STATISTICS, OPTIMIZATION AND INFORMATION COMPUTING
Stat., Optim. Inf. Comput., Vol. **x**, Month 202x, pp 0–12.
Published online in International Academic Press (www.IAPress.org)

Efficient Cloud Task Scheduling Framework Based on M/M/N Queueing Optimization

Zaid Sabeeh [1,*], E. I. Elsedimy [2,1], Ahmad Hailan [3], Naba Abd [4]

[1] *Department of Cybersecurity, College of Sciences, Al-Esraa University, Iraq*
[2] *Department of IT Management, Faculty of Management Technology and IS, Port Said University, Egypt*
[3] *College of Computer Science and Mathematics, Thi Qar University, Iraq*
[4] *State Commission on Survey, Ministry of Water Resources, Iraq*

Abstract Task scheduling in the cloud data center is an optimization problem that aims to improve the resource utilization and hence reduces the response time. It faces challenges in assigning tasks to a suitable virtual machine such as throughput, waiting time and degree of imbalance. This paper presents a multi-objective algorithm for task scheduling problem based a novel M/M/N optimization model. The goal is to identify the optimal task scheduling strategy that 1) decreases response time by minimizing the volume of pending requests; 2) mitigates performance degradation by maintaining the service level agreement (SLA) at the desired standard; and 3) enhances load balancing by ensuring servers operate at optimal utilization levels. The results show that the proposed algorithm has better results in response time, throughput and load imbalance when compared to other existing algorithms.

Keywords Cloud Computing, Task Scheduling, M/M/N model, Virtual Machine, Workload distribution.

DOI: 10.19139/soic-2310-5070-3969

1. Introduction

Cloud computing [1] is of paramount importance in IT industry. It can deliver infrastructures, platforms and applications as hosted services to large numbers of users in the pay-as-you-go model. To meet the high-performance expectations of cloud users in a cost-effective manner, cloud service providers need to address several aspects, such as reducing energy consumption to gain more profit and ensuring the Quality of Service (QoS) delivered to the cloud users, in addition providers try to keep the server's utilization in an optimal utilization level. Virtualization technologies such as OpenStack and VMware, make it possible for cloud provider to place multiple Virtual Machines (VMs) into a single physical machine to maximize provider's revenue. With the support of virtualization, the resources of a single server such as the amounts of CPUs, main memory, storage and network bandwidth are partitioned into multiple execution environments for several VMs. Task scheduling has an important role in cloud environment visualization, especially in Infrastructure as a Service (IaaS) model. It maps a set of requests to a set of VMs. A good task scheduling solution should satisfy cloud providers' requirement to reduce power consumption and SLA violations.

The steps of task scheduling operation can be spilt into two main parts (1) sorting all VMs in decreasing order according to its current CPU utilization; and (2) mapping each VM into the server that have enough resources and provides the least energy consumption due to this mapping. Calculating all possible VM-server mappings in the large-scale data center and finding the optimal VM placement is not feasible, since the complexity will

*Correspondence to: Zaid Sabeeh (Email: zaid.ali@esraa.edu.iq). Department of Cybersecurity, College of Sciences, Al-Esraa University, Andlus Square, Baghdad, Iraq

ISSN 2310-5070 (online) ISSN 2311-004X (print)

grow quickly with the number of VMs and servers. Therefore, the development of a high efficient VM placement approach that taken in to account the requirements of both cloud provider and consumer, has become a critical issue in cloud computing environment and still a challenge for researchers. Whereas [2] suggested a different approach for enhance task scheduling by following the principles of Game theory to address the load balancing and makespan challenges especially considering the waiting time and task processing time. Their results indicated better rates of resolving conflicting objectives and allocates resources efficiently. Likewise, other researchers continue to focus on the task completion time, [3] explored a Heuristic-Guided Butterfly Swarm Optimization (BSO) algorithm dealing with completion time of the tasking requests. They attempted to better the quality of convergence and scheduling. Simulations with 40-500 tasks indicated that up to 33.5% faster execution and overall efficiency in fog-cloud environments with minimal cost overhead. Another angle to this problem is defining metric performance which had been the focus of Ref. [4] who studied the workflow scheduling while considering the motivation and application scenarios. Their models discussed the metric performance, followed by a taxonomy of existing scheduling strategies categorized by research issues, optimization objectives, and techniques. Additionally, [5] conducted a comprehensive study on the vital role of devising agile scheduler by exploring researcher attempts in cloud workload balancing and task scheduling They also indicated the need for new methods to improve feasibility and effectiveness scheduler the cloud environment.

1.1. Study's Motivation and Contributions

This study is motivated by the need to find solutions for the scheduling of multiple tasks in cloud ecosystem. Thus, the aim is to introduce an agile model for task scheduling for objective of extensile provisioning of cloud services in while ensuring less response time, less resource allocation time and better experience for the requester of the cloud services. In this study, we define tasks scheduling problem as a multi-objective optimization problem. The main contributions of this study are summarized as follows: (1) present a multi-objective optimization model to satisfy the requirements of both cloud providers and consumer; (2) build three models based on the previous work and experiments to estimate the total violation in SLA, load imbalance degree, and the response time; (3) Finally, we have design simulation experiments in order to verify the effectiveness and efficiency of the proposed algorithm. Simulation results show that the proposed algorithm has better results in throughput, violation in SLA and degree of load imbalance when compared to other existing algorithms.

Effective resource allocation and task scheduling are issues of prime importance in cloud computing due to increasing user's expectations. Besides, there is a requirement of the fulfilment of even completely opposite goals, such as minimization of response time and compliance with SLAs. Traditional approaches are mostly based on the single metric measures and Because of this they cannot adequately handle dynamic scenarios. This is the time when a task scheduling setup based on multi-objective optimization can be a great contributor to the improvement of quality of services, the reduction of SLA breaches, and the optimization of resource utilization.

Besides, even though M/M/N queueing models have been massively used for the performance analysis of cloud systems, very few research works have used these models beyond mere performance evaluation to adaptive scheduling decisions making. So, there are still no scheduling structures that use queueing-theory analysis and multi-objective optimization at the same time to be able to minimize response time, decrease load imbalance, and increase resource utilization in dynamic cloud environments, based on the recent literature.

Effective resource allocation and task scheduling are issues of prime importance in cloud computing due to increasing user's expectations. Besides, there is a requirement of the fulfilment of even completely opposite goals, such as minimization of response time and compliance with SLAs. Traditional approaches are mostly based on the single metric measures and Because of this they cannot adequately handle dynamic scenarios. This is the time when a task scheduling setup based on multi-objective optimization can be a great contributor to the improvement of quality of services, the reduction of SLA breaches, and the optimization of resource utilization.

Besides, even though M/M/N queueing models have been massively used for the performance analysis of cloud systems, very few research works have used these models beyond mere performance evaluation to adaptive scheduling decisions making. So, there are still no scheduling structures that use queueing-theory analysis and multi-objective optimization at the same time to be able to minimize response time, decrease load imbalance, and increase resource utilization in dynamic cloud environments, based on the recent literature.

This paper is organized as follows. Section two discusses previous research attempts that suggested task-scheduling methods in the cloud environment. Section Three explains this present the proposed framework. Section Four views the results of implementing the methodology. Section Five contains a summarized discussion on this study and suggests several future research directions for those interesting in pursuing the study in the field of cloud task scheduling. A list of the references followed in this study which have been excerpted from reputable academic research resources in the field of task scheduling of cloud services.

2. Related Works

In recent years, many researches have been conducted to address the problem of VM placement in cloud data center by using diverse approaches. Currently, the majority of existing researches concentrates on a single target and rarely optimizes the VM placement problem taken into account multiple targets in order to obtain the optimal placement. Also, most of attempts have investigated the impact of task scheduling on response time and resource utilization while the issues related with load balancing and the violations in SLA due to place task on the target VM are not discussed. For instance, [6] introduced a scheduling algorithm based on Mixed Integer Linear Programming (MILP) with the goal of improve task allocation across cloud layers. reduces energy consumption, execution time, and failure rates. The proposed scheduler is said to be of great value in achieving 9.63% increase in task throughput and consume less power by 18.20%, and decrease execution times by 9.35%, and lowers failure probabilities by 11.50% across all layers of the distributed cloud system. Alternatively, a group of researchers chose to focus on reducing the cost [7], they presented a heuristic task scheduling algorithm to improve the makespan-cost-reliability (TSO-MCR) using Pareto dominance, and crowding distance approaches. The simulation of twelve scenarios has resulted in 4.24% improvement in terms of makespan, total monetary cost, reliability, and the final score function incorporating all weighted objectives. Whereas [8] explained a strategy called “Efficient Multiverse Electro Search Optimization” to enhance the scheduling behavior. Their technique adapted an exploration, exploitation, and local search find the best solution among several potential alternatives. Their results claimed reducing 6% of the operational costs, and meeting deadlines 8% shorter time. Putting the customer as the focal point in their research [9], they utilized hyper-heuristic algorithm to suggest customer-oriented multi-task scheduling model that prioritize cloud service availability. Experiments against five baselines for moderate-scale and large-scale instances indicated enhanced solution quality by 31.60% and decreasing time by 46.35%. Furthermore, [10] introduced hybrid method that combines two popular algorithms, Grey Wolf Optimizer and Particle Swarm Optimization using key performance indicators such as make-span, throughput, and load balancing. The results indicated 15% increase in the level of make-span, increased throughput by 10% and balanced placement of tasks in the created VMs. [11] suggested a framework for task scheduling and suitable allocation of computational by using nodes within a selected network topology. The pricing scheme in their framework depended on the optimal distribution of tasks. They claimed their model excelled in scheduling of workload both in heterogeneous and homogeneous scenarios. [12] proposed an improved whale migration algorithm which was verified using CEC2022 test problem set and was then tested for cloud task scheduling problems of various scales. Their results show that the proposed algorithm can decrease the overall cost of task distribution by about 9%.

Another group of studies focused on a different aspect of the scheduling process. They focused on the time required to respond to the cloud service requests. For example, [13] proposed a systematic step to manage the task scheduling required for providing services in the cloud architecture. They introduced a task scheduler that adapts Reinforced Learning for the purpose of decreasing the time needed for task completion. They claimed that 271% enhancement in task scheduling was achieved when implemented in online cloud settings and scenarios. A similar aspect of learning was also proposed an intelligent learning-based cloud task scheduling (ILbCTS) algorithm [14]. The suggested algorithm dynamically scales up and down based on changing states and the available rewards. The empirical studies with job set of 1000, 5000 and 10,000 show that the ILbCTS algorithm outperform other algorithms when it comes to execution time, energy conservation and success rate of task scheduling. Additionally, a deep learning technique was adapted to tackle the workload balancing and scheduling problem [15]. Random generated workload is used for simulation then HPC2N and NASA workload are used to measure performance.

The results showed reduction in both makespan and energy consumption. Whereas [16] suggested an improved artificial lemming algorithm then conducted experiments are carried out through global optimization test problems. Their results indicated enhanced accuracy and speed of the solution and verified the feasibility and effectiveness of their method when applied to real case studies in the cloud architecture. [17] suggested task scheduling method that follow "meta reinforcement learning" Through comparing the performance of decreasing makespan in several environments with different configurations. They stated that they found a better adaptive method for a quick and more efficient task scheduling. the experimental results showed that their proposed technique can maintain a high utilization after a few steps of gradient update. [18] proposed a "Q-learning-based Multi-Task Scheduling Framework (QMTSF)". This framework has two layers the first dealing with elastic allocation of servers and instances while the second follows an improved version of the Q-learning algorithm The experimental results showed improvement in the scheduling framework when compared to other previous methods in terms of the makespan and task handling time. Another approach utilized A Priority-Based Heuristic Approach which illustrated in a step-by-step manner with an appropriate number of tasks [19]. The performance of the proposed model is compared in terms of overall waiting time and CPU time against some existing techniques like BATS, IDEA, and BATS+BAR to determine the efficacy of our proposed algorithms. Table 1 presents a summary of key recent studies on methods suggested for task scheduling of workload in cloud ecosystem and their key results.

The synthesis of the recent literature showed a set of methods that did not follow the aforementioned two themes. Workflow management through the use of algorithms was reported in the researchers' attention to solve the latency in task scheduling. For example, subjective taxonomy on the task scheduling schemes as guideline for scheduler improvement was suggested for the scheduling purposes [20]. Whereas, [21] applied Robust Security Approach and Optimized Hybrid for efficient task scheduling. Reducing energy consumption and resource allocation time was the focus of several studies [22, 23, 24]. On the other hand, the reducing of execution time and cost was the main concern that urged another group of studies to improve the overall task scheduling process [25, 26, 27, 28, 29, 30, 31]. For instance, [32] suggested improved particle swarm optimization algorithm for task scheduling in cloud computing. Whereas, [33] proposed a third-generation genetic algorithm NSGAIII for task scheduling in cloud computing. [34] attempted to deploy AI solutions for energy-efficient task scheduling for cloud–fog computing environments. [35] suggested hybrid optimization algorithm for creating cloud service scheduler. [36] Adaptive Optimal Deep Learning for Task Scheduling in Cloud Computing. [37] adapted Dynamic Jellyfish Search Algorithm Based on Simulated Annealing and Disruption Operators for Global Optimization with Applications to Cloud Task Scheduling. [38] came up with a new Hybrid Weighted Ant Colony Optimization Algorithm for Task Scheduling in Cloud Computing. [39] proposed Heuristic initialization task scheduling algorithm in cloud computing. [40] introduced a pair-based task scheduling algorithm for cloud computing environment. Based on the aforementioned studies and research directions, this study proposes a novel framework for adequate task scheduling in the cloud ecosystem.

Despite a lot of changes in the ways to schedule cloud tasks and placing VMs still the researches have not been able to solve every issue. Most of the researches are focused on one main objective - cutting down the energy consumption, improving the running time, lowering the costs, or increasing the throughput. Even the ones that have multi-objective optimization have gone for only a few performance metrics and don't give a full-fledged system that does response time optimization, load balancing, SLA compliance, resource utilization as well as energy efficiency at the same time.

Also, many of the algorithms put forward are heavily based on heuristic and metaheuristic optimization techniques like Particle Swarm Optimization (PSO), Whale Optimization Algorithm (WOA), Grey Wolf Optimizer (GWO), Ant Colony Optimization (ACO), and Genetic Algorithms (GA). While these methods produce really good results in simulation settings, they usually have high computational complexity, long convergence time, and limited scalability when used in large-scale dynamic cloud settings.Besides, many scheduling methods based on learning like reinforcement learning and deep learning, need massive amounts of training data and computing resources. They are also highly reliant on training data quality and the surrounding conditions. That might affect their adaptability factors to real-time cloud systems that have continuously changing loads.

One more drawback noticed in earlier research is that their evaluations are mostly based on simulations using artificial workloads or benchmark data sets. Real-world cloud environments are dynamic and unpredictable, and

simulation is not always able to represent this accurately. This is why the practical usefulness and robustness of many methods proposed are still not thoroughly verified. Besides, most of the published works measure only a restricted set of performance indicators like makespan, execution time, or cost, and ignore other essential Quality of Service (QoS) metrics like SLA violation rate, fault tolerance, fairness in resource allocation, and system stability. Lastly, the research done so far doesn't have a consolidated setup that could efficiently tackle multi-objective VM placement and intelligent task scheduling in changing cloud environments. This raises the issue that there needs to be a creation of an adaptive and integrated scheduling system that supports workload distribution balancing, short response times, fewer SLA violations, efficient use of resources, and scalability in heterogeneous cloud environments.

Table 1. Synthesis of literature on Recent Methods for Task-scheduling in the cloud.

Author, Year	**Methodology**	**Improvements**
Saberikia et al. (2025) [6]	Mixed Integer Linear Programming (MILP)	Increased Throughput and Reduced power consumption
Boroumand et al. (2025) [7]	Heuristic task scheduling algorithm to improve the makespan-cost-reliability (TSO-MCR) using Pareto dominance.	4.24% improvement in terms of makespan, total monetary cost, reliability.
Ravi and Pillai (2025) [8]	Efficient Multiverse Electro Search Optimization	Reducing 6% of the operational costs, and meeting deadlines 8% shorter time
Mengjiao et al. (2025) [9]	Hyper-heuristic algorithm to suggest customer-oriented multi-task scheduling model	indicated enhanced solution quality by 31.60% and decreasing time by 46.35%.
Prasad et al. (2025) [10]	Combination of two popular algorithms, Grey Wolf Optimizer and Particle Swarm Optimization.	15% increase in the level of makespan, increased throughput by 10% and balanced placement of tasks in the created VMs
Ma et al. (2025) [11]	Set of nodes within a selected network topology for pricing scheme for optimal distribution of tasks.	Enhanced scheduling of workload both in heterogeneous and homogeneous scenarios
Lu et al. (2025) [12]	Whale migration algorithm which was verified using CEC2022 test problem set	decrease the overall cost of task distribution by about 9%.
Shen et al. (2025) [13]	Reinforced Learning	271% enhancement in task scheduling was achieved when implemented in online cloud settings and scenarios
Ahmed and Kavitha (2026) [14]	Intelligent learning-based cloud task scheduling (ILbCTS) algorithm	Their algorithm claims to have outperformed other algorithms in execution time, energy conservation and

3. The Proposed Framework

In cloud computing, there are a lot of users who access the service. The proposed framework model presents in figure 1. This framework consists of cloud architecture which can be a cloud service provider center. The service center is a single point of access for all kinds of end-users all over the world. The service center is a collection of service resources which is provided by the provider to host all applications for users. Each user can apply to use the service according to the different kinds of requesting and pays some money to the provider of the service. The focus is only on the works that balances the load among VMs and kept application performance according to SLAs while reducing the response time of end-users' requests. Here, the proposed framework uses a cloud broker for

balancing load among VMs as shown in figure 1. The broker handles a large number of requests, with different requirement, from different end-users, and distributes it over available VMs. Generally, our proposed framework

that outlined in figure 1 is described as follows. The Cloud Broker is responsible for implementing a balancing algorithm and mapping the new request to available VMs. Also, it is applying the changes in available number of VMs to index table of the load balancer. Moreover, it is periodically adjusting the balance of incoming requests among different VMs based M/M/N model [26]. The VM migration policy is employed to collect run-time properties of VMs in cloud datacenters, such as queue length and response times based on queue model. Next, the monitored values pass to the optimization algorithm that sort VMs and compute the optimization function.

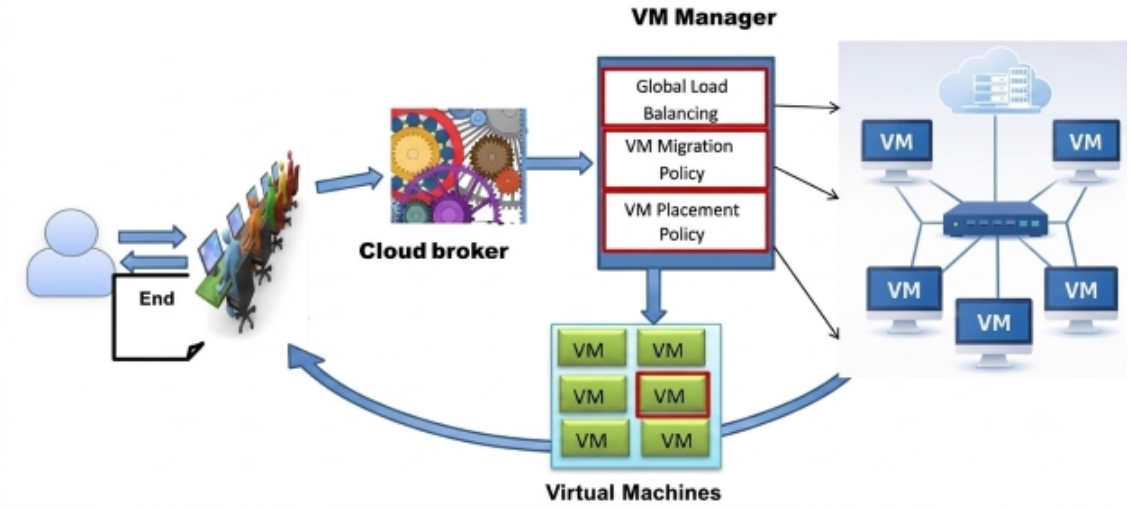

Figure 1. The proposed framework in cloud data center environment

Here, the VM Manager is employed to collect the run-time information of VMs in cloud datacenters, such as queue length and response times. Also, it manages the global load balancing, VM placement and VM migration policies. After that, the VM migration and placement policies are employed to reduce energy consumption, load imbalance, and SAL violation at cloud data centers. In short, in this framework we focus only on the problem of load balancing.

3.1. Dynamic Load Balancing Algorithms

This section presents the key of M/M/N model that plays important role in VM scalability. M/M/N model is applying VM performance monitors to adjust utilization and response time. Arrival requests accepted by cloud broker should be used to validate the VM scalability model based on multiple queues as follows:

$$Lq = f(N) \quad \ldots\ldots\ldots\ldots (1)$$

Where, L(q)is average response time that related to a function of concurrent requests f(N) . This function is called scalability function that is derived from a mathematical M/M/N model, where a relation among response time and the number of requests N is modeled by a non-linear function, such as the one that characterizes a simple M/M/N queuing model as follows:

$$Lq=Lw + Rt = \sum_{M=0}^{n} \frac{C*r}{n(1-u)} \quad \ldots\ldots\ldots\ldots (2)$$

Where, and u are average response times, service times, queue times and utilization of VMs, respectively. In order to obtain optimum result and keep load balance, optimization function is introduced as follows:

$$\min f(n) = Lw + Rt = \sum_{M=0}^{n} \frac{C*r}{n(1-u)} + Rt \quad \ldots\ldots\ldots\ldots (3)$$

We proposed the load Balancing Factor as follows:

Lw = $\sum_{M=0}^{n} \frac{C*r}{n(1-u)}$ . . . . . (4)

Substituting into the main objective:

$minf(n) = \alpha(\sum_{M=0}^{n} \frac{C*r}{n(1-u)}) + \beta R_t$. . . . . (5)

- if $\alpha > \beta$: the system prioritizes load balancing.

- if $\beta > \alpha$: the system prioritizes faster response time.

Where, the input parameter of optimization function is average response times, VM utilizations, and total number of requests waiting in global queue. Minimizing the response time of requests is the objective goal of optimization function. In addition, the load balancing algorithm selects the next request that maps to VM according to the result of the optimization function. Also, the proposed algorithm is derived from optimization function that compute the average response time and VMs utilizations according to equation 3. It accepts a collection of n requests that must be mapping to m VMs. Moreover, the input of proposed algorithm is a finite set of requests and the output mapping requests to VMs basing on optimization function as presented in the formal pseudo code as follows:

The proposed Algorithm

Input:
m: list of VMs
n: list of user's requests
R_t: services times
Output:
Mapping Requests to VMs based on optimization function
For (i = 0; i < n; i++) do
Compute average response time for each request according to Equation 2
Compute VMs utilization according to $U = \lambda_i/\mu_i$
Compute min $f(n) = L_w + R_t$

$$= \left(\sum_{i=0}^{n} \frac{C * r}{n(1-u)} + R_t \right)$$

Sorting VMs according to its f(n) value in ascending order
End
For (j = 0; j < m; j++) do
Mapping the first request to the VMs with minimum response time
End

The goal of proposed algorithm is to reduce response time and increase VM utilization at cloud datacenters by ascending VMs according to optimization function as presented in equation 3. It has three steps; in the first step it computes the total service time based on M/M/N model. The second step is the core of algorithm where we compute the optimization function for each request using VM utilization and average response time. Finally, it is starting by

mapping the first request to the VMs with minimum response time. The computation time of global optimization VM algorithm is O (n x m), where n and m are numbers of requests and VMs, respectively. The proposed model is presented along three phases: in the first phase, initialization of hosts and VMs that are running concurrently on the same physical host; second phase, the performance of proposed model based on queuing systems in a cloud data center is evaluated. Finally, the decision of auto scale VMs is made. Experimental results show that in general VMs utilization, response time, queue length and throughput are important indicators for running VMs. Furthermore, queue length threshold decreases virtual machines response time. The consideration of more general queuing models and other types of VMs to extend this work will be conducted in the future.

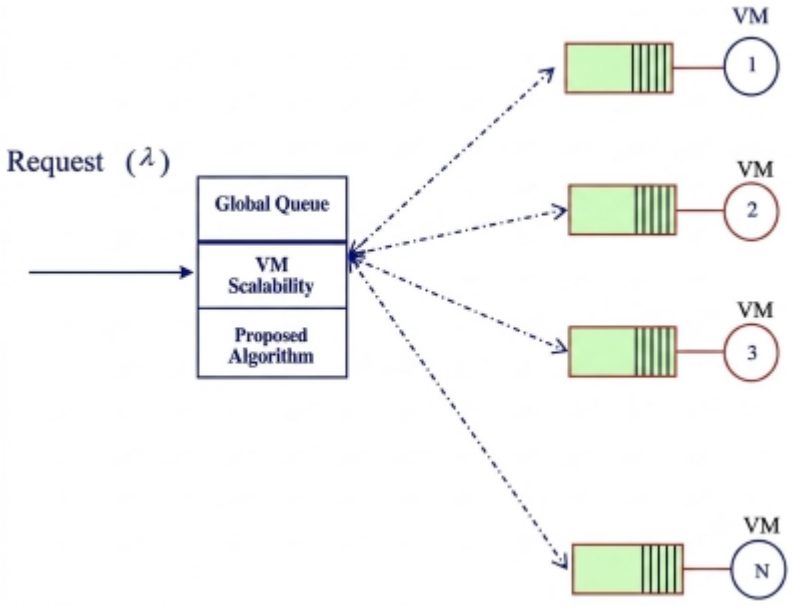

Figure 2. The proposed Algorithm distributes requests based on M/M/N Model

3.2. Parameters Setting of CloudSim

In this section, the proposed algorithm evaluated based on CloudSim toolkit [40] which enables system designer to implement and evaluate complicated cloud system before actual implementation. CloudSim reduces time, risk and cost of implemented complicated cloud systems. We first compare the performance between the proposed algorithm, in which the requests for schedule and computation are allocated optimally by the different load balancing algorithms (i.e., RR, FCFS, proposed algorithm), in which the cloud services provider has four Data Centers (DCs) and each one has different number of VMs independent that accept globally a limited number of requests in concurrent execution. The experiment parameters setting are shown in tables 2 and 3.

Table 2. Data Center Parameters and Their Values.

Data center	RAM	# of processor	BW	Storage	VMM	MIPS (speed)
DC1	10240	4	10000	100000	XEN	10000
DC2	10240	4	10000	100000	XEN	10000
DC3	10240	4	10000	100000	XEN	10000
DC4	10240	4	10000	100000	XEN	10000

Table 3. VM parameters and their values.

Data center	RAM	# of request	Size of request	VM Policy	BW
DC1	1025	12-18	10000	Time Shared	1000
DC2	1025	18-24	10000	Time Shared	1000
DC3	1025	24-30	10000	Time Shared	1000
DC4	1025	30-36	10000	Time Shared	1000

4. Results and Analysis

The simulation results comparison of response time and different arrival rates are shown in figure 3. The response time of three scheduling algorithms are compared to the proposed model as shown in figure 3, where x-axis denoted throughput and y-axis denoted the arrival rates of the VMs. Details in figure 3 demonstrates the throughput of proposed model outperforms the other scheduling algorithms.

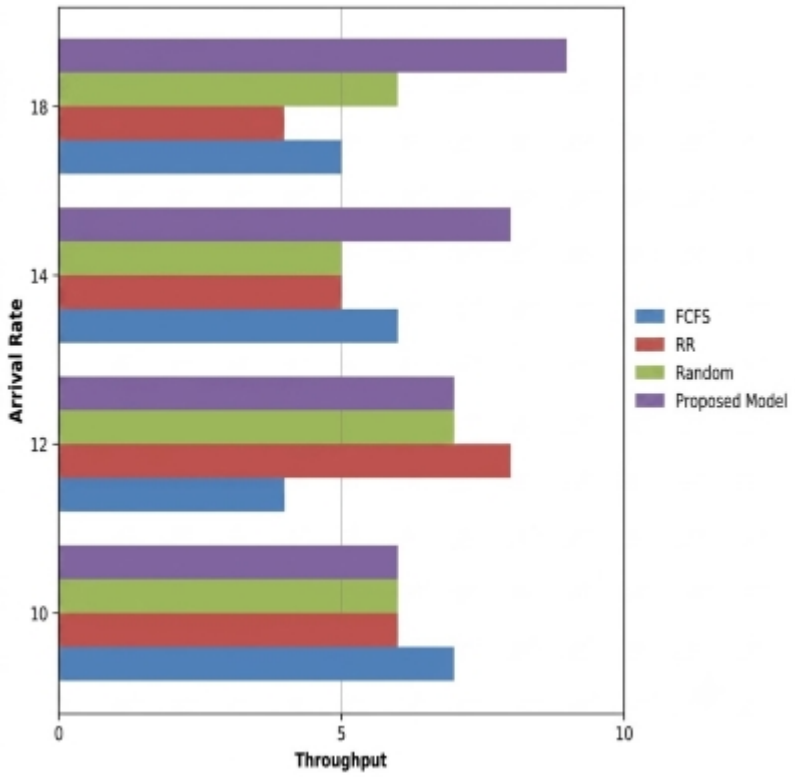

Figure 3. The throughput of RR, FCFS, random and proposed algorithm.

It is because of assigning arrival rates to VMs according to the queue length threshold. Moreover, proposed model assigns arrival rates to the VMs with considering arrival rate length and VMs utilization. So, the total throughput of each VM will be increase. Similarly, it is illustrated in figure 4, proposed model has better resource utilization than FCFS, RR and Random algorithms. Figure 4 demonstrates that when the arrival rates are increased, the VMs utilization increases. In summary, queue length is easy to implement and it performs well in the medium and high workloads. However, when the workload is very low, the queue length threshold performs worse in VMs utilization and throughput.

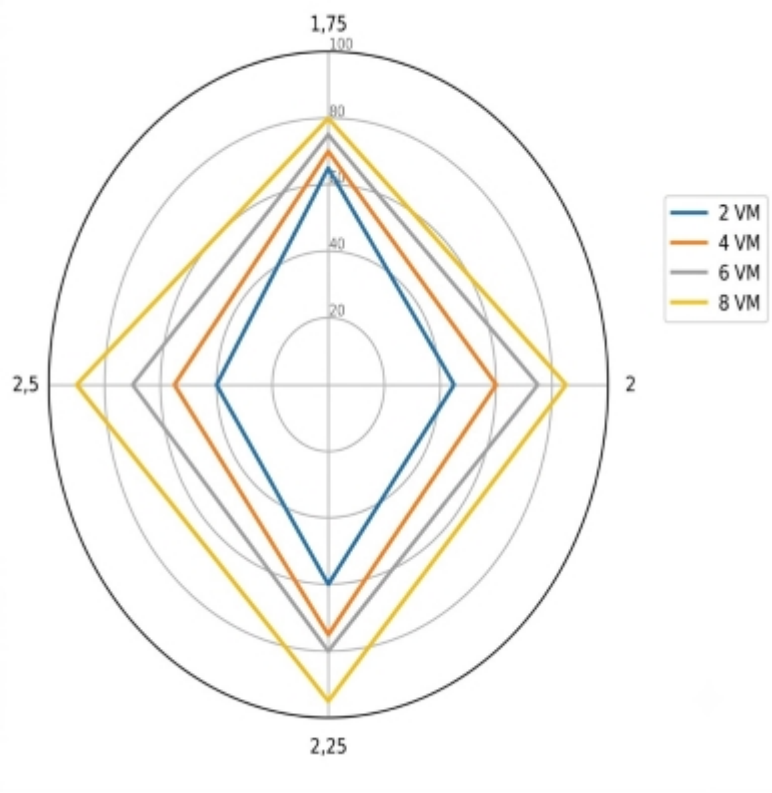

Figure 4. The utilization of proposed algorithm under different VMs sizes

The above table show the effect of our proposed algorithm on response time and total processing time of end-users requests. The impact of the proposed algorithm can be measured using average response time and processing time for different number of VMs and requests numbers under the same constrains. In the first we scale the number of VMs and requests. We can observe the changes of performance metrics as depicted in Table 4. oud datacenters.

Table 4 clearly indicates that the average response time per request under different workloads with utilization of 4 scheduling algorithms, namely FCFS RR PSO, and proposed algorithm. It was noticed from the results that when the number of virtual machines (VMs) was increased, the response time got reduced, which is an indication of the ability of the cloud environment to scale-up. The proposed algorithm points out that the workload with 200-260 VMs and 30-36 requests gets the minimum average response time of 48.43 m/s. For configurations with lesser VMs, the average response time is 50.45 m/s for 80140 VMs and 51.84 m/s for 2080 VMs. These results are a proof that the proposed algorithm is capable of running with low response time despite fluctuating workloads.

The comparison results show that our algorithm keeps topping the performance of PSO, RR, and FCFS irrespective of the workload. The new method stands at the base with the minimum average response time implying that it has a better capacity to manage resources and schedule tasks in an effective manner. PSO comes in as the 2nd best while RR is the third due to its very simple cyclic scheduling approach. FCFS has the highest response times in all the trials thereby showing that it is hardly able to cope with the changing nature of cloud workloads.The duration to wait figures which were given are not entirely clear. Our scheduling method is a multi-objective optimization approach, with performance of the whole system as the main objective, including response time minimizing, load balancing, resource utilization, and SLA compliance. That means, in some cases of

high workload, some tasks might be waiting for a long time since the scheduler is deciding the resource allocation for optimizing global system objectives.

The aforementioned results shows that the response time lessens with an increasing number of available VMs, which corroborates the efficiency of the proposed scheduling technique in cloud resource utilization. For the scenario with the highest workload, the proposed algorithm shows an improved response time of about 8% over FCFS and about 5% over RR. In general, the findings support the idea that the proposed algorithm offers better responsiveness, improved resource utilization, and enhanced quality of service over traditional and heuristic scheduling approaches, Because of this making it a trustworthy solution for cloud task scheduling environments.

Table 4. The Response Time of the proposed algorithm, RR, and FCFS algorithms

Parameters		**The proposed algorithm**			**RR**			**FCFS**		
# of VMs	**# of requests**	**Processing Time (m/s)**			**Processing Time (m/s)**			**Processing Time (m/s)**		
		Avg	**Max**	**Min**	**Avg**	**Max**	**Min**	**Avg**	**Max**	**Min**
20-80	12-18	4.105	6.01	2.20	4.345	6.14	2.55	4.67	6.14	3.20
80-140	18-24	3.645	6.09	1.20	3.991	6.09	2.45	4.145	6.09	2.20
140-200	24-30	3.54	6.07	1.01	3.675	6.07	1.28	3.64	6.08	1.20
200-260	30-36	2.45	6.01	0.89	2.801	6.14	1.20	3.67	6.14	1.20

The Processing Time of proposed algorithm, RR, and FCFS algorithms.

Table 5. The Processing Time of proposed algorithm, RR, and FCFS algorithms.

Parameters		**The proposed algorithm**			**RR**			**FCFS**		
# of VMs	**# of requests**	**Response Time (m/s)**			**Response Time (m/s)**			**Response Time (m/s)**		
		Avg	**Max**	**Min**	**Avg**	**Max**	**Min**	**Avg**	**Max**	**Min**
20-80	12-18	53.84	64.04	43.64	54.425	64.04	44.81	54.54	64.04	45.04
80-140	18-24	53.445	64.04	42.85	54.33	64.04	44.62	53.59	64.04	43.14
140-200	24-30	52.975	64.04	41.91	52.975	64.04	41.91	53.425	64.04	42.81
200-260	30-36	51.425	64.04	40.81	52.425	64.04	40.81	52.975	64.04	41.91

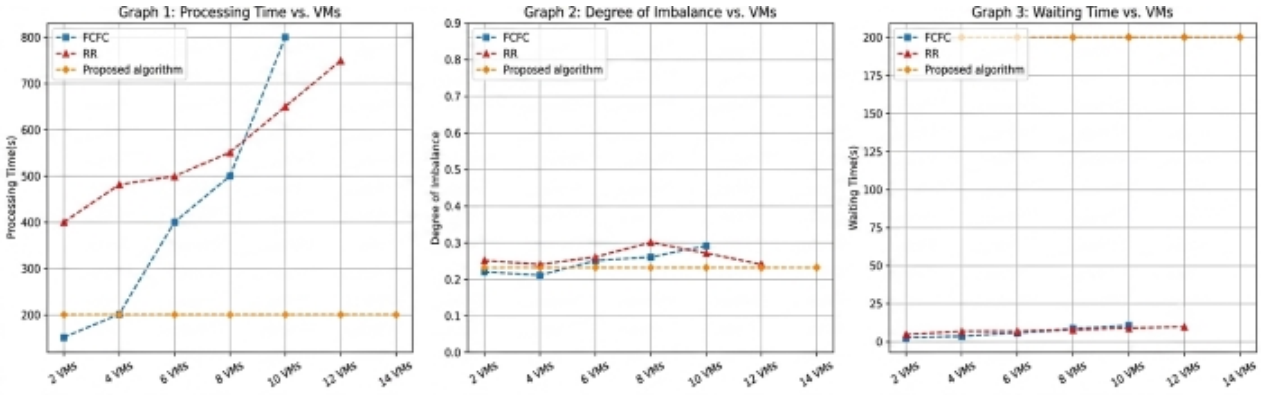

Figure 5. Performance Evaluation of FCFS, Round Robin, and Proposed Scheduling Algorithms Across Varying VMs

In addition, results in table 5 show the increase of VMs leads to stable of processing time. According to the proposed algorithm, 200- 260 of VMs achieves 2.45 m/s processing time. For a smaller number of VMs, 3.64 m/s resulted in 80-140 of VMs and 4.10 m/s resulted in 20-80 of VMs. In generally, the proposed algorithm decreases processing time at cloud system.

Figure 5 evaluates the performance of three scheduling algorithms across 2 to 14 Virtual Machines (VMs), highlighting the Proposed algorithm's consistent processing time of approximately 200s, which contrasts sharply with the poorer scalability of FCFC and RR. FCFC's time increases to 800s at 10 VMs, while RR shows a steady but slower performance. The Proposed algorithm also maintains a low Degree of Imbalance at about 0.23, demonstrating stability in workload distribution, compared to the higher volatility of FCFC and RR.

However, this stability comes at the cost of increased latency, as the Proposed algorithm's waiting time remains high at 200s, unlike FCFC and RR, which have waiting times below 10s. Thus, the Proposed algorithm excels in high-throughput tasks but is less suitable for real-time applications requiring quick responsiveness.

5. Conclusions and Future Works

This paper has introduced optimization load balancing algorithm based on M/M/N model. This algorithm proposed some of contributions to reduce response and processing times of the end-users, requests. Comparing the response and processing times of end-users' requests with different algorithms, results showed that the proposed algorithm outperformed FCFS and RR algorithms in average. From the results, the proposed algorithm can be proving by the truth that the proposed algorithm based on optimization function can reduce response time. The consideration of more general global load balancing algorithm to extend this work will be conducted in the future. Researchers interested in task-scheduling of cloud services could improve on the proposed framework or suggested an alternative to the optimization algorithm to improve throughput and reduce response time.

REFERENCES

1. Alattraqchi, A.A.A., Khalilian, M., Alsalamy, A. et al. *The art of scheduling: ANFIS-GPC synergy for energy-aware cloud optimization*, Journal on Computing, vol. 108, no. 20, 2026
2. Jalali Khalil Abadi, Z., Javidi, M.M., Mansouri, N. et al. *Game theory-based framework for efficient task scheduling in cloud computing* , Cluster Comput 29, 106 (2026).
3. Pinky, and Verma, K. *Heuristic-guided BSO for efficient task scheduling in IoT-driven fog–cloud environment*. Cyber-Physical Systems, 1–24. (2026)
4. Shenghai Li, Wentai Wu, Haotong Zhang, Yongheng Liu, Weiwei Lin, Keqin Li *Revisiting workflow scheduling with the power of edge computing: Taxonomy, review, and open challenges*, Computer Science Review, Volume 60, 2026.
5. González-San-Martín, J. et al. . *A Comprehensive Review of Task Scheduling Problem in Cloud Computing: Recent Advances and Comparative Analysis. In: Castillo, O., Melin, P. (eds) New Horizons for Fuzzy Logic, Neural Networks and Metaheuristics.* , Studies in Computational Intelligence, vol 1149. Springer, Cham. (2024)
6. Mohammadreza Saberikia, Hamed Farbeh, Mahdi Fazeli, *Improving energy efficiency and fault tolerance of mission-critical cloud task scheduling: A mixed-integer linear programming approach*, Sustainable Computing: Informatics and Systems, Volume 45, 2025, 101068, ISSN 2210-5379.
7. Boroumand, A., Hosseini Shirvani, M. and Motameni, H. *A heuristic task scheduling algorithm in cloud computing environment: an overall cost minimization approach*. Cluster Comput 28, 137 (2025).
8. Ravi, R., Pillai, M.J. *Efficient multiverse electro search optimization for multi-cloud task scheduling and resource allocation*. Multimed Tools Appl 84, 24351–24376 (2025).
9. Mengjiao Chen, Jiyuan Xu, Wenyu Zhang, Zhenghui Li, *A new customer-oriented multi-task scheduling model for cloud manufacturing considering available periods of services using an improved hyper-heuristic algorithm*, Expert Systems with Applications, Volume 269,2025,126419, ISSN 0957-4174.
10. Prasad, R., Roy, A., and Kumari, S. *Enhancing Cloud Task Scheduling Using a Hybrid Particle Swarm and Grey Wolf Optimization Approach*. arXiv preprint arXiv:2505.15171. (2025)
11. Q. Ma, Y. Qin, C. Zhu, L. Gao and X. Chen, *Joint Resource Trading and Task Scheduling in Edge-Cloud Computing Networks*, IEEE Transactions on Networking, vol. 33, no. 3, pp. 994-1008, June 2025,
12. Lu, H., Cheng, S., and Zhang, X. *An Improved Whale Migration Algorithm for Global Optimization of Collaborative Symmetric Balanced Learning and Cloud Task Scheduling.* Symmetry, 17(6), 841. (2025)
13. Shen, W., Lin, W., Wu, W. et al. *Reinforcement learning-based task scheduling for heterogeneous computing in end-edge-cloud environment*. Cluster Comput 28, 179 (2025).
14. Ahmed, M. W., and Kavitha, G. *Implementing an intelligent learning-based algorithm for efficient task scheduling in cloud computing environments.* Information Security Journal: A Global Perspective, 35(1), 112–123. (2026)
15. Mangalampalli, S., Karri, G.R., Kumar, M. et al. *DRLBTSA: Deep reinforcement learning based task-scheduling algorithm in cloud computing.* Multimed Tools Appl 83, 8359–8387 (2024).
16. Tan, Y., Wang, J., and Wang, B. *An Improved Bionic Artificial Lemming Algorithm for Global Optimization and Cloud Task-Scheduling Problems.* Biomimetics, 10(9), 572, (2025).

17. Xiu, X., Li, J., Long, Y. et al. *MRLCC: an adaptive cloud task scheduling method based on meta reinforcement learning.* J Cloud Comp 12, 75 (2023)
18. Wang, Y., Dong, S., and Fan, W. (2023). *Task Scheduling Mechanism Based on Reinforcement Learning in Cloud Computing.* Mathematics, 11(15), 3364.
19. S. Lipsa, R. K. Dash, N. Ivković and K. Cengiz, *Task Scheduling in Cloud Computing: A Priority-Based Heuristic Approach*, IEEE Access, vol. 11, pp. 27111-27126, 2023
20. Hosseini Shirvani, M. *A survey study on task scheduling schemes for workflow executions in cloud computing environment: classification and challenges*. J Supercomput 80, 9384–9437 (2024).
21. S. V. A. Kumer, N. Prabakaran, E. Mohan, B. Natarajan, G. Sambasivam and V. B. Tyagi, *Enhancing Cloud Task Scheduling With a Robust Security Approach and Optimized Hybrid POA*, IEEE Access, vol. 11, pp. 122426-122445, 2023
22. Hojjat Emami, *Cloud task scheduling using enhanced sunflower optimization algorithm*, ICT Express, Volume 8, Issue 1, 2022, Pages 97-100
23. Zhixia Zhang, Mengkai Zhao, Hui Wang, Zhihua Cui, Wensheng Zhang, *An efficient interval many-objective evolutionary algorithm for cloud task scheduling problem under uncertainty*, Information Sciences, Volume 583, 2022, Pages 56-72
24. Prity, F.S., Gazi, M.H. and Uddin, K.M.A. *A review of task scheduling in cloud computing based on nature-inspired optimization algorithm.* Cluster Comput 26, 3037–3067 (2023).
25. Ghafari, R., Kabutarkhani, F.H. and Mansouri, N. *Task scheduling algorithms for energy optimization in cloud environment: a comprehensive review.* Cluster Comput 25, 1035–1093 (2022).
26. Hussain, A. A., and Al-Turjman, F. *Hybrid Genetic Algorithm for IOMT-Cloud Task Scheduling.* Wireless Communications and Mobile Computing, 2022(1), 6604286. (2022)
27. Nabi, S., Ahmad, M., Ibrahim, M., and Hamam, H. *AdPSO: Adaptive PSO-Based Task Scheduling Approach for Cloud Computing.* Sensors, 22(3), 920, (2022).
28. Liu, S., Ma, X., Jia, Y., and Liu, Y. *An Energy-Saving Task Scheduling Model via Greedy Strategy under Cloud Environment.* Wireless Communications and Mobile Computing, 2022(1), 8769674. (2022)
29. Abdulgader, D. A., Yousif, A., and Ali, A. *A Discrete Prey–Predator Algorithm for Cloud Task Scheduling.* Applied Sciences, 13(20), 11447. (2023)
30. Saravanan, G., Neelakandan, S., Ezhumalai, P. et al. *Improved wild horse optimization with levy flight algorithm for effective task scheduling in cloud computing.* J Cloud Comp 12, 24, (2023).
31. Xie, N., Li, W., Zhang, J., and Zhang, X. *Research on Cloud Task Scheduling Algorithm with Conflict Constraints Based on Branch-and-Price.* Applied Sciences, 13(13), 7505. (2023)
32. Pirozmand, P., Jalalinejad, H., Hosseinabadi, A.A.R. et al. *An improved particle swarm optimization algorithm for task scheduling in cloud computing.* J Ambient Intell Human Comput 14, 4313–4327 (2023).
33. Imene, L., Sihem, S., Okba, K., and Mohamed, B. *A third generation genetic algorithm NSGAIII for task scheduling in cloud computing.* Journal of King Saud university-computer and information sciences, 34(9), 7515-7529. (2022)
34. Iftikhar, S., Ahmad, M. M. M., Tuli, S., Chowdhury, D., Xu, M., Gill, S. S., and Uhlig, S. *HunterPlus: AI based energy-efficient task scheduling for cloud–fog computing environments.* Internet of Things, 21, 100667. (2023)
35. Khan, M. S. A., and Santhosh, R. *Task scheduling in cloud computing using hybrid optimization algorithm* Soft computing, 26(23), 13069-13079. (2022).
36. Badri, S., Alghazzawi, D. M., Hasan, S. H., Alfayez, F., Hasan, S. H., Rahman, M., and Bhatia, S. *An efficient and secure model using adaptive optimal deep learning for task scheduling in cloud computing.* Electronics, 12(6), 1441. (2023)
37. Attiya, I., Abualigah, L., Alshathri, S., Elsadek, D., and Abd Elaziz, M. *Dynamic jellyfish search algorithm based on simulated annealing and disruption operators for global optimization with applications to cloud task scheduling.* Mathematics, 10(11), 1894. (2022)
38. Chandrashekar, C., Krishnadoss, P., Kedalu Poornachary, V., Ananthakrishnan, B., and Rangasamy, K . *HWACOA scheduler: Hybrid weighted ant colony optimization algorithm for task scheduling in cloud computing.* Applied Sciences, 13(6), 3433.(2023)
39. Alsaidy, S. A., Abbood, A. D., and Sahib, M. A. *Heuristic initialization of PSO task scheduling algorithm in cloud computing.* Journal of King Saud University-Computer and Information Sciences, 34(6), 2370-2382. (2022).
40. Panda, S. K., Nanda, S. S., and Bhoi, S. K. *A pair-based task scheduling algorithm for cloud computing environment.* Journal of King Saud University-Computer and Information Sciences, 34(1), 1434-1445. (2022)