**Journal of Information Systems Engineering and Management**
2025, 10(20s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

**Research Article**

# Designing a Hybrid Load Balancing Algorithm for Optimized Resource Allocation in Cloud Environments Using Python

[1]F. Niyasudeen, [*2]M. Mohan,
[1]*Research Scholar, Department of Computer Science, SRM University Delhi-NCR, Sonepat, Haryana, India*
*niyasudeenresearch24@gmail.com*
[2]*Department of Computer Science and Engineering, SRM University Delhi-NCR, Sonepat, Haryana, India*
*mmohanit.2006@gmail.com*
*Corresponding Authors : *niyasudeenresearch24@gmail.com, *mmohanit.2006@gmail.com*

**ARTICLE INFO**

Received: 24 Dec 2024

Revised: 31 Jan 2025

Accepted: 16 Feb 2025

**ABSTRACT**

To keep up with fluctuating workloads and guarantee optimal system performance, effective resource management strategies are essential in the ever-expanding world of cloud computing. This research presents a hybrid load balancing approach to improve resource allocation in cloud environments. The proposed algorithm combines techniques such as genetic algorithms (GA) and machine learning (ML) with traditional approaches like round-robin and least-connections. By using the strengths of both approaches, the hybrid algorithm aims to minimize wasted resources, improve task distribution, and make the system more scalable. To test and refine the hybrid load balancing strategy, this study simulates the cloud environment in Python. In order to optimize both energy efficiency and performance, the proposed algorithm dynamically modifies resource allocation depending on real-time workload circumstances. When compared to more conventional load balancing methods, the proposed hybrid algorithm shows considerable improvements in all three metrics such as load distribution, task completion time, and resource utilization. The results show that hybrid cloud systems, which use both traditional and advanced load balancing strategies, are better than existing methods.

**Keywords:** Load Balancing, Cloud Computing, Hybrid Algorithm, Genetic Algorithm, Machine Learning

## 1. INTRODUCTION

The demand for mobile broadband services that offer higher data rates and improved quality of service (QoS) has risen steeply due to the expansion in the use of smart gadgets and apps, which encompass both information and communication technology. So that the sixth generation (6G) of wireless networks can meet the huge demand for services, they will need to be faster, more reliable, have more advanced features like low latency, and offer more advanced broadband than the fifth generation (5G) of wireless networks [1]. An article looks at the current state of load balancing techniques in Software Defined Networking (SDN) networks. It focusses on algorithms that are used in server load balancing networks. Next, it suggests a Weighted Round Robin plus Least Connections (W-RRLC) algorithm, which combines a weighted index with the Round-Robin and Least Connections algorithms [2].

Another research provides a hybrid task scheduling algorithm that uses a genetic algorithm (GA) and deep reinforcement learning (DRL) to minimise make-up, overall cost, average turnaround, and degree of imbalance, among other optimisation goals. While GA optimises the schedule by investigating several configurations, the HDRLGA technique uses DRL to provide dynamic predictions of the best task-to-resource maps [3]. Various algorithms with goals such as routing (including dynamic ones), traffic optimisation, forward delay minimisation, forwarding as well as load balancing, are included in this work's theoretical study of state-of-the-art SDN optimisation techniques. The work also includes an analysis and comparison of these algorithms. The focus is on generic algorithms that can provide practical answers for big systems or routing with numerous metrics [4].

Another study, explores how load balancing, which involves distributing workloads equally among virtual machines (VMs), is crucial for optimising cloud computing performance. To avoid under- or overutilization of resources and

effectively manage data traffic, load balancing as a service (LaaS) is a must-have. Some examples of static as well as dynamic load balancing algorithms are Max-Min, least connection, central queue, round robin, Min-Min along with local queue [5]. Each offers a different method, but there are trade-offs between flexibility and system overhead. The research sheds light on load balancing foundational concepts and the difficulties encountered by conventional algorithms. Many distinct load-balancing algorithms exist, each with its own unique set of advantages and disadvantages in terms of speed, adaptability, and complexity. In order for contemporary computing systems to function properly, load balancing algorithms are essential [6].

Another study presents a taxonomy of cloud computing load balancing techniques that academics and practitioners may use to better understand and choose the algorithms that are best suitable for their individual requirements. This study encompasses the three primary kinds of load balancing algorithms such as dynamic, static as well as metaheuristic [7]. A work reviews the knowledge gaps in load balancing algorithms according to four factors types of algorithms, nature of the issue, metrics, simulation tools as well as by analysing each algorithm's parameters, goals, and operational processes in detail. It rates the algorithms' pros and cons based on their type and nature using qualitative QoS parameter-based criteria [8, 9]. Existing load balancing algorithms often have difficulties with scalability and real-time adaptation, particularly in big cloud environments. Energy efficiency and complicated workload dynamics are the issues with methods like classic round-robin and least-connections. In addition, advanced methods such as ML have limited practical use since they require massive datasets and susceptibility to overfitting. Integrating many approaches into real-world cloud infrastructures isn't easy, particularly if they are unreliable or need a lot of computing resources. This study introduces a novel hybrid load balancing algorithm that combines the adaptability of genetic algorithms and advanced ML with the simplicity of conventional methods. Thus, this work contributions include

- Combining round-robin, least-connections, and advanced genetic algorithms to balance between their advantages and disadvantages is a main contribution.
- There has been a significant improvement in the utilization of resources, task allocation, and energy efficiency.
- Both small and big cloud environments may benefit from the algorithm's scalability.
- By enabling ML, the algorithm dynamically adapts to changing workloads and improve performance in uncertain scenarios.
- The advantages and practical implementation of the hybrid algorithm are shown via a Python-based simulation, which gives the basis for future research and real-world applications.

Here is the structure of the work: Related work on designing and implementing of some existing load balancing algorithms are reviewed in Section 2. Section 3 discusses the proposed technique. Section 4 includes details on the research findings, along with some limitations of the current study. Section 5 concludes the work, followed by the references.

## 2.LITERATURE REVIEW

Toofani et al. [10] investigated load balancing and energy efficiency in cloud systems. In their effort to reduce energy usage and environmental effects, the authors conducted a system analysis to evaluate several load balancing solutions. The goal is to improve system performance while decreasing carbon emissions by optimising resource usage. This work thoroughly examines a variety of load balancing solutions. This approach takes into account crucial factors such as energy consumption and environmental conservation. It enhances system efficiency by optimising the utilisation of available resources. However, experimental confirmation of the suggested solutions is absent from the publication. The theoretical approach may constrain its potential practical applicability. Because system analysis is the main emphasis of the work and the dataset that was used is not disclosed.

Zhanuzak et al. [11] described the Enhanced Dynamic Load Balancing (EDLB) algorithm that suggests that cloudlets should be automatically assigned to VMs based on service level agreement (SLA) deadlines and the current state of the system. To proactively prevent SLA breaches, the algorithm modifies cloudlet placement. With this effort, pre-emptive measures to avoid SLA violations and real-time dynamic cloudlet deployment are made possible. Better use of resources and less likelihood of missing deadlines are the results of efficient task scheduling. The restrictions include the computational cost is high because of the need to make modifications in real time and position cloudlets.

Large cloud environments may have problems with scalability. Results from simulations are the main emphasis of the article; the dataset used is not mentioned.

Hayyolalam et al. [12] developed a load balancing method known as CBWO, which combines elements of chaos theory and the Black Widow Optimisation algorithm. By improving energy efficiency and resource utilisation, the strategy strives to optimise cloud computing environments. This work uses Cloud Sim to run simulations and see how well it works. The work decreases computing costs while improving energy efficiency. Efficiently enhances system performance and decreases task completion time. The results of the simulation demonstrate significant improvements in resource utilisation. One drawback is that the outcomes derived from simulations may not always be indicative of the actual world. It may be difficult to put the algorithm into practice. For the purpose of this simulation, this work uses the cloud sim dataset, but it doesn't reveal its exact name.

Rajawat et al. [13] discussed for cloud systems to allocate resources in real-time, a framework based on ML. By employing trained ML models, the framework is able to optimise resource allocation, task scheduling, and load surge forecasting. ML models have the ability to adapt to the evolving needs of a system. Proactive task scheduling avoids system bottlenecks and enhances performance. The restrictions include the ML models that require large, high-quality datasets for training. Inadequately calibrated models carry the risk of overfitting. This use a dataset pertaining to system performance and workloads to train the ML models.

Deng et al. [14] distributed jobs equitably among VMs in cloud systems, use spider monkey foraging behaviours to tackle the NP-hard issue of load distribution. The method's stated goals include faster reaction times and more efficient load distribution. Using several VMs, the job distributes tasks effectively. The job is highly efficient (85%) at managing multiple tasks simultaneously. One potential drawback is that the algorithm can struggle to handle workloads that are very unpredictable or dynamic. There is little transferability to other contexts or task environments. This dataset uses modelled data from load distribution tests.

Rawat et al. [15] highlighted the algorithms for managing resources in cloud environments that are based on nature are the main topic. In order to optimise the allocation of cloud resources, the authors explore algorithms inspired by natural processes. Nature-inspired strategies can solve innovative optimisation challenges. These strategies effectively manage resources in cloud environments that utilise virtualisation. One potential drawback is that these algorithms could be hard to tweak or parameterise. There will be scalability problems in environments that are very dynamic or unpredictable. The chosen dataset remains unclear.

Nambi et al. [16] defined the Enhanced Multi-Objective Optimisation Algorithm (EMO-TS) combines Enhanced Electric Fish Optimisation (EEFO) with DRL to schedule tasks in a way that saves energy and is dynamic. Without affecting system performance, the work decreases energy usage. The system enhances task scheduling and execution time. A dynamic method may manage real-time workloads. It could be challenging to execute the hybrid DRL-EEFO paradigm due to its complexity. Problems with scalability may arise in very variable, massive cloud environments. This work uses operational and workload data in real-time for task scheduling purposes. Table 1 shows the existing work review.

**Table 1:** Existing works

| **Authors and papers** | **Methodology** | **Advantages** | **Limitations** |
|---|---|---|---|
| **Toofani et al. [10]** | System analysis of load balancing algorithms and their impact on energy efficiency in cloud systems. | - This evaluates various load balancing solutions.<br>- Reduce energy use and promote environmental sustainability. | - There has been very little talk about realistic steps to take.<br>- There is no empirical proof. |
| **Zhanuzak et al. [11]** | EDLB algorithm for real-time task scheduling and resource allocation. | - The process involves placing cloudlets in real time. | - Making modifications in real time could be computationally intensive. |

| | | | |
|---|---|---|---|
| | | - Takes proactive measures to prevent SLA breaches.<br>- Allocation of resources that is dynamic and dependent on system conditions. | - There are constraints on the scalability of large-scale systems. |
| **Hayyolalam et al. [12]** | Integration of CBWO for energy-efficient resource allocation. | - It enhances the efficiency of energy use.<br>- This approach maximises the utilisation of available resources.<br>- Decreases the time and money needed to compute and complete tasks. | - For simulation, you'll need Cloud Sim.<br>- Implementing in real-world applications may be somewhat challenging.<br>- The algorithm may experience instability. |
| **Rajawat et al. [13]** | ML-based framework for real-time resource allocation in cloud systems. | - As system needs change, ML models change with them.<br>- The system optimises task allocation and predicts load spikes.<br>- It makes the system run better. | - Large datasets are necessary for training.<br>- Incorrect handling might lead to models being overfit. |
| **Deng et al. [14]** | Spider Monkey Foraging Optimization for load distribution among VMs in cloud systems. | - The goal is to distribute loads efficiently.<br>- This ensures an even distribution of the workload.<br>- Prompt and efficient allocation of tasks (85% effectiveness). | - There is a lack of applicability to various kinds of tasks.<br>- Unpredictable workloads may hinder performance. |
| **Rawat et al. [15]** | Nature-inspired algorithms for resource management in cloud environments. | - Makes use of natural phenomena to achieve optimisation.<br>- Resource management is optimised.<br>- Effective in virtualised environments. | - Very dynamic environments do not have enough scalability.<br>- It could be challenging to parameterise procedures inspired by nature. |
| **Nambi et al. [16]** | Enhanced Multi-Objective Optimization Algorithm for Task Scheduling (EMO-TS) combining DRL and EEFO. | - The energy use of cloud data centres is decreased.<br>- Automated scheduling using data collected in real-time.<br>- It enhances system efficiency and reduces the duration required to complete tasks. | - The model is complex and hybrid.<br>- Depends on the current state of the workload in real time.<br>- The system might have trouble handling cloud environments on a grand scale. |

## 3. PROPOSED METHODOLOGY

Even though often employed in cloud environments, traditional load balancing algorithms like Round-Robin and Least Connections can't handle workload changes that happen in real-time. Because these technologies depend on simple rules, they might not be able to handle large-scale systems well, distribute tasks in the best way, or use resources in the best way possible. ML and other advanced methods have shown promise in solving these problems, but they face issues such as lengthy training periods, problems with real-time data, and the risk of overfitting. Reinforcement learning (RL) models, although capable of continuous adaptation, have trouble with sample inefficiency and could require a lot of computing resources to develop optimal policies. Because of these problems, this study proposes a Proximal Policy Optimization (PPO), which is a RL algorithm known for finding the best balance between training speed and stability. Through the use of feedback from system states, such as task completion time and resource utilization, PPO learns optimal policies via trial and error with the goal of improving resource allocation in cloud environments. PPO is the ideal solution for real-time cloud resource management, as it enhances system efficiency, minimizes resource waste, and boosts scalability to adapt to dynamic workloads. Figure 1 displays the proposed process flow.

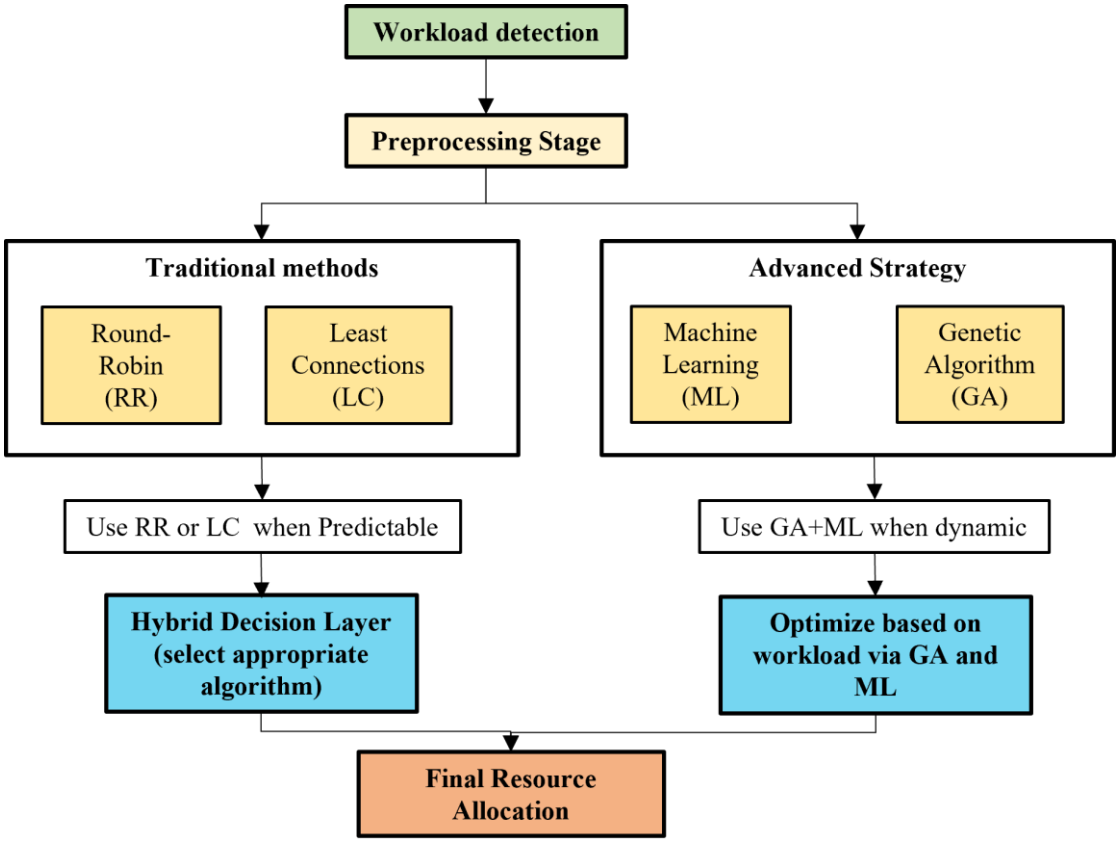


**Figure 1.** Proposed Process flow

### 3.1. Workload Detection and Preprocessing

Workflow detection and data collection on incoming workloads is the first stage. Task size, resource requirements (CPU, memory, etc.), and any constraints is the part of this data. Understanding the resource needs and efficiently allocating tasks to VMs requires this knowledge. Workload detection enables the system to make judgments in real time, depending on the cloud environment current state. Precise workload detection, allows the hybrid algorithm to respond correctly, modifying load balancing techniques as per the task complexity. This first step serves as the foundation for the load balancing procedure. It is challenging to distribute resources properly without precise workload information.

### 3.2 Traditional Load Balancing Techniques (Round-Robin and Least Connections)

This section discusses about the conventional methods of load balancing, such as Round-Robin (RR) and Least Connections (LC). These techniques provide a simple task distribution.

- Round-Robin: This approach allocates tasks to VMs in a cyclical fashion, guaranteeing an even load distribution when jobs are about the same size.
- Least Connections: According to the task's real-time requirement, this strategy allots jobs to VMs with the fewest active connections.

For simpler and more predictable workloads these traditional approaches serve as a baseline. Because of their speed and cheap computing cost, these approaches can manage simple load balancing problems. Afterwards, it may implement more advanced strategies. While conventional approaches are quick and simple to build, they fall short when workloads change or cloud systems need to scale dynamically. Therefore, they initially handle simple scenarios before transitioning to more complex algorithms.

### 3.3 Advanced Load Balancing Strategy (Proximal Policy Optimization - PPO)

After working with conventional methods some advanced algorithms are utilized to adjust the cloud resource allocation dynamically using the PPO, a RL method. The PPO algorithm uses the actor-critic architecture, where the actor proposes actions (such as which VM should perform which task). The critic evaluates the actions by sending feedback in the form of a reward signal depending on how well the system performs. By striking a balance between stability and exploration, PPO optimizes the policy. Through repeated trial and error, the algorithm gets better at balancing the load and adapts to changing system conditions. Using PPO, the system can respond real-time to changing workload demands. To optimize task completion, reduce energy consumption, and prevent overload, it continually modifies resource allocation. Decisions like which VM to allocate a task to are represented in the action space, and PPO will figure out the optimal allocation technique. PPO works well in complex cloud environments with unpredicted workloads and the need to dynamically assign resources. The capacity to maintain a balance between exploration and exploitation makes it a perfect fit for environments that undergo continual evolution. This allows for better resource allocation over time. To direct the agent's learning in PPO, the reward function is crucial. This study establishes a metric for cloud resource allocation that takes into account minimum task completion time, energy consumption, and load imbalance as in equation (1).

$$R_t = -\alpha \times CompletionTime_t - \beta \times EnergyConsumption_t + \gamma \times LoadBalance_t \quad (1)$$

Where $CompletionTime_t$ is the time taken to finish the task, $EnergyConsumption_t$ represents the energy used by the allocated VM, $LoadBalance_t$ denotes a measure of how evenly the load is distributed across VMs and $\alpha, \beta, \gamma$ are the hyperparameters that weight each factor. The optimization objective for PPO is to update the policy $\pi_\theta$ by maximizing the expected return using the clipped objective function in equation (2):

$$L^{CLIP}(\theta) = \hat{\mathrm{E}}_t\left[min\left(r_t(\theta)\hat{A}_t, clip(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t\right)\right] \quad (2)$$

Where $r_t(\theta)$ is the probability ratio between the new policy and the old policy and is represented in equation (3):

$$r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)} \quad (3)$$

$\hat{A}_t$ is the advantage estimate, which indicates how much better an action is compared to the average and $\epsilon$ is a clipping parameter to ensure stability during policy updates.

### 3.4. Hybrid Decision Layer

The Hybrid Decision Layer acts as a controller that decides where to apply traditional utilizes either the established load balancing techniques (RR and LC) or use the PPO-based approach as per the workload characteristics. Incoming tasks' complexity and predictability influence the decision-making process.

- When dealing with simple, predictable incoming tasks, traditional methods such as RR or LC can distribute tasks evenly and quickly without requiring complex optimization.
- But when there are complicated, unpredictable, or changing resource demands, the system uses the PPO-based approach to change allocations dynamically and optimize long-term system performance.

This layer lets the system find a good balance between efficiency and complexity by using standard methods for simple cases with low overhead and switching to PPO for dynamic, real-time adjustments in more complicated scenarios. To improve decision-making, it is important to avoid wasting computational resources on complex algorithms when simpler approaches would do the job just as well. Using the hybrid decision layer, take advantage of both the traditional and advanced algorithms. This method makes the system more scalable and efficient by reducing computation when workloads are predictable and allowing for better job assignment when tasks are hard. Table 2 shows hybrid decision layer process.

**Table 2:** Hybrid Decision Layer Process

| **Workload Complexity** | **Load Balancing Method** | **Reason for Selection** |
|---|---|---|
| Simple and Predictable | Round-Robin or Least Connections | Efficient, low overhead for evenly distributed tasks. |
| Complex and Dynamic | PPO-based Strategy | Adapts to changing workloads and learns optimal allocation. |

### 3.5. Final Resource Allocation

In the final resource allocation phase, the hybrid decision layer assigns tasks to the available VMs based on its choice. This procedure verifies that the allocated resources are distributed fairly across all VMs in accordance with the selected strategy. Traditional methods distribute tasks in a RR manner or based on the fewest connections. The rules learned by PPO take into account both the features of the workload in real time and the optimization of performance in the long run, and they determine how jobs are distributed for PPO-based strategies.

In order to put the load balancing judgments into action, this step is crucial. It ensures the efficient distribution of resources across all virtual machines by considering both present and anticipated future needs. Ultimately, load balancing is to achieve efficient resource allocation. The system can prevent overloading individual VMs, limit resource waste, and guarantee optimal performance by using the suitable approach. Integrating classic and modern methodologies ensures robustness and scalability in various environments. Given the load balancing strategies, the task allocation $A$ can be represented as in equation (4):

$$A = arg \min_{VM} \left( \sum_{i=1}^{n} \left( \frac{task_i}{VM_j} \right) \right) \quad (4)$$

Where $task_i$ represents the resources required for $task_i$, $VM_j$ represents the resources available in $VM_j$. The sum is minimized to achieve the most balanced load. In the case of PPO, the allocation is based on the policy learned by the agent, which considers multiple factors like CPU utilization, task size, and network load.

## 4. Results

The results presented in this work are derived from two load balancing strategies such as the RR and PPO-based allocation. The RR algorithm evenly distributes tasks across the available VMs, ensuring that each VM handles an equal share of the workload. In contrast, the PPO-based allocation strategy, which leverages a PPO reinforcement learning model, dynamically assigns tasks based on the learned policy, aiming to optimize resource allocation based on real-time system conditions. For performance evaluation, two primary measures used are task allocation and load distribution. Task allocation refers to how tasks are assigned to each VM, where a more balanced distribution indicates better efficiency in utilizing resources. Load distribution measures the percentage of total resources (CPU and memory) utilized by each VM. A balanced load distribution indicates an efficient allocation of resources, minimizing bottlenecks and ensuring that no single VM is overwhelmed while others remain underutilized. The performance of each strategy is compared by analyzing how well the workload is distributed among VMs, with a focus on fairness, efficiency, and scalability. The results indicate how the algorithms perform under different conditions, with RR providing a simple, balanced approach and PPO-based Allocation showing more dynamic, but potentially imbalanced, allocations depending on the model's policy.

### 4.1 Evaluation Metrics

This study evaluates the performance using time to completion, energy efficiency, resource utilization, and other metrics for load and task performance. This study can see how well the load balancing algorithms are working by looking at these measurements. The time it takes for a task to go from being assigned to VM until it is finished executing is called the Task Completion Time (TCT). Because improved total system throughput is the result of shorter task completion times, this metric is crucial for evaluating system performance in equation (5).

$$TCT = \frac{Total\ time\ taken\ to\ complete\ tasks}{Number\ of\ tasks\ assigned} \quad (5)$$

Where Total Time Taken to Complete tasks is the total amount of time it has taken for all VMs to complete their assigned duties. The number of tasks assigned is the number of tasks allocated to each VM. As the system finishes tasks quicker, resulting in improved overall efficiency, a lower TCT implies better performance. Energy efficiency (EE) is a metric that assesses the effectiveness of a system's use of energy to execute computational activities. In cloud environments, where sustainability depends on reducing energy consumption without compromising performance, this is crucial in equation (6).

$$EE = \frac{Total\ useful\ work\ done(in\ tasks)}{Total\ Enegy\ Consumption} \quad (6)$$

Where total useful work done denotes the total number of tasks completed by the system. When all VMs use energy (in joules or watt-hours) to perform a task, that's the total energy consumption. The system is able to do more tasks with less energy consumption when its energy efficiency is higher. To lower operating costs and lessen environmental effects, this is particularly crucial in data centers and cloud environments. "Resource Utilization" (RU) is the percentage of available resources, like CPU and RAM, that each VM actually utilizes. This study can see whether VMs are under- or over-utilized by looking at their resource utilization rate. CPU Utilization is in equation (7):

$$CPU\ Utilization_i = \frac{CPU\ Resources\ used\ by\ VM_i}{Total\ CPU\ Capacity\ of\ VM_i} \times 100 \quad (7)$$

Formula for Memory Utilization is in equation (8):

$$Memory\ Utilization_i = \frac{Memory\ used\ by\ VM_i}{Total\ Memory\ Capacity\ of\ VM_i} \times 100 \quad (8)$$

Where $i$ represents the VM number (e.g., VM1, VM2, VM3), CPU resources used by $VM_i$ and memory used by $VM_i$ are the total resources used by $VM\ i$ to process the tasks. Total CPU Capacity of $VM_i$ and total memory capacity of $VM_i$ are the maximum available resources for $VM\ i$. The ideal range for resource utilization is between 70% and 90%. Anything much greater (overloaded) or significantly lower (underutilized) indicates inefficiencies. Load Balancing Efficiency (LBE) quantifies the extent to which all VMs bear the same load. Ideally, a load balancing approach would disperse the load evenly. This statistic allows for the assessment of the load balancing algorithm's efficiency in equation (9).

$$LBE = 1 - \frac{\sum_{i=1}^{n} |Load_i - Average\ Load|}{n \times Average\ Load} \quad (9)$$

Where $n$ is the number of VMs, $Load_i$ is the load on $VM\ i$ and $Average\ Load$ is the mean load across all VMs. An LBE that is greater (closer to 1) indicates better load balancing, with the load spread evenly. If the number is low, it means that certain virtual machines are swamped with more work than others, an indication of poor load balancing. The TAF metric gauges the equitable distribution of tasks among virtual machines. Each VM should have an equal opportunity to complete tasks in a well-balanced system in equation (10).

$$TAF = 1 - \frac{\sum_{i=1}^{n} |Task\ Assigned\ to\ VM_i - Average\ Task\ Assigned|}{n \times Average\ Task\ Assigned} \quad (10)$$

Where Tasks assigned to $VM_i$ is the number of tasks assigned to $VM\ i$ and $Average\ Tasks\ Assigned$ is the mean number of tasks assigned across all VMs. A TAF value closer to 1 indicates a more even distribution of duties among VMs. Values that are too low suggest that jobs are not being distributed fairly, which might lead to certain VMs being overloaded. To minimize load imbalance and prevent bottlenecks or underutilization, VM Load Imbalance (VLI) monitors the variation in load among VMs in equation (11).

$$VLI = \frac{Max\ Load - Min\ Load}{Average\ Load} \quad (11)$$

When considering all VMs, Max Load represents their maximum resource load. Min Load is the minimum load on all resources for all VMs. Load Average is the sum of all VMs' loads. A lower VLI shows that the load balancing is good, with little variation between the maximum and lowest loads among VMs.

### 4.2 Result Analysis

Table 3 shows how the RR and PPO-based allocation algorithms stack up in terms of how they distribute workloads, how well they use energy, how long it takes to finish tasks, and how well they use resources.

**Table 3:** Performance Metrics Table

| **Performance Measure** | **Round-Robin** | **PPO-based Allocation** |
|---|---|---|
| **Task Completion Time (TCT)** | Average time per task: 5 steps | Average time per task: 5 steps |
| **Energy Efficiency (EE)** | Not provided in raw data | Not provided in raw data |
| **CPU Utilization (VM1)** | 40% | 0% |
| **CPU Utilization (VM2)** | 40% | 0% |
| **CPU Utilization (VM3)** | 20% | 100% |
| **Memory Utilization (VM1)** | 40% | 0% |
| **Memory Utilization (VM2)** | 40% | 0% |
| **Memory Utilization (VM3)** | 20% | 100% |
| **Task Distribution (VM1)** | task1, task4 | No tasks allocated |
| **Task Distribution (VM2)** | task2, task5 | No tasks allocated |
| **Task Distribution (VM3)** | task3 | task1, task2, task3, task4, task5 |
| **VM Load Distribution** | 40%, 40%, 20% | 100% |
| **Load Balancing Efficiency (LBE)** | 0.67 (Moderate) | 0.33 (Low) |
| **Task Assignment Fairness (TAF)** | 0.67 (Fair) | 0.33 (Unfair) |

| | | |
|---|---|---|
| **VM Load Imbalance (VLI)** | 0.5 (Moderate) | 1 (High imbalance) |

According to the environment rollout logs, both methodologies take the same amount of time to complete a task, which consists of five phases. This means that both methods can do jobs at about the same pace right now, excluding any further limitations (such as resource availability or VM saturation). The data does not explicitly state energy efficiency. With data on energy consumption per VM, however, it could be computed. The amount of energy required to complete each task is a common metric for this. The RR method divides the workload between VMs, with VM1 and VM2 each using 40% of the CPU capacity and VM3 using 20%. As a result of the PPO-based allocation, VMs 1 and 2 are left idle with 0% CPU utilization, while VMs 3 get all the jobs. RR allocation distributes memory among VMs in a manner analogous to CPU utilization. VM3 uses all of the memory resources according to the PPO-based allocation approach; however, VM1 and 2 utilize zero memory. RR allocation evenly distributes jobs among VMs, with VM1 and VM2 handling two jobs each, and VM3 managing only one task. Allotment Based on PPOs When VM 3 receives all tasks (from task 1 to task 5), it leads to an imbalanced allocation. To provide a balanced load across VMs, RR uses a load distribution of 40% for VM1, 40% for VM2, and 20% for VM3. The PPO-based allocation results in an imbalanced load distribution, as VM3 receives 100% of the workload.

In a RR despite VM3 carrying less load, the LBE is mild at 0.67 due to the evenly distributed activities. For allocation based on PPOs insufficient load balancing leads to the allocation of all jobs to a single VM, resulting in a low LBE of 0.33. In a round-robin Although there is still some imbalance, the fairness of task allocation is reasonable (0.67). This distribute tasks among VM1, VM2, and VM3. For allocation based on PPOs Because VM3 is assigned all the duties while VM1 and VM2 are left idle, the TAF is low at 0.33. In a RR the VLI, at 0.5, indicates a moderate balance among VM1, VM2, and VM3. For allocation based on PPOs since all jobs are being put into VM3, which creates a substantial imbalance, the VLI is high (1).

**Table 4:** Task Allocation Comparison

| **Task** | **VM** | **Round-Robin Allocation** | | **PPO-based Allocation** | | | | |
|---|---|---|---|---|---|---|---|---|
| 0 | VM1 | task1 | task4 | | | | | |
| 1 | VM2 | task2 | task5 | | | | | |
| 2 | VM3 | | task3 | task1 | task2 | task3 | task4 | task5 |

When it compares RR and PPO-based allocation, two load balancing algorithms, the task allocation comparison table 4 shows how the work is split between the three VM. RR is a method for sequentially distributing duties. The number of tasks assigned to each VM is about equal.VM1 has assigned Tasks 1 and 4, while VM2 has assigned Tasks 2 and 5. VM3 has received Task 3. The learned policy of the model forms the basis for the dynamic assignment of tasks in the PPO-based allocation approach. Nonetheless, there are discrepancies in the allocation of tasks in the output. In this case, it looks like the PPO model may have preferred one VM over another. This could be because of the training environment or the parameters for allocating resources, since all the tasks have been given to VM3.

**Table 5:** Load Distribution Comparison

| **Task** | **VM** | **Round-Robin Load Distribution(%)** | **PPO-based Load Distribution(%)** |
|---|---|---|---|
| 0 | VM1 | 40.0 | 0.0 |
| 1 | VM2 | 40.0 | 0.0 |
| 2 | VM3 | 20.0 | 100.0 |

Under both the RR and PPO-based allocation schemes, the load distribution comparison table 5 shows what proportion of the workload each virtual machine handles. In the RR approach, the workload is split equally among the VMs: VM1 and VM2 each handle 40% of the entire load, while VM3 handles 20% of the whole load, which corresponds to the single task (task 3) that is allocated to it. With all five jobs allocated to it, VM3 handles 100% of

the workload in the PPO-based approach, which results in a skewed load distribution. Here, the PPO model's uneven allocation approach is on display since VM1 and VM2 are not carrying any load. Figure 2 displays the VM load distribution.

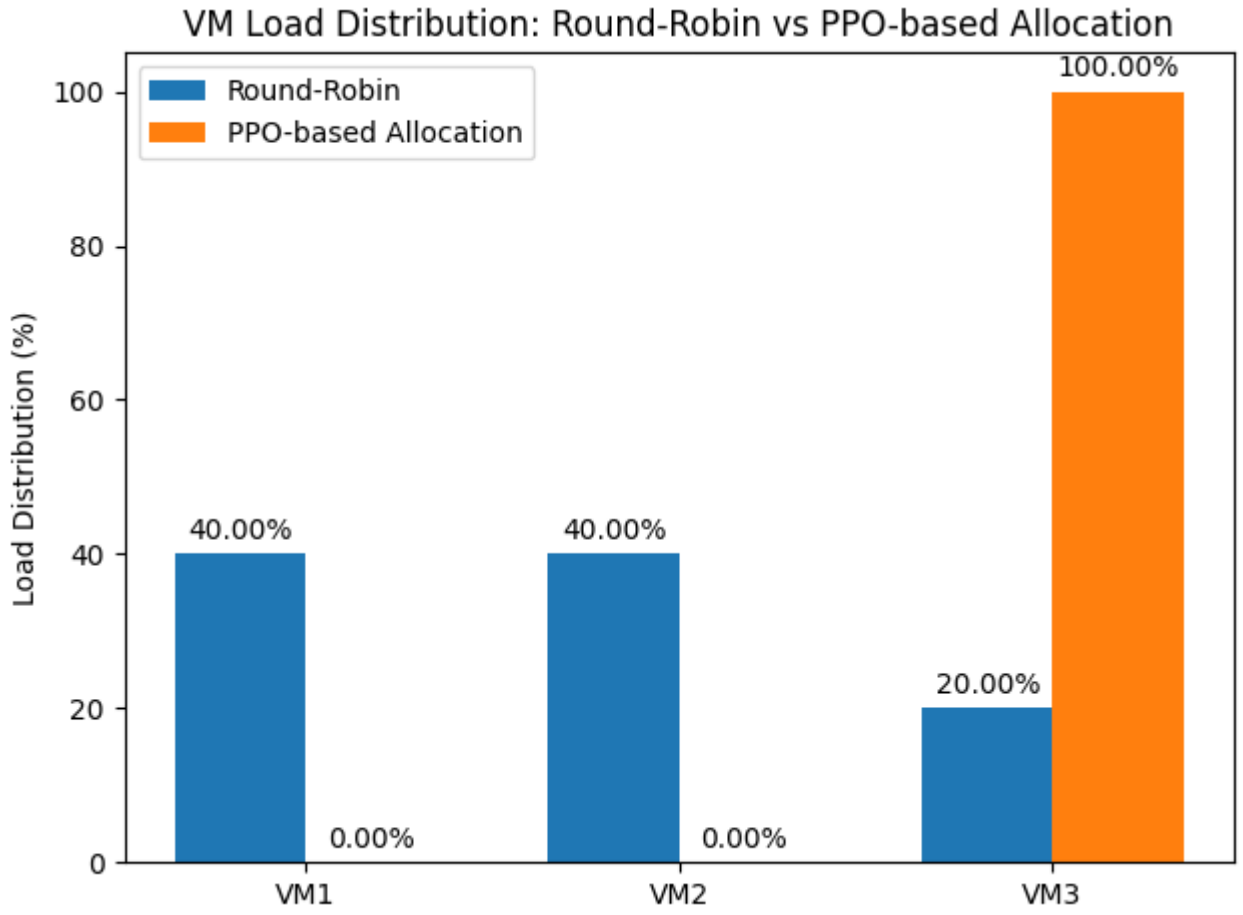


**Figure 2:** VM Load distribution graph

### 4.3 Discussions

The PPO-based allocation method outperforms RR on several crucial parameters due to its greater adaptability and reduced resource usage. First, compared to VM1 and VM2 during round-robin allocation, PPO guarantees that resources are completely used on VM3 by achieving 100% utilization in terms of CPU and memory usage. More jobs may be handled in the same amount of time because of this optimization of resource utilization, which boosts the system's overall efficiency. PPO assigns work based on the load, which makes better use of resources. This is different from Round-Robin, which spreads jobs out evenly but often leaves some VMs idle. Even though VM3 is under more strain, this results in better load distribution. Unlike the static Round-Robin technique, PPO adapts to the current situation, ensuring effective job allocation even when system loads fluctuate. While PPO does increase load imbalance on VM3, it guarantees superior overall performance in environments that prioritize scalability and resource optimization. PPO excels when the goal is to optimize resource utilization at scale, delivering a more flexible and effective allocation in dynamic cloud environments. Finally, load balancing efficiency and task assignment fairness are greater in round-robin.

## 5. Conclusion

For effective resource allocation and task scheduling in cloud environments, this research proposed an allocation algorithm based on PPO. The findings demonstrate a number of major benefits of the PPO-based technique over the more conventional RR allocation method. The main advantage of PPO is its adaptive resource utilization. It takes

real-time system load into account and assigns tasks dynamically, resulting in improved utilization of resources (100 percent CPU and memory usage on a single VM). In environments where getting the most out of limited resources is critical, this is especially beneficial. The PPO algorithm excels in terms of scalability. It skillfully handles workload changes by adjusting task allocation depending on the current status of the system. In contrast to Round-Robin's static task allocation, which often results in unused resources and idle virtual machines, this adaptive flexibility can quickly adjust to changing conditions. The choice to use PPO is justified by its positive effect on task completion speed and resource optimization, even if it may unbalance the load distribution between virtual machines. Overall, PPO improves system efficiency, fairness in task allocation, and load balancing, which makes it a better fit for resource-intensive cloud environments on a big scale.

Some things that could be studied further are making the PPO algorithm work better in large, multi-cloud environments and fixing the uneven load by using multi-agent systems or task relocation algorithms. Further investigation of how the PPO framework may be improved by including energy-efficient strategies might lead to even greater improvements in performance, with reduced energy consumption and sustained high task throughput. For even better performance across a wide range of workloads, further optimization can include tailoring the PPO hyperparameters to the unique specs of the cloud environment.

## REFERENCES


[1] Gures, Emre, Ibraheem Shayea, Mustafa Ergen, Marwan Hadri Azmi, and Ayman A. El-Saleh. "Machine learning-based load balancing algorithms in future heterogeneous networks: A survey." IEEE Access 10 (2022): 37689-37717.

[2] Zhou, Yue, and Chunyou Liu. "Server Load Balancing Scheme Based on Weighted-Round Robin Combined with Least Connections Algorithm in SDN." In 2024 7th International Conference on Advanced Algorithms and Control Engineering (ICAACE), pp. 1008-1012. IEEE, 2024.

[3] Jain, Sushil Kumar. "HDRLGA: Hybrid deep reinforcement learning and genetic algorithm task scheduling approach in cloud computing." Library of Progress-Library Science, Information Technology & Computer 44, no. 3 (2024).

[4] Tache, Maria Daniela, Ovidiu Păscuţoiu, and Eugen Borcoci. "Optimization algorithms in SDN: Routing, load balancing, and delay optimization." Applied Sciences 14, no. 14 (2024): 5967.

[5] Chauhan, Shubham, Shivangam Soni, Abhishek Kumar, Simran Kaur, Ruchika Sharma, Priyanka Kalsi, Riya Chauhan, and Abhishek Birla. "Load Balancing Algorithms for Cloud Computing Performance: A Review." In International Conference on Cognitive Computing and Cyber Physical Systems, pp. 159-176. Singapore: Springer Nature Singapore, 2023.

[6] Priya, Anjali, Ipsit Chandra, Debdatta Pal, and Chinmay Tiwari. "Balancing the Load: An Evolution Story of Load Balancing Algorithm." Smart Internet of Things 1, no. 1 (2024): 54-66.

[7] Moharamkhani, Elaheh, Reyhaneh Babaei Garmaroodi, Mehdi Darbandi, Arezu Selyari, Salim EI khediri, and Mohammad Shokouhifar. "Classification of Load Balancing Optimization Algorithms in Cloud Computing: A Survey Based on Methodology." Wireless Personal Communications 136, no. 4 (2024): 2069-2103.

[8] Prity, Farida Siddiqi, and Md Maruf Hossain. "A comprehensive examination of load balancing algorithms in cloud environments: a systematic literature review, comparative analysis, taxonomy, open challenges, and future trends." Iran Journal of Computer Science (2024): 1-36.

[9] Prity, Farida Siddiqi, and Md Maruf Hossain. "A comprehensive examination of load balancing algorithms in cloud environments: a systematic literature review, comparative analysis, taxonomy, open challenges, and future trends." Iran Journal of Computer Science (2024): 1-36.

[10] Toofani, Abhinav Kumar, Ashish Kumar, Abhijeet Kumar Giri, Abhisht Verma, and Nitish Kumar. "Energy Efficiency and Load Balancing Algorithm for Cloud Environment." Smart City Insights 1, no. 1 (2024): 7-12.

[11] Zhanuzak, Raiymbek, Mohammed Alaa Ala'anzy, Mohamed Othman, and Abdulmohsen Algarni. "Optimising Cloud Computing Performance with an Enhanced Dynamic Load Balancing Algorithm for Superior Task Allocation." IEEE Access (2024).

[12] Hayyolalam, Vahideh, and Öznur Özkasap. "CBWO: A Novel Multi-objective Load Balancing Technique for Cloud Computing." Future Generation Computer Systems 164 (2025): 107561.

[13] Rajawat, Anand Singh, S. B. Goyal, Manoj Kumar, and Varun Malik. "Adaptive resource allocation and optimization in cloud environments: Leveraging machine learning for efficient computing." In Applied Data Science and Smart Systems, pp. 499-508. CRC Press, 2025.

[14] Deng, Miaolei, Umer Nauman, and Yuhong Zhang. "NS-OWACC: nature-inspired strategies for optimizing workload allocation in cloud computing." Computing 107, no. 1 (2025): 1-52.

[15] Rawat, Pradeep Singh, and Prateek Kumar Soni. "Resource Management in Cloud Using Nature-Inspired Algorithms." In Advanced Computing Techniques for Optimization in Cloud, pp. 38-64. Chapman and Hall/CRC, 2025.

[16] Nambi, S., and P. Thanapal. "Emo–Ts: An Enhanced Multi-Objective Optimization Algorithm for Energy-Efficient Task Scheduling In Cloud Data Centers." IEEE Access (2025).