigorously considering the recent feedback.

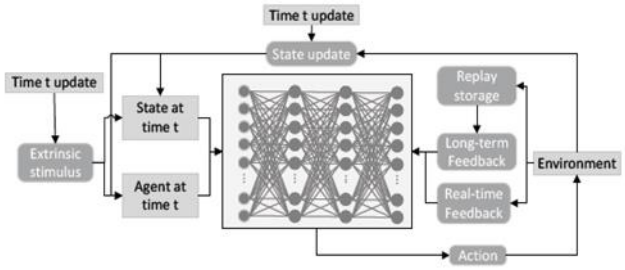

Fig. 3. RL framework with neural network-based decision maker

1.2. Research Gaps

In this paper, we discuss about neural network-based reinforcement learning system (DNN-Deep Neural Network) to resolve some gaps of task scheduling in cloud. They are as follows:

1) Although the RL paradigm appears to align with our problem, it important to determining the appropriate timing for action initiation and the action type [6].

2)The poor initial convergence [7].

3) Focusing on the evaluation of state-action values is crucial for initiating new actions when dealing with expensive state-action spaces [1].

The aforementioned are the identified main issues that greatly increase the dimensionality of the state-action space and the complexity of the problem.

1.3. Paper Organization

Section II delves into the background and explores related research. In Section III, we present the decision-making system for task scheduling during the planning phase. Furthermore, this section outlines the system framework and introduces the proposed algorithm. Section IV provides comprehensive details on the experimental results, elucidates the evaluation metrics used, and conducts an in-depth analysis of the outcomes. Section V serves as the conclusion of this work.

2. Related works

In the section, works done by various authors have been discussed and gaps in their work is identified to work on as a part of my research work. Finally, a comparative statement is drawn out of the

observations. The author of the work [10] as proposed RLPAS (Reinforcement Learning based Proactive Auto-Scaler) calculation, drawing motivation from the laid out RL- SARSA algorithm. This adaptation allows RLPAS to simultaneously learn about the environment and manage resource allocation. Through rigorous testing with real workloads, the RLPAS algorithm demonstrates superior performance compared to existing auto-scaling methods, exhibiting notable improvements in CPU utilization, response time, and throughput. The challenges of the work [11] revolves around devising effective strategies for dynamically allocating budgets in order to auto-scale scientific workflows within a cloud environment. The proposed approach involves implementing a Markov Decision Process (MDP) model [3], which enables the learning of novel policies by leveraging feedback, specifically information concerning the infrastructure's state and the execution progress of workflows. These insights are gathered from the execution of different policies within the auto-scaler, facilitating an iterative improvement process. This work as faced severe makespan and /or cost degradation. An innovative approach to auto-scaling cloud applications, leveraging the Q-learning technique is presents with the goal of helping stack of software As a Service an Assistance (SaaS) suppliers in settling on ideal asset portion choices inside a dynamic and flighty cloud environment [12]. The work incorporates various virtual machine (VM) pricing mechanisms, encompassing both the on-demand and reserved patterns. It is been acknowledge that auction-based pricing mechanisms play a significant role in the ever-evolving cloud trading market. The author of the work [13] implemented a decentralized architecture, which enhances the system's resilience in the event of a centralized controller failure. By distributing the responsibility for resource allocation to local controllers rather than relying on a single centralized controller, the system gains the ability to respond swiftly to sudden load spikes. The proposed approach in [14] is designed to exhibit dynamic adaptability in the face of uncertainties and workload spikes, effectively addressing the unwanted scenarios of overprovisioning and under-provisioning. This method is executed at regular intervals, allowing to finely tune the allocation of resources to align with the workload demands of cloud services. The author is [15] as introduces scheduling algorithm that dynamically allocates the resources that is dependent on Q-learning. The author in [16] has introduced a dynamic resource scheduling algorithm that relies on Q-learning. Recognizing the competitive nature of resource allocation among the PM nodes within the environment, a Q-learning algorithm that operates on a competition-based framework is devised. This approach enables the system to iteratively learn and refine the optimal resource scheduling strategy. The work [17] detail about a novel DeepBS solution effectively addresses the issue commonly encountered by RL algorithms. This is achieved through the implementation of our QoS Guarantor mechanism.

This section highlights that many previous studies have focused solely on scaling or scheduling challenges in the cloud, neglecting to address both simultaneously. However, our innovative approach treats scaling and scheduling as interconnected subproblems within the autoscaling framework, leading to remarkable results. While addressing only one of these subproblems partially resolves the autoscaling challenge, our contribution emphasizes the importance of concurrently tackling both aspects for a comprehensive solution. In this paper, we specifically address the scheduling problem.

Table 1. Comparative statement of the related works considered in the literature survey.

References	***Algorithms/ techniques***	***Problem***	***Objectives***
Seyed Mohammad Reza Nouri, 2018,[7]	*Q Learning[7]*	*Scaling*	*Reduces SLA violation, minimizing cost*
Junjie Cen, 2022 [9]	*ModifiedQ-learning[9]*	*Scaling*	*Minimizing System Delay*
J. V. Bibal Benifa, D. Dejey ,2019 [10]	*SARSA [10]*	*Scaling*	*Increase CPU utilization, increase response time and increase throughput*
Yisel Garı´, 2019,[11]	*Q learning[11]*	*Scaling*	*Increase Make span and reduce execution cost*
YiWei , 2019 [12]	*Q Learning[12]*	*Scaling*	*Reduce Cost*
Seyed Mohammad Reza Nouri, 2019 [13]	*Q learning[13]*	*Scaling*	*Reduce Cost and reduce SLA violation*
Mostafa Ghobaei-Arani a, 2018,[14]	*Q Learning [14]*	*Scaling*	*Reduces cost and increases resource utilization*
Xin Sui ,2019, [15]	*Support vector Regression*	*Scheduling*	*Reduces virtual machine migration and energy consumption*
Bin Wang, 2021,[16]	*Deep Q Learning*	*Scheduling*	*Reduce energy consumption and reduce SLA violation rate*
Xingjia Li [17]	*Deep Q Learning*	*Scheduling*	*Reduce cost*

The table 1 depicts that majority of the work is done in the area of scaling compared to scheduling and none of the works considered scaling and scheduling as two sub problems of autoscaling.

3. Proposed work

Our work incorporates the MAPE architecture [8], comprising Monitoring, Analysis, Planning, and Execution phases. Monitoring involves tracking metrics like CPU usage, memory usage, and network bandwidth. The Analysis phase identifies overutilized and

underutilized VMs. Planning decides task-to-VM mappings based on pricing models, followed by executing Load-Based VM scheduling policies. The integration of Deep Q Networks (DQN) with Reinforcement Learning techniques in the planning phase is crucial in our approach.

Figure 4 shows the MAPE architecture integrated with different components in the cloud paradigm. In the proposed approach, the planning phase of the Control MAPE means (Monitor, Analyze, Plan, Execute) circle is executed utilizing a RL-based specialist. This specialist ceaselessly changes the provisioned resources in view of the conditions of the framework including the current exhibition useful by property of Q Function i.e Q(S,A), which addresses the worth capability of the specialist at time t while making a move at in the present status st. $Q\pi(s_t,a_t)$ quantifies the total future rewards that result from the selection of action a_t while taking into account a discount factor λ and adhering to policy π within the current state s_t [9], addressed in equation 1 .

$$\boldsymbol{Q_{\pi}(s_t, a_t) = E_{\pi}\left[\sum_{i=0}^{T} \lambda^i \; r_{t+i+1} \mid S = s_t \; A = \; a_t\right]} \quad \textbf{(1)}$$

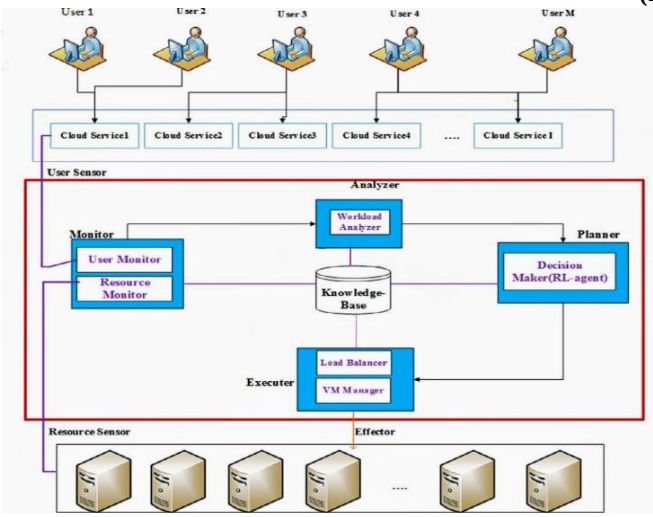

Fig. 4. MAPE architecture

3.1. Pricing model

The three significant price models are reserved, on-request and spot. The spot evaluating model absolutely rely upon the ongoing accessibility of the assets and bid cost model relies upon of the ongoing business sector. We have planned on-request and spot evaluating model. Cloud computing is pay as you go, only as costs arise example where assets are normally brought On-request in a rental length, signified as $Tperiod$, which < $T_{\text{reservation}}$, is a denotation applied for all VM instances that are treated to be reserved. Cloud service providers (CSPs) are considered without terminating their active reserved VM instances as long as these instances are until their reserved period is alive, $T_{\text{reservation}}$. The renting policy ($RP_{\Delta t}{}^{jR}$,$RP_{\Delta t}{}^{jO}$,$RP_{\Delta t}{}^{jS}$) at time Δt has two concepts to be considered one is the pricing model and the different VM types, where $RP_{\Delta t}{}^{jR}$ = number of reserved VM instances with type j for the period [Δt,Δt+1], $RP_{\Delta t}{}^{jO}$ = rented on-demand VM instances with respect to type *j* for a specific period, $RP_{\Delta t}{}^{jS}$ = new rented spot VM instances with respect to type *j*.

The Total Cost refers to the overall cost borne by the Cloud service provider to serve all requested services.

Total VM cost is the cost of all the VMs as given in equation 2.

$$\textbf{Total Cost} = \sum_{n=1}^{N} \textbf{VM } Cost_n \quad \textbf{(2)}$$

Each VM Cost depends on the renting policy at time t as expressed equation 3

$$\textbf{VM Cost}_n = \sum_{j=1}^{N}((VM_t^{jO} * C_t^{jO}) + P_t^{jO} + (VM_t^{jR} * C_t^{jR}) + P_t^{jR} + (VM_t^{js} * bid_t^j) + P_t^{js}) \quad \textbf{(3)}$$

C_t^{jO}, C_t^{jR} *and* bid_t^j cost of VM type j rented on-demand, the cost of reserved VM type j and the bid price of VM type j during [Δt, Δt+1] respectively. As per renting policy $RP_{\Delta t}{}^{jO} = VM_t{}^{jO}$ and VM_t^{jR} can be defined as in equation 4.

$$\boldsymbol{VM_t^{jR} = VM_t^{jAliveR} + RP_{\Delta t}^{jR}} \quad \textbf{(4)}$$

Table 2. Types of VMs

VM Types	Core	CPU(MIPS)	RAM (GB)	Disk	VM price($/hr)
t3.small	2	500	16	2GB	0.0224
t3.medium	2	1000	64	4GB	0.0448
t3.large	2	2000	128	8GB	0.0893
t3.xlarge	4	5000	256	16GB	0.1786

where $VM_t^{jAliveR}$ is the number of alive reserved VM instances with type j at time Δt.

It can be computed as shown in equation 5.

$$\boldsymbol{VM_t^{jAliveR} = VM_{t-1}^{jAliveR} + VM_t^{jNotAliveR}} \quad \textbf{(5)}$$

Where, $VM_{t-1}^{jAliveR}$ = number of in process reserved VM instances which are active with category type j at time Δt-1 and $VM_t^{jNotAliveR}$ = passive reserved VM instances with type j category until Δt.

If the delay of the user request is more than 0, then SLA is violated. The main aim of scaling is to reduce the total cost by maintaining SLA as equated in 6.

Minimize (Total cost) **(6)**

3.2. Pseudocode for MAPE architecture

The Fig.4 depicts the MAPE architecture, their interaction with external users and the cloud environment. The user request is fed to load balancer which distributed among the could services. The algorithm for MAPE architecture is shown in algorithm 1.

Algorithm 1: Pseudocode for MAPE architecture

Input: Initialize state with user model, cloud model and price model.

Output: scheduling i.e which involves mapping appropriate VM to the task corresponding to pricing model.

While (time period (Δt))

for each user request at the time interval Δt

Monitor user and cloud environment

Analyse user request and cloud capacity

Plan an action for scaling operation by minimizing the total cost and maintaining the SLA.

Execute the planned operation by reducing the makespan time.

End for

End while

3.3. Detailed Planning Phase

In a unique cloud market, SaaS suppliers face the test of enhancing asset portion in the midst of vulnerability. We tackle this by figuring out the issue as a Markov Decision Process (MDP) and utilizing Q-learning coordinated with neural computing networks Our self-adaptive algorithm minimizes costs in cloud resource allocation, with the task scheduling pseudo-code outlined in algorithm 2.

i) State space: The optimal action selection relies on the current observation denoted as x, which is a composite of two essential components: x_{server} and x_{task}. The x_{server} component encapsulates the present server state, encompassing available CPU resources (Av_CPU) and available memory resources (Av_MEM). These resources pertain to the Virtual Machines (VMs) currently running on the server. Conversely, x_{task} represents the existing task conditions, including the requested CPU resources (Rq_CPU), required memory (Rq_MEM), and task deadlines for multiple VM types and tasks. In this context, determining the optimal action entails making decisions based on these observations. These decisions may encompass resource allocation, VM placement, or task scheduling, all predicated on the current server state and the specific requirements of the tasks and VMs.

ii) Action space: The agent requires a defined set of actions to manipulate its environmental state. These actions should align with the system's objectives. Be that as it may, not all activities are proper for each state, requiring the foundation of a strategy to confine the accessible activities for each state. In our framework, picking a proper VM machine out the four accessible is the activity to be take in light of the evaluating model utilizing DRL strategy. Taking into account state $s \in S$, the at the same time activity set $A(s)$ list down every one of the vital moves that can be made. Likewise state s is communicated as a = (VMtjR,VMsjO,VMsjS), where VMtjR is the hold VM example of type j dispensed for the period Treservation, VMsjO , on-request VM cases of type j designated for the period Δt . VMsjS, is the spot VM occurrences of type j.

iii) Reward system: An adequate processing capacity should be maintained to accommodate the dynamic service requests generated by continuous workloads. When the current processing capacity falls short of meeting current workload demand, the resources will be overutilized and can be neutralized by scale-up. Conversely, if an excessive number of VM instances is allocated which causes underutilization and the processing capacity will far exceed the current workload. In this scenario, unnecessary expenses in renting VM instances will be incurred and also squanders valuable cloud resources which can be controlled by scale-down. Both of these inappropriate resource allocation behaviours have detrimental effects on overall system utility.

The above three points are treated as individual factors referring to the same in this paper an optimized reward function is described to address cloud providers utility function. The decision time is the crucial one for all the considered states S which can be denoted as Ti stating state s=(AW_{T_i}, VM_{T_i}), where $AW_{T_i}$is the average customer workload [$AW_{T_i} = avg(Rq_{\text{CPU}}, \text{Rq}_{\text{MEM}})$]and $VM_{T_i}$is the number of VM instances available in type [VM_{T_i}=avg(Av_CPU, Av_MEM)], an action a =(VM_s^{jR},VM_s^{jO}, VM_s^{jS}), and so the updated state s'=($AW_{T_{i+1}}$, $VM_{T_{i+1}}$), where active workload is W_{T_i}, then the below equation 7 defines reward function i.e R(s',a) .

R(s',a)=Profit(a)+Performance(a) [12] **(7)**

Profit(a)=Income- TotalCost **(8)**

TotalCost=$\sum_{n=1}^{N}$ VM $Cost_n$ **referred from equation (2)**

VM Costn= $\sum_{j=1}^{N}((VM_T^{jO} * C_T^{jO}) + P_T^{jO} + (VM_T^{jR} * C_T^{jR}) + P_T^{jR} + VM_T^{jS} * bid_T^j)$ **referred from equation (3)**

Penalty for on-demand and spot VM type is described in equations 9, 10

$$P_T^{jO} = \begin{cases} 0, & if\ RP_{T_i}^{jO} \le RP_{T_{i-1}}^{jO} \\ newC_T^{jR} * (RP_{T_i}^{jO} - RP_{T_{i-1}}^{jO}), & else \end{cases} \quad (9)$$

$$P_T^{jR} = \begin{cases} 0, & if\ RP_{T_i}^{jO} \le V_{T_i}^{j\_notAliveR} \\ newC_T^{jR} * (RP_{T_i}^{jO} - V_{T_i}^{j\_notAliveR}), & else \end{cases} \quad (10)$$

Equation (11) utility function that characterizes the mean utilization across all resources, with representing U_j^{max} the upper limit for each specific resource and U_j^{min} represents lower limit for ach specific resource, which means above U_j^{max} overutilization and below U_j^{min} as underutilization of a resource. Increased utilization has a beneficial effect on the final reward (defined in 1st and 2nd part of equation (11)). Nevertheless, when utilization surpasses the established minimum threshold leading to a detrimental influence on the ultimate reward (defined in 3rd part of equation(11)). Initial bonus and penalty value is assumed to be 2 and 1 respectively [12].

Performance(a)=

$$\text{VM}(u_j)=\begin{cases} bonus, U_j^{min} \le u_j \le U_j^{max} \\ bonus * \left(\frac{\sum_{j=1}^{N} U_j - U_j^{max}}{N}\right)^2, u_j > U_j^{max} \\ -penalty * \left(\frac{\sum_{j=1}^{N} U_j^{min} - u_j}{N}\right)^2, \quad u_j < U_j^{min} \end{cases} \quad (11)$$

Algorithm 2: Pseudo code for task scheduling

Input:

State space (s): Initialize the state space with cloud and user request information

Action space (a): Number of VMs of type j scheduled (includes on-demand and spot)

Load state space and action space into input nodes of neural network.

Output: Reward obtained for optimal resource allocation at the output node.

For each s,a introduce the table passage Q'(s,a) to nothing.

Notice the present status s

Do forever:

a`←get_action() using neural network and Q-value table. Execute the action

r←Compute_reward () as show in equation 10 through 12

Observe the new state s`

Update the table entry for Q`(s,a) as follows:

Q`(s,a)← r + γ max Q`(s`,a`) for all a

s←s`

In the Execution Phase of auto-scaling, scaling involves adjusting instances during planning, while scheduling tasks to active VMs occurs during execution. Minimizing makespan time is the planning phase's goal, achieved by prioritizing tasks with margin for delays. Critical tasks are given priority, with a policy implemented to allocate them to on-demand instances when

possible, minimizing the impact of failures. Future work aims to integrate scaling with scheduling.

4. Experimental Setup and Results Discussion

We conducted a comprehensive analysis using the GWA-T-12 Bitbrains dataset to schedule tasks to appropriate VMs based on the proposed pricing model, measuring the performance alone dedicated to terms of throughput, CPU utilization, cost, and an aggregate metric. We integrated the SDAEM-MMQ technique with the examined policies, chosen for its novel nature and promising results. The dataset comprises performance metrics from 1,750 VMs within a distributed datacenter operated by Bitbrains, catering to prominent institutions like major banks, credit card operators, and insurers. The fastStorage trace, containing 1,250 VMs, is linked to high-speed SAN devices, while the Rnd trace includes 500 VMs with varying storage performance characteristics. Our work focused on the fastStorage dataset, with each VM file recording 30 days of samples monitored every 5 minutes.

We have examined four distinct types of Amazon EC2 virtual machines (VMs), as detailed in Table 2. These Amazon EC2 instances have been configured with identical settings in the Cloud Sim simulator. In the table, Core indicates the number of available CPUs for each VM type, CPU (MIPS) addresses the computer processor limit in great many directions executed each second, and in Table 2 as shown with VM cost are related with each hour of calculation. Furthermore, spot examples share similar attributes as the on-request occurrences recorded in the table, however their costs fluctuate over the long haul. The gave information relates to the period spreading over from September eighteenth to November seventh, 2023, inside the AWS US-East (Ohio) district.

The RL algorithm operates without the need for a pre-existing model and, as it runs, it iteratively learns and refines policies by interacting with the target environment using new observations. Throughout the experiments, each agent's action set involves selecting the appropriate VM for the demanding tasks, with the individual VMs categorized by central processor and memory. To survey the viability of the support learning procedures in a true setting, our work is contrasted with a cutting edge booking strategies utilized in SDAEM-MMQ [16] by examination throughput boundary. In the work SDAEM-MMQ creator focused on the really decreasing energy utilization while keeping up with most reduced help level understanding infringement rate, throughput is one of the boundaries being viewed as in the process which we analyzed in our evaluating model based VM planning. The goal of this assessment is to guarantee the expanded throughput and high computer processor use with negligible expense while versatile scheduler is carried out..

4.1. Performance study

This subsection delves into the performance of DRL_with_PM across BitBrains workloads. Initially, it outlines the convergence outcomes of DRL _with _PM under diverse parameter configurations like different learning rate and Gama value. Subsequently, it evaluates the algorithm's performance against BitBrains VM traces considering the parameters like accuracy, precision, recall, f1 score and loss. Finally, an examination of the algorithm's performance under more various works is provided.

Figure 5 and Figure 6 shows our model training utilizing the request arrival pattern derived from BitBrains dataset and initially assess the impact of fundamental attributes defining convergence and parametric outcomes, in specific it describes only discount factor and the learning rate. These findings. Initially, we scrutinize the effect of various learning rates, as shown in Figure 5. During this examination, typically the parameters are set to their know values. Maximum learning rates result in more rapid attainment of larger episode rewards. Perphase inorder to attain excepted training outcomes excessively high learning rates will not always yield favorable training outcomes; where learning rate could be set to 0.1 led to diminished rewards for the agent and convergence that implies to a possible local optimum. To consider specialist reflections for all the impending potential compensation we really want to distinguish the rebate factor likewise the bigger markdown is straightforwardly corresponding to specialist's drawn out remunerations. A too enormous markdown factor suggests the specialist's accentuation on long haul rewards, possibly reaching out past the ongoing activity's impact, and dangers expanding the Q-esteem unnecessarily. In this examination, DRL with PM sets the gamma worth to 0.9, accomplishing ideal execution as far as combination and results is displayed in Figure 6.

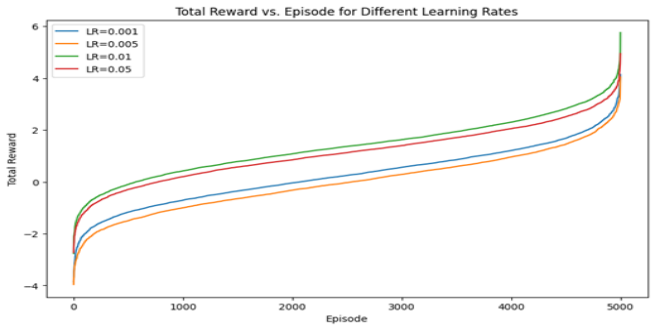

Fig. 5. Average reward for heterogenous learning rate.

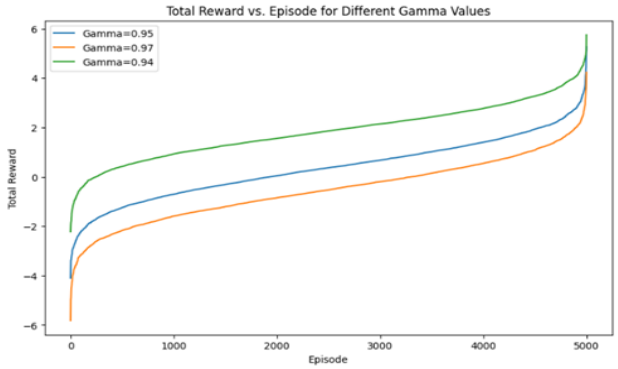

Fig. 6. Average reward consideration for unique gamma values.

In our study, we compared the performance of four distinct reinforcement learning (RL) algorithms. The comparison was based on their rewards accumulated over a series of episodes. Overall, the results demonstrate varying degrees of effectiveness among the different algorithms. Notably, Deep Reinforcement Learning (DRL) and Deep Reinforcement Learning with Pricing Model (DRL with PM) consistently outperform traditional Reinforcement Learning (RL) and Reinforcement Learning with Pricing Model (RL with PM) across all episodes. This observation suggests that the incorporation of deep neural networks enhances the learning capabilities of the agents, enabling them to achieve higher rewards more efficiently.

Table 3 shows the results of various evaluation metrics for all the variants of reinforcement learning algorithms. Furthermore, the introduction of the pricing model appears to have a positive impact on the performance of both RL and DRL algorithms.

Reinforcement Learning with Pricing Model (RL with PM) and Deep Reinforcement Learning with Pricing Model (DRL with PM) consistently outperform their counterparts without the pricing model, indicating the importance of incorporating domain-specific knowledge into the learning process.

Table 3: The results of various evalution metrics for all the varinats of reinforcement learning algorithms.

Algorithms	Accuracy threshold	Precision Marking	Recall factor	F1 Score	Loss
RL	0.79	0.79	0.79	0.79	3.219980239868164
DRL	0.85	0.85	0.85	0.8500000000000001	1.299985885620117
RL with PM	0.81	0.81	0.81	0.81	2.384927194792y
DRL with PM (Proposed)	0.8795959191838368	0.8795959191838368	0.8795959191838368	0.8795959191838368	0.8795959191838368

If accuracy, precision, recall, and F1 score all have the same value, it typically means that the model is performing equally well in terms of correctly classifying instances, both in terms of positive and negative classes, without favoring one over the other.

Figure 7 shows the performance metric comparison for various RL algorithm with the proposed work. The graph effectively illustrates the RL model's performance with VM pricing, demonstrating superior results compared to models without pricing considerations.

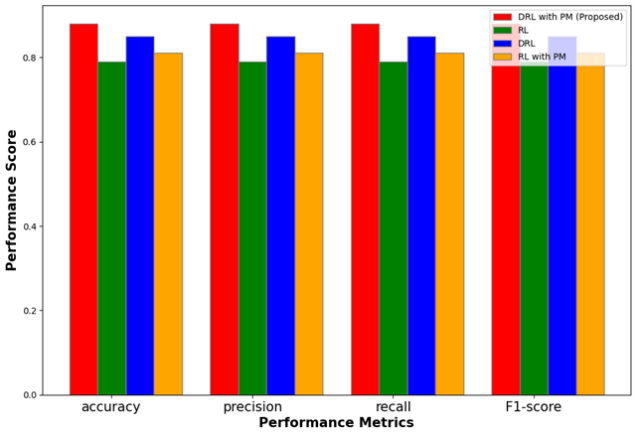

Fig. 7. Performance metric comparison for various RL algorithm with proposed work.

4.2. Models for comparison

To demonstrate the effectiveness of the deep reinforcement learning with pricing model framework for VM scheduling, we compared our algorithm, against various decision-making techniques under different scenarios of cloud paradigm.

VPBAR [16] employs conventional task arrival modelling and energy utilization modelling techniques in cloud, which severely restrict the model's adaptability in scenarios traffic bursts.

LRR_MMT [16] is explored upon a strong local regression evaluation for minimal migration with respect to time.

DthMf [16] a metaheuristic based on Genetic Algorithms (GA) facilitates power-aware VM request allocation across multiple sustainable cloud datacenters, considering power heterogeneity.

VMTA [16] is a scheduling strategy that relies on a method for predicting traffic burst workloads.

Megh [16] it is a type of RL algorithm associated with online involving in VM migrations to evaluate energy efficiency.

EQBFD [16] it is states the throughput for all the server integration.

SDAEM-MMQ [16] involves Q-learning for decision making and optimizing the energy.

Computational throughput refers to the processing speed of computational tasks within the cloud infrastructure. It measures the rate at which computational operations can be performed. Higher the computational throughput allows for faster execution of the tasks and improved performance of cloud-based applications. Optimizing throughput in a cloud scenario is essential for ensuring efficient and responsive cloud services. It involving optimizing network infrastructure, storage systems and computational resources to maximize data transfer rates, minimize latency and enhance overall performance for cloud users and applications. Throughput is considered as a comparison parameter for all the above mentioned works and its shown that proposed approach for VM scheduling using deep reinforcement learning with pricing model has higher throughput value.

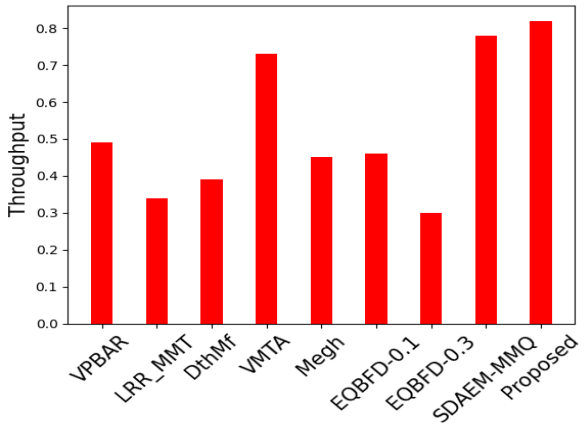

Fig. 8. Throughtput of various works compared with the proposed work.

Figure 8 illustrates that the decision-making system, enhanced with reinforcement learning techniques and variants, yields promising outcomes. The proposed approach and the nearer SDAEM-MMQ models demonstrate superior throughput values compared to other referred decision-making techniques for task scheduling.

5. Conclusion

In conclusion, the multifaceted nature of cloud computing offers extensive services to end-users, yet numerous challenges persist,

including security, sustainability, and availability. Among these challenges, efficient resource allocation remains paramount. Inefficient resource allocation can lead to financial risks for providers and end-users through over-provisioning, while under-provisioning can result in service latency, potentially violating service level agreements and leading to customer loss. To address this critical issue, researchers are exploring various approaches such as load balancing, workflow scheduling, container positioning, and QoS parameter-based scheduling. Our work focuses on task scheduling using Deep Reinforcement Learning within the MAPE (Monitor, Analyze, Plan, Execute) architecture, integrating VM pricing models (reserve, on-demand, and spot). We conducted experiments on the BitBrains dataset comprising 1750 VM traces monitored for 30 days with the interval of 5 minutes and evaluated the performance of different reinforcement techniques. Our findings demonstrate that combining reinforcement learning with neural networks, specifically the Deep Reinforcement technique, yields superior results compared to other approaches. Additionally, we compared our work's throughput with that of existing research, confirming that our task scheduling approach achieves promising outcomes. In summary, our research contributes to addressing the resource allocation challenge in cloud computing by proposing a novel approach to task scheduling. Our results underscore the efficacy of our approach and its potential to enhance overall system performance and user satisfaction in cloud environments. In future work, we plan to incorporate autoscaling capabilities into our cloud environment, encompassing both scheduling and scaling functionalities. This addition will serve as a significant advantage, enhancing overall cloud performance and facilitating optimal resource utilization.

6. References and footnotes

Acknowledgements

This research received support, either full or partial, from PES College of Engineering. We extend our appreciation to our colleagues at PES College who served as subject experts, offering invaluable insights and expertise that greatly contributed to this research. While they may not endorse all interpretations or conclusions presented in this paper, their assistance was indispensable. Sincere gratitude to the supervisor for unwavering support throughout the research process.

Author contributions

Shruthi.P.S: Conceptualization, Philosophy, Programming, Composing - Unique draft, Composing audit and Altering, Dr.D.R Umesh: Conceptualization, Technique, Approval, Composing - survey and altering ,Management.

Conflicts of Interest

The creators affirm that they don't have any contending monetary interests or individual connections that might have affected the work introduced in this paper.

References

[1] Sara Kardani-Moghaddam, Rajkumar Buyya," ADRL: A Hybrid Anomaly-Aware Deep Reinforcement Learning-Based Resource Scaling in Clouds", IEEE Transactions on Parallel and Distributed Systems, VOL. 32, NO. 3, March 2021.

[2] Yisel Garí, David A. Monge, Elina Pacini, Cristian Mateos, Carlos García Garino, "Reinforcement Learning-based Application Autoscaling in the Cloud: A Survey", arXiv:2001.09957, Top of Form https://doi.org/10.48550/arXiv.2001.09957,2020

[3] Mohit Kumar S.C. Sharma, Anubhav sGoel, S.P. Singh, "A comprehensive survey for scheduling techniques in cloud computing", Published by Elsevier Ltd Journal of Network and Computer Applications 143 (2019) 1–33, 2019.

[4] Raouf Boutaba, Mohammad A. Salahuddin, Noura Limam, Sara Ayoubi, Nashid Shahriar, Felipe Estrada-Solano and Oscar M. Caicedo, "A comprehensive survey on machine learning for networking: evolution, applications and research opportunities", Journal of Internet Services and Applications, by Springer Open,2018.

[5] Guangyao Zhou, Wenhong Tian, Rajkumar Buyya, "Deep Reinforcement Learning-based Methods for Resource Scheduling in Cloud Computing: A Review and Future", arXiv preprint arXiv:2105.04086, 2021.

[6] Teemu J. Ikonen, Keijo Heljanko, Iiro Harjunkoski, "Reinforcement learning of adaptive online rescheduling timing and computing time allocation" in Computers and Chemical Engineering volume 141, 25 June 2020.

[7] Seyed Mohammad Reza Nouri, Han Li Srikumar Venugopal, Wenxia Guo, MingYun He, Wenhong Tian "Autonomic decentralized elasticity based on a reinforcement learning controller for cloud applications", Elsevier Future Generation Computer Systems 94 (2019) 765–780, 2018.

[8] Merzoug Soltane, Yudith Cardinale, Rafael Angarita, Philippe Rosse, Marta Rukoz, Derdour Makhlouf, Kazar Okba, "A Self-adaptive Agent-based System for Cloud Platforms " , IEEE explorer 3rd International Conference on Pattern Analysis and Intelligent Systems (PAIS),2019.

[9] Junjie Cen and Yongbo Li," Resource Allocation Strategy Using Deep Reinforcement Learning in Cloud-Edge Collaborative Computing Environment" Hindawi Mobile Information Systems Volume 2022, Article ID 9597429, 10 pages.

[10] J. V. Bibal Benifa, D. Dejey, "RLPAS: Reinforcement Learning-based Proactive Auto-Scaler for Resource Provisioning in Cloud Environment", Springer Science+Business Media, Mobile Networks and Applications 24:1348–1363, 2019.

[11] Yisel Garı´, David A. Monge, Cristian Mateos, Carlos Garcı´a Garino, "Learning budget assignment policies for autoscaling scientific workflows in the cloud", Springer Science+Business Media, LLC, Cluster Computing (2020) 23:87–105.

[12] YiWei , Daniel Kudenko,Shijun Liu , Li Pan , LeiWu,Xiangxu Meng, "A Reinforcement Learning Based Auto-Scaling Approach for SaaS Providers in Dynamic Cloud Environment" Hindawi Mathematical Problems in Engineering Volume 2019, Article ID 5080647, 11 pages.

[13] Seyed Mohammad Reza Nouri, Han Li, Srikumar Venugopal, Wenxia Guo, MingYun He, Wenhong Tian, "Autonomic decentralized elasticity based on a reinforcement learning controller for cloud applications", Future Generation Computer Systems by Elsevier, 94 (2019) 765–780, 2018.

[14] Mostafa Ghobaei-Arani a, Sam Jabbehdari b, Mohammad Ali Pourmina," An autonomic resource provisioning approach for service-based cloud applications: A hybrid approach", Future Generation Computer Systems by Elsevier, 78 (2018) 191–210.

[15] Xin Sui, Dan Liu, Li Li, Huan Wang, Hongwei Yang, "Virtual machine scheduling strategy based on machine learning algorithms for load balancing" EURASIP Journal on Wireless Communications and Networking (2019) 2019:160 by Springer Open.

[16] Bin Wang, Fagui Liu, Weiwei Lin, "Energy-efficient VM scheduling based on deep reinforcement learning", Future Generation Computer Systems 125 (2021) 616–628 by Elsevier.

[17] Xingjia Li, Li Pan, Shijun Liu, "A DRL-based online VM scheduler for cost optimization in cloud brokers", Springer Science+Business Media, LLC by Springer, 2023.