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

Enhanced Cloud Load Balancing With MPSOA- LB: A Multi-Objective PSO Approach for Dynamic Task Allocation and Performance Optimization

Shameer A P[1*], Haseeb V V[2], Minimol V K[3], Reshma P K[4], Aneesh Kumar K[5]

Submitted: 17/11/2022 **Revised:** 28/12/2022 **Accepted:** 12/01/2023

Abstract—Cloud computing is a computing environment that involves the process of accessing of services, which includes the storage environments, various applications and servers through the Internet. Cloud is a large pool of information, software packages, shared resources, storage and enormous applications based on user demands at any point of time. In other words, cloud computing refers to system-oriented software, physical hardware devices, and day-to-day applications delivered to the users through the medium of the internet as services. Cloud resources can be accessed in diverse ways for multiple purposes, making scheduling crucial for providing optimal services to users. With data and resource availability increasing constantly, the need for efficient scheduling algorithms becomes paramount. Effective load-balancing techniques can significantly enhance system performance while reducing costs and energy consumption. Various heuristic algorithms have been proposed to tackle these challenges, with intelligent approaches being widely adopted. This paper portrays the detailed description of the MPSOA-LB scheme propounded for attaining substantial load balancing in a cloud computing setting. The planned system model explores various factors that contribute to the development of a fitness function, which helps evaluate the over-utilization and under-utilization in the MPSOA-LB. This algorithm focuses on efficient load distribution among virtual machines and hosts within cloud environments. The paper also discusses the simulation setup, and the results obtained from implementing the MPSOA-LB under varying conditions, including the quantity of tasks, instruction finishing lengths, and increasing the quantity of virtual machines.

Keywords*: quantity, simulation, techniques, algorithms*

INTRODUCTION

A computing environment known as cloud computing allows users to access services, including servers, storage, and apps, over the Internet. It utilizes the existing resources of various organizations as remote services available for payment. In cloud computing, load balancing entails allocating tasks among several virtual machines within a data center. The workload entering into the cloud computing environment need to be significantly allocated to the resources, such that each share is responsible for sharing an equivalent quantity of loads at any particular moment. The performance of the cloud environment completely varies on the degree to which the resources are equally shared since imbalance in load leads to deterioration in the network efficiency. Further, the system needs to follow a potential load-balancing scheme for facilitating the promotion of resource availability that in turn leads to the increased performance of the cloud computing environment.

Moreover, the balancing load process consists of complex issues for facilitating optimal resource utilization with rapid processing time. The workload associated with each individual VM present in the datacenter is presented based on the cumulative sum of the forecasted or expected computation of time associated with respect to the assigned independent tasks. In particular, load balancing aims in effective resource utilization for maximizing the throughput with reduced response time through the equal sharing of workloads among the servers in the cloud environment. The main objective of load balancing approach aims at enhancing the performance based on workload balancing phenomenon among the virtual machines. It also concentrates towards the achievement of ideal resource consumption, maximizing output, and maximizing latency for preventing overloads.

WHY LOAD BALANCING IS ESSENTIAL

The following are the primary components of load balancing in cloud computing.

i) The load balancing process are considered to be potential in handling any sudden traffic received into the cloud computing environment at any particular point of time.

ii) The load balancing mechanisms are capable enough in handling any amount of traffic burst incoming into environment, since they distribute the load uniformly to

1,2,3Department of Computer Science, NAM College Kallikkandy, Kannur, Kerala, India ,670693, shameerap@gmail.com, haseebvvs@gmail.com, minimoldeepak@gmail.com*
4Department of Computer Science, Mahatma Gandhi College Iritty, Kannur, Kerala, India ,670703, pkreshma@gmail.com
5Department of Statistics, Mahatma Gandhi College Iritty, Kannur, Kerala, India ,670703. aneesh.anek@gmail.com

the available VMs of the data for maximizing the response time.

iii) The load balancing techniques are highly flexible in nature by distributing the workload among the number of servers and network units even if any of the node ends in failure.

iv) The load balancing techniques are highly scalable in handling huge amount of traffic with robustness and fault tolerance capability.

v) They are significant in easily managing the high-end user data traffic with the existence of network devices and servers.

vi) It plays an anchor role in e-commerce, since they help the commercial websites of Flipkart and Amazon to deal with millions of customers every single second of time.

vii)It is helpful in e-commerce activities by managing and distributing workloads during the offers of sale and promotional time.

viii) They are also easily implementable and less expensive as compared to their counterparts since they enable the organizations to work on their clients' applications in a rapid speed for delivering optimal performance at a comparatively lower cost.

PROPOSED MPSOA-LB SYSTEM

The MPSOA-LB scheme focuses on three key capabilities:

1. **Classification of VMs:** It categorizes virtual machines into under-loaded and over-loaded groups to facilitate effective load balancing.
2. **Energy Minimization:** It aims to reduce the overall incurred costs by minimizing energy consumption in the data centre.
3. **Utilization Identification:** This component of the scheme identifies Virtual Machines (VMs) within the data center that are either not being fully utilized or are overloaded

Additionally, includes a method for defining specific metrics that set upper and lower limits, which act as benchmarks for recognizing when VMs are experiencing either excessive or insufficient workloads. These metrics are based on the volume of tasks entering the cloud system.

ALGORITHM OF MPSOA-LB

The MPSOA-LB uses a multi-objective function to manage how tasks are allocated and reassigned among VMs or hosts. This process is guided by primary constraints, which require that the load on VMs should not exceed the upper limit after task assignment. Deadline constraints are contemplated when there is a substantial quantity of available VMs. Furthermore, transferring tasks from VMs that are severely loaded to those with lighter loads is essential, depending on the tasks' deadlines or required completion times. In this approach, VMs with the earliest deadlines are prioritized for tasks that have high completion times. Conversely, for tasks with moderate completion times, VMs with more flexible deadlines are selected. Additionally, VMs are categorized based on their current load into two groups: those that are under-loaded and those that are over-loaded.

Algorithm 1: Steps for Implementing the MPSOA

Step1: The population is initialized with S_j such that value of j satisfies the condition1:
≤j≤n //n is the number of VMs present in the data centre of clouds.

Step 2. Set the speed or velocity $S_{Vel(j)}$ of the particle (VM) with which it could be allocated. However, the speed or velocity $S_{Vel(j)}$ of the particle is initially 0.

Step3. Compute the availability of each particle available in the data centres.

Step 4. Aggregate the identified non-dominated solutions in the repository (VMs that are lightly loaded) for confirming its under-loaded conditions.

Step5. Initiate the generation of hyper cubes.

Step 6. Set the memory corresponding to each particle by aggregating information related to the stored initial positions, in order to identify the best position of the particles identified so far, based on $S_{\text{BEST}}(j) = S_j$

Step7: Calculate and update the velocity of the particle based on

$$S\_\text{Vel}(j) = w \cdot S\_\text{Vel}(j) + \text{rand}_1 \cdot (SBEST(j) - S_j) + \text{rand}_2 \cdot (SREP(j) - S_j)$$

Where the value of inertial weight is set to 0.4, with the random numbers rand1 and rand2 ranging between 0 and 1.

Step 8. Manipulate the updated positions of each particle based on the equation
Sj=Sj+S_Vel(j).

Step 9. The particles that are estimated to be within the search boundaries are maintained in the repository.

Step 10. Again, estimate the fitness of each particle.

Step 11. Employ the operations of mutation over each and every particle.

Step 12. The hypercube and repositories are updated by preventing the worst particles from participating in the allocation of virtual machines (VMs) to incoming tasks.

Step 13. Update the memory of each particle by substituting the previous best position with the newly identified best position achieved by that particle.

Step 14. Terminate the iteration if the maximum number of iterations has been completed.

Step 15. Otherwise, iterate from Step 7 until the termination conditions are satisfied.

In the MPSOA-LB scheme, virtual machines (VMs) in overloaded groups are required to offload their tasks and wait until they can find a suitable VM for real location in the subsequent iteration. Meanwhile, VMs in underloaded groups receive tasks that are pending or require redistribution. This process of removing tasks from overloaded VMs continues until all underloaded VMs are allocated tasks. Potential allocation solutions, represented as particles, are generated randomly to explore possible options for task allocation. The MPSOA-LB scheme leverages Pareto ranking to address multi-objective optimization challenges associated with task allocation based on VM availability, which is assessed in terms of their overloaded or underloaded states. It also maintains a record of non-dominating solutions from previous iterations, tracking the best solutions identified by the particles. As detailed in Algorithm 1, these procedures contribute to the effective distribution of jobs/duties to VMs, considering constraints related to over-allocation and under-allocation, thereby enhancing the "load-balancing process".

SIMULATION SET UP OF THE MPSOA-LB SCHEME

The execution of the MPSOA-LB program was assessed through experiments conducted using CloudSim. CloudSim proved crucial for modelling and simulating various activities, allowing an in-depth study of how varying resource levels affect cloud environments. The evaluation of the MPSOA-LB scheme involved testing across diverse hosts, data centers, and virtual machines to gauge the impact of scalability within the cloud infrastructure. For the simulations, the MPSOA-LB scheme was implemented in an environment comprising 10 data centers, 50 virtual machines, and a range of tasks from 100 to 1000. The tasks had the length of executable instructions between 1000 and 20000 Million Instructions (MI). Table 3.1 also outlines key simulation parameters used in the experimentation of the MPSOA-LB scheme.

Table 3.1: Key factors for the MPSOA-LB Scheme Execution

Category	**Type**	**Setting**
Tasks	#Tasks	100-1000
	Task Length	2000-20000
Data Center	VM-Scheduler	Time-Shared
	#Hosts	02-04
	#DataCenters	10
Virtual Machine (VM)	Cloudlet-Scheduler	Time-Shared
	Bandwidth	500-1200
	Essential Processor Count	01-02
	Processor Speed	4000-8000MIPS
	No. of VMs	50
	Available storage area in each VM	256-2018Mb

ANALYSIS AND FINDINGS OF THE PROPOSED WORK

The following criteria are used to assess the effectiveness of the proposed MPSOA-LB scheme.

1. Mean Response Time: Analysed concerning variations in the amount of work and the part of executable instructions.
2. Performance Comparison with Traditional Algorithms: Comparison of "mean response time, executable instruction length, and mean execution time" between the proposed scheme and conventional algorithms.
3. Mean Response Time, Executable Command Length, and Mean Execution Time are evaluated against algorithms based on swarm intelligence.
4. Investigation of the quantity of migrated work as the number of virtual machines (VMs) increases and the task count remains constant.
5. Task Migration Analysis with Increasing Task Count: Examination of the no. of migrated tasks as the task count grows, with a fixed number of VMs.

Response Time with Diff**erent Numbers of Tasks and Practicable Instruction Measurements**

The importance of the MPSOA-LB scheme is evaluated in this fragment through the mean response time for varying task counts the lengths of executable instructions in a cloud computing environment. Figure 1 plots the "MPSOA-LB scheme", where the average response time is measured for varying task count and instruction length (i.e. in bytes). The mean response time increases from 8.14 seconds to 21.24 seconds when the task count increases from 100 to 1000, while the executable instruction length is 2000 bytes. For instance, when the instruction length is 8000 bytes, increasing the number of tasks leads to the mean response time increasing from 8.96 seconds to 45.48 seconds. As tasks increase, the mean reaction time

raises from 9.12 seconds to 76.42 seconds for an instruction length of 14,000 bytes. Finally, the mean response time increases from 8.12 seconds to 102.14 seconds as the task count increases steadily, while the instruction length is set to 20,000 bytes. The response time increases consistently with varying instruction length because of the threshold parameters used to measure the levels of over utilization and underutilization of the system.

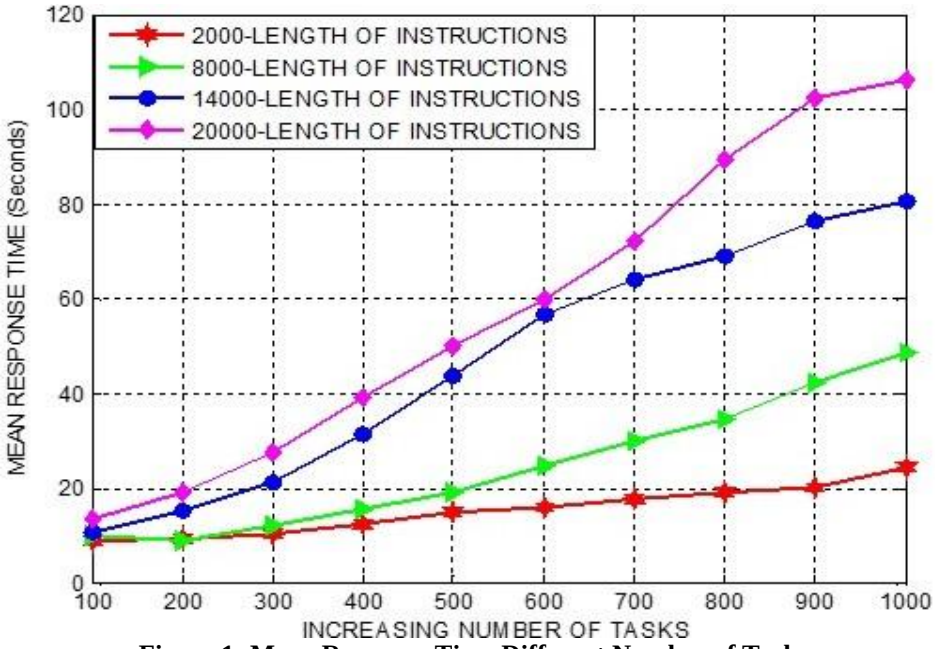

Figure 1: Mean Response Time Different Number of Tasks

Figure 2 shows the variation of average response time of the MPSOA-LB program with the feasible instruction length for different task counts. As the instruction extent increases from 200 to 2000, the mean reaction time increases from 23.42 to 38.76 seconds with 200 tasks. For 400 tasks, the mean response time also rises from 31.21 seconds to 56.12 seconds when the instruction length increases by the same amount. The response time is observed to increase from 36.54 seconds to 64.42 seconds as the executable instruction length varies for 700 tasks. Furthermore, the mean response time rises from 42.32 seconds to 78.18 seconds when the instruction length increases from 200 to 2000 with 1000 tasks. The rise of response time as task count and instruction length vary can mainly be explained by the allocation and deallocation policies, as well as the threshold parameters used in the MPSOA-LB scheme.

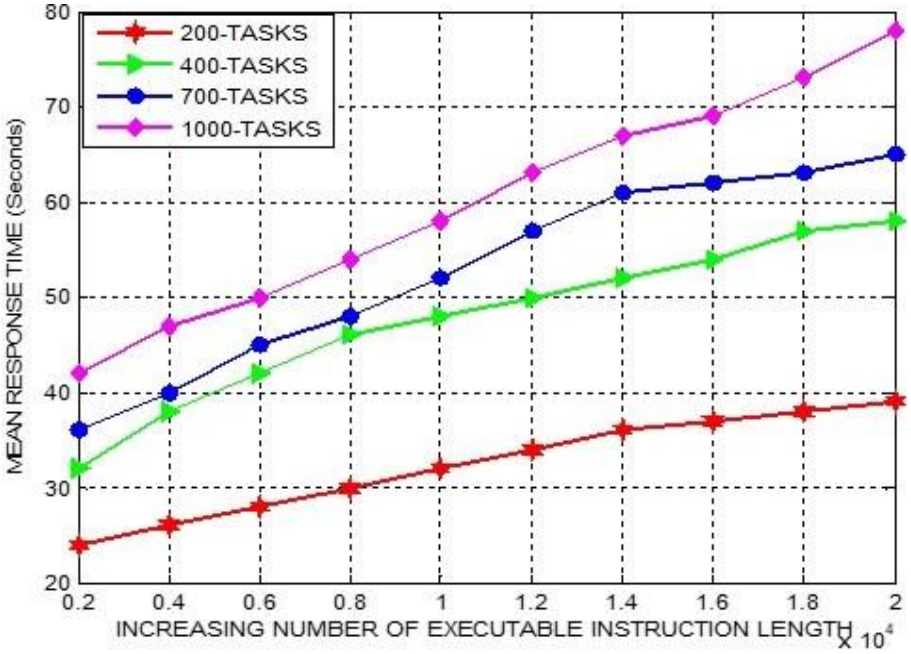

Figure 2: Mean Response Time under Different Executable Instruction Length

Performance Evaluation Over Benchmarked Swarm Intelligent Schemes

This section compares the MPSOA-LB scheme with three swarm intelligence-based approaches: ACO-LB, ABC-LB and PSO-LB. The evaluation is done by changing the quantity of tasks from 100 to 1000 and the executable command length from 2000 to 20,000 for a upper limit value of 0.1. The average response period of the MPSOA-LB system is compared with ACO-LB, ABC-LB, and PSO-LB across different task counts as shown in Figure 3. The results show that the MPSOA-LB scheme reduces the mean response time by 8.12%, 9.84% and 10.21% as compared to ACO-LB, ABC-LB and PSO-LB respectively. The multi-objective optimization function in the task allocation process is the reason for the improvement in response time as the process of assigning tasks to virtual machines (VMs) is more efficient

.

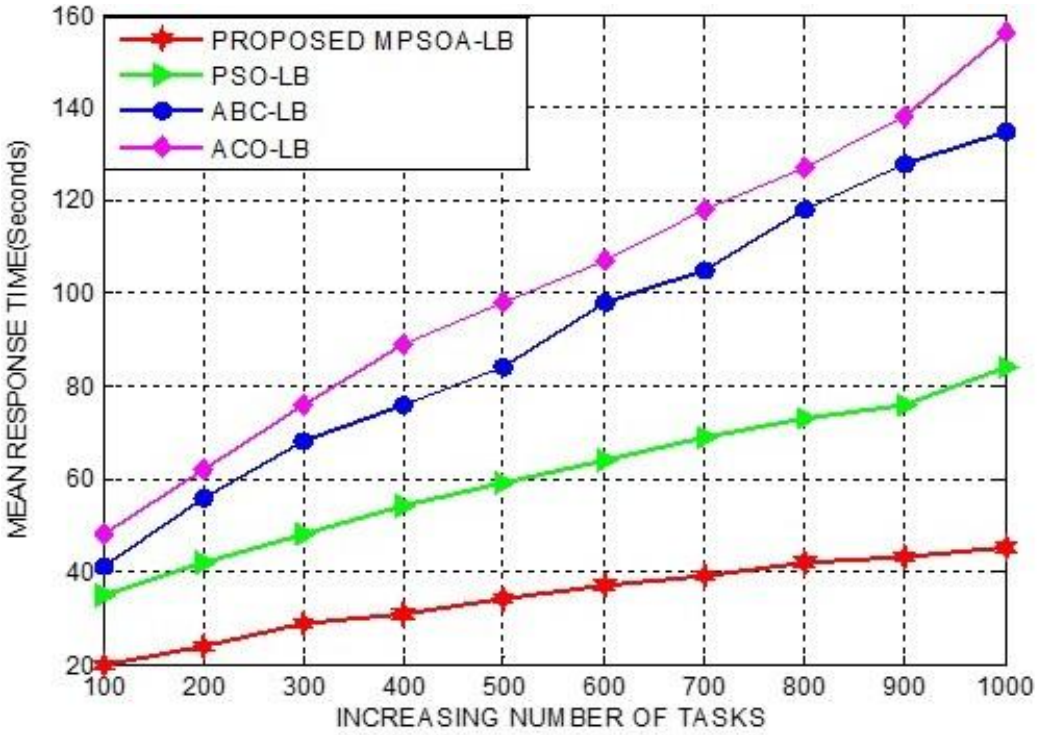

Figure 3: Mean Response Time under Number of Tasks

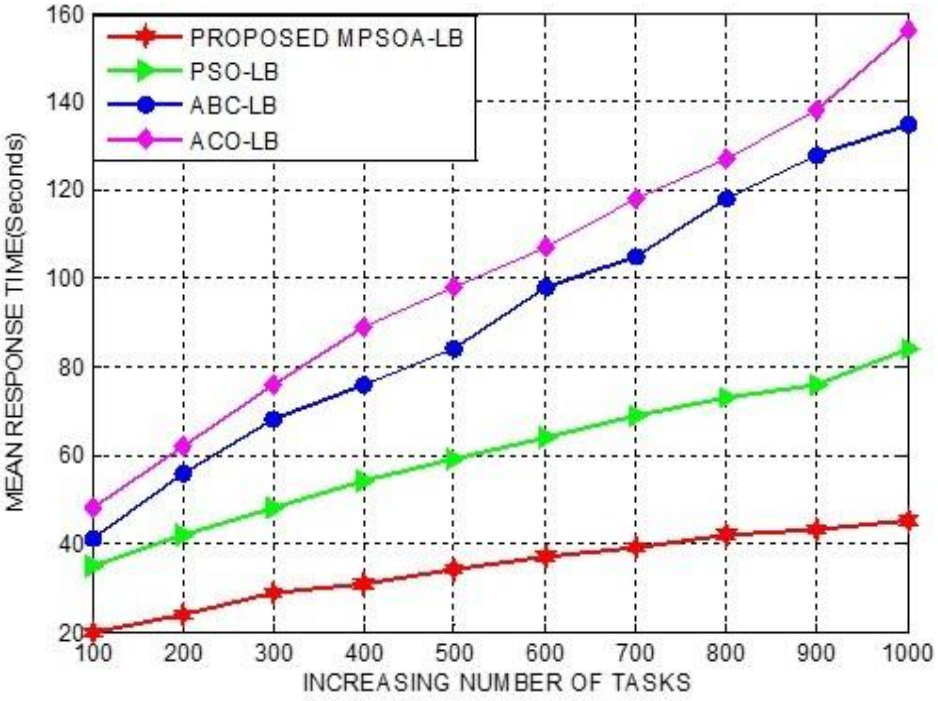

Figure 4: Mean Response Time under Different Executable Instruction Length

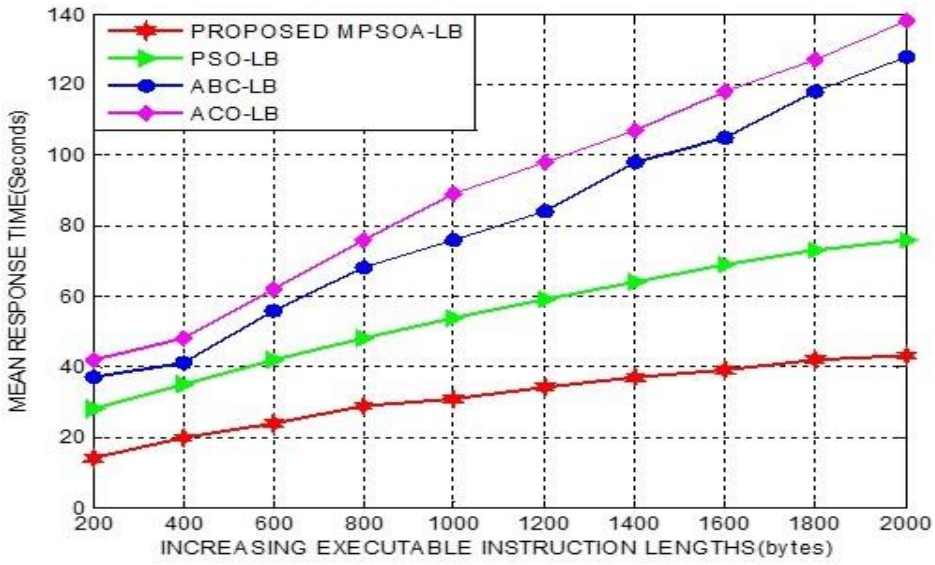

Figure 5: Mean Execution Time under Number of Tasks

The mean response time of the MPSOA-LB, ACO-LB, ABC-LB, and PSO-LB schemes is presented in Figure 4, for feasible instruction length of 2000 to 20,000. The results show that the MPSOA-LB program reduces response time by 6.68%, 7.26% and 8.56% compared to ACO-LB, ABC-LB and PSO-LB respectively. The reason for this improvement is that the scheme's objective function effectively reduces the standard deviation in load distribution among virtual machines (VMs). Figure 5 also shows the mean execution time of these schemes as the no. of tasks is varied from 100 to 1000. The projected outline shows a good efficiency, reducing the execution time by 6.13%, 7.85%, and 8.94% compared to ACO-LB, ABC-LB, and PSO-LB, respectively. The reason for this decrease is mainly because of the scheme's use of dynamic upper and lower threshold values, which help to keep the load balance across the system optimal.

Performance of Migrated Tasks with Varying No. of VMs

The execution of the aimed program has been estimated by analyzing the number of migrated tasks as the no. of VMs increases. Figures 6 and 7 exhibit the efficiency of the scheme by assessing the number of migrated tasks with VM counts for task sets of 200 and 400. The results indicate a significant decrease in the quantity of transferred tasks using the MPSOA-LB scheme, regardless of the number of VMs. This reduction is primarily due to the scheme's flexible allocation and deallocation strategies, governed by the upper and lower availability limits of VMs and hosts, which optimize load balancing. For 200 tasks, the MPSOA-LB scheme reduces migrated tasks by 5.68%, 6.59%, and 7.56% associated to the ACO-LB, ABC-LB, and PSO-LB schemes, respectively.

Figure 6: Migrated Tasks Under Increasing No. of VMs (Tasks-200)

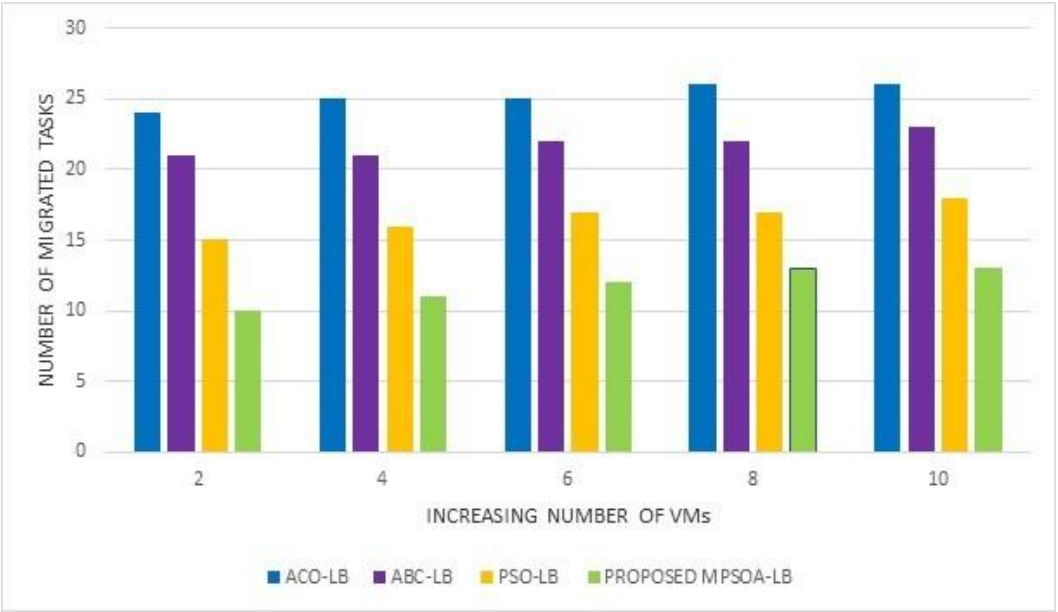

Figure 7: Migrated Task under Increasing Number of VMs (Tasks-400)

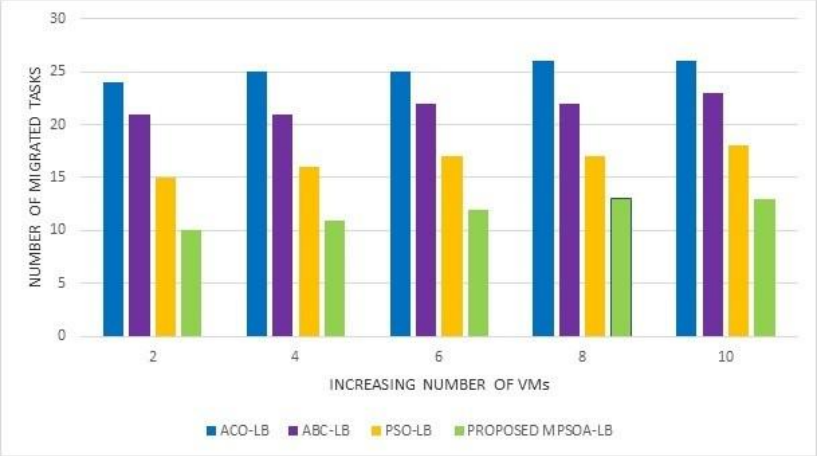

Figure 8: Migrated Task under Increasing Number of VMs (Tasks-600)

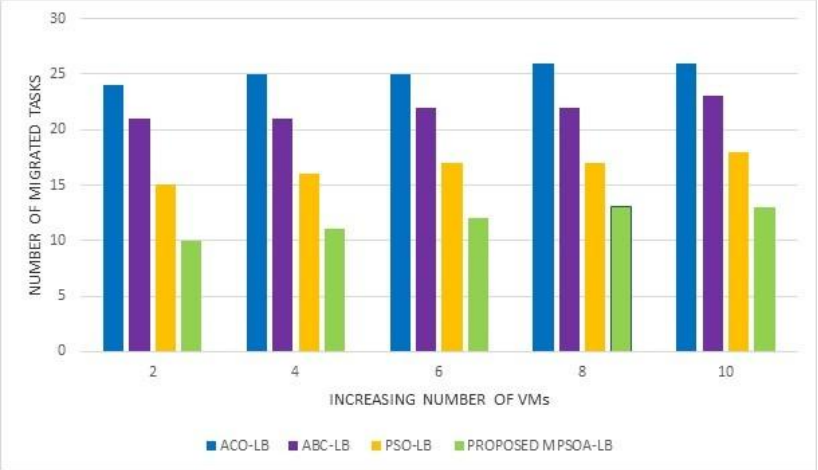

Figure 9: Migrated Task under Increasing Number of VMs (Tasks=800)

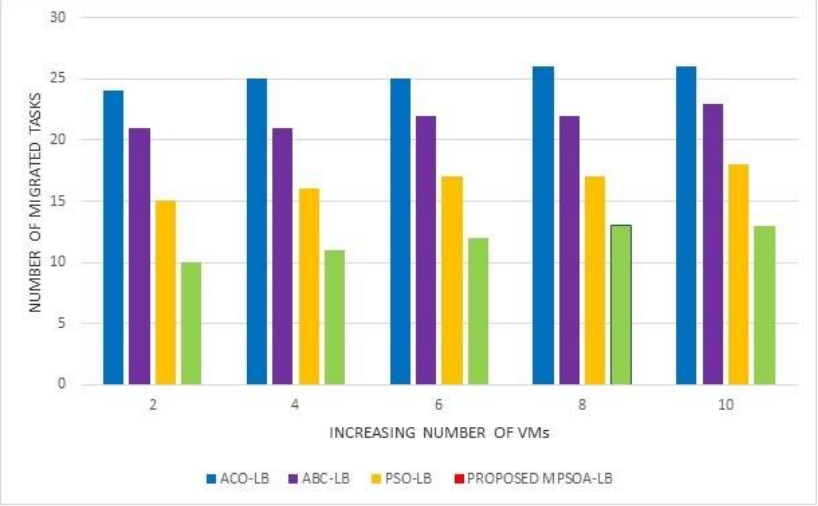

Figure 10: Migrated Task under Increasing Number of VMs (Tasks=1000)

The MPSOA-LB scheme shows a significant decrease in the number of migrated tasks across various VM configurations. When the number of tasks is set to 400, the scheme reduces task migration by 4.86%, 5.68%, and 6.32% equated to the ACO-LB, ABC-LB, and PSO-LB schemes, respectively. Figures 3.8 and 3.9 further demonstrate the efficiency of the MPSOA-LB scheme by evaluating the number of migrated tasks for task counts of 600 and 800, under different VM settings. The MPSOA-LB consistently reduces task migration, irrespective of the VM count, which can be attributed to the multi-objective PSO algorithm's ability to balance exploration and exploitation while considering the accessibility of VMs and hosts in the cloud environment. With 600 tasks, the scheme achieves reductions of 4.12%, 5.68%, and 6.82% compared to ACO-LB, ABC-LB, and PSO-LB, respectively. For 800 tasks, the reductions are 4.32%, 5.94%, and 6.28% compared to the baseline methods. Furthermore, Figure 10 demonstrates that the MPSOA-LB program is still able to minimize the amount of migrated tasks with 1000 tasks and different VM configurations. The

reductions are 4.58%, 5.32%, and 6.58% for 1000 tasks compared to ACO-LB, ABC-LB, and PSO-LB, respectively, which further verifies the effectiveness of the scheme in optimizing load balancing for a growing task load.

Impact of Increasing Tasks on the No. Of Migrated Tasks

To assess the efficiency of the MPSOA-LB structure, the amount of migrated tasks is analyzed as the total task count increases. The scheme is capable of reducing task migration as shown in figures 11 and 12 when the number of Virtual Machines (VMs) is set to 2 and 4 respectively. In both cases, the quantity of migrated tasks decreases significantly with the total quantity of tasks. The adaptive allocation and deallocation mechanism of the MPSOA-LB scheme contributes a lot to this improvement, which adapts to the load balancing threshold dynamically. In particular, the MPSOA-LB scheme reduces the number of migrated tasks by 3.12%, 5.32%, and 6.65% compared to ACO-LB, ABC-LB, and PSO-LB, respectively, when there are 2 VMs. Also, when 4 VMs are used, the scheme shows a considerable decrease of 5.24%, 6.04%, and 7.28% in migrated tasks for the ACO-LB, ABC-LB, and PSO-LB methods.

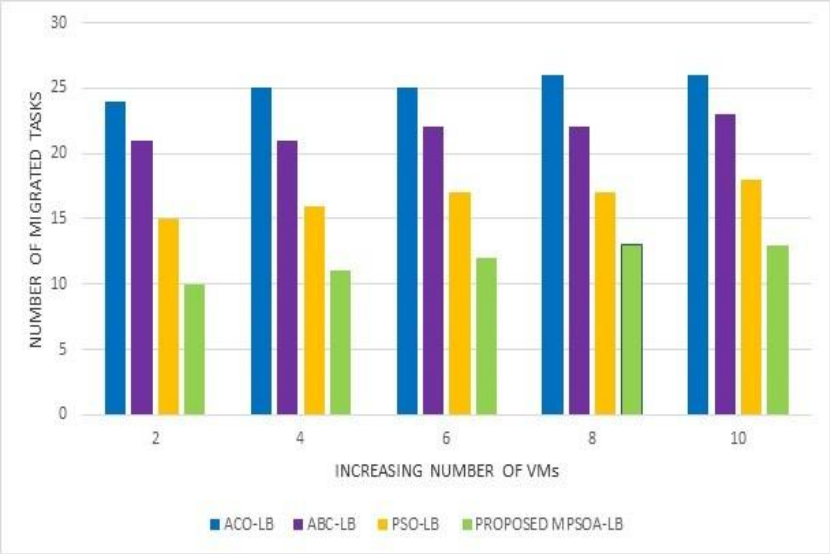

Figure 11: Migrated Tasks under Increasing Number of Tasks (VMs=2)

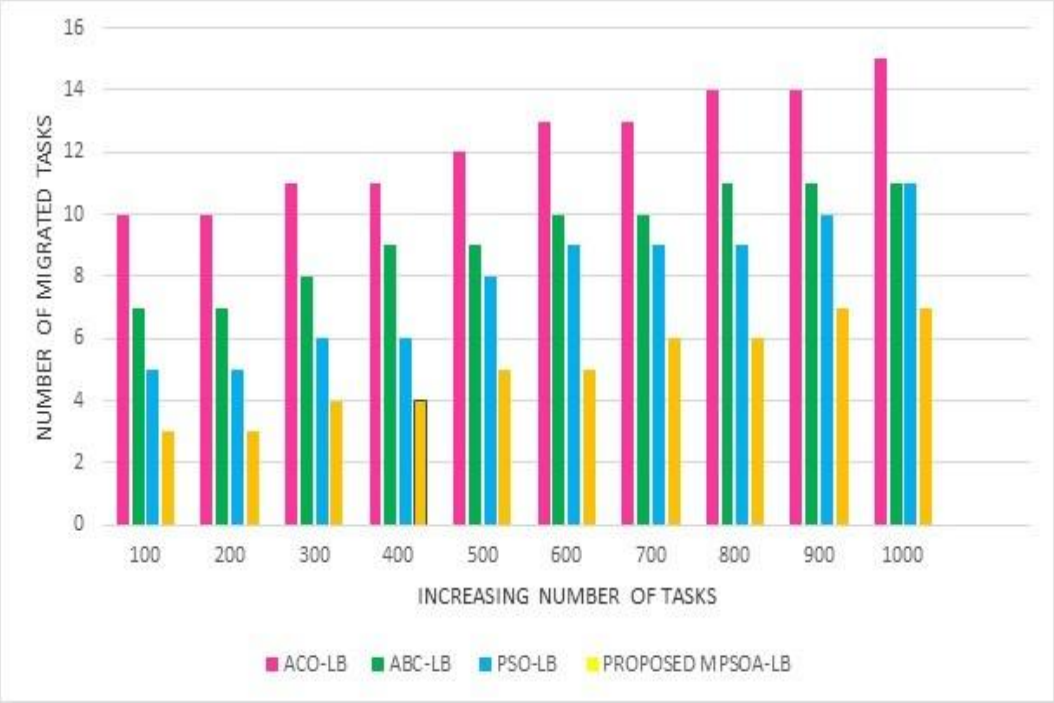

Figure12: Migrated Tasks under Increasing No. of Tasks (VMs=4)

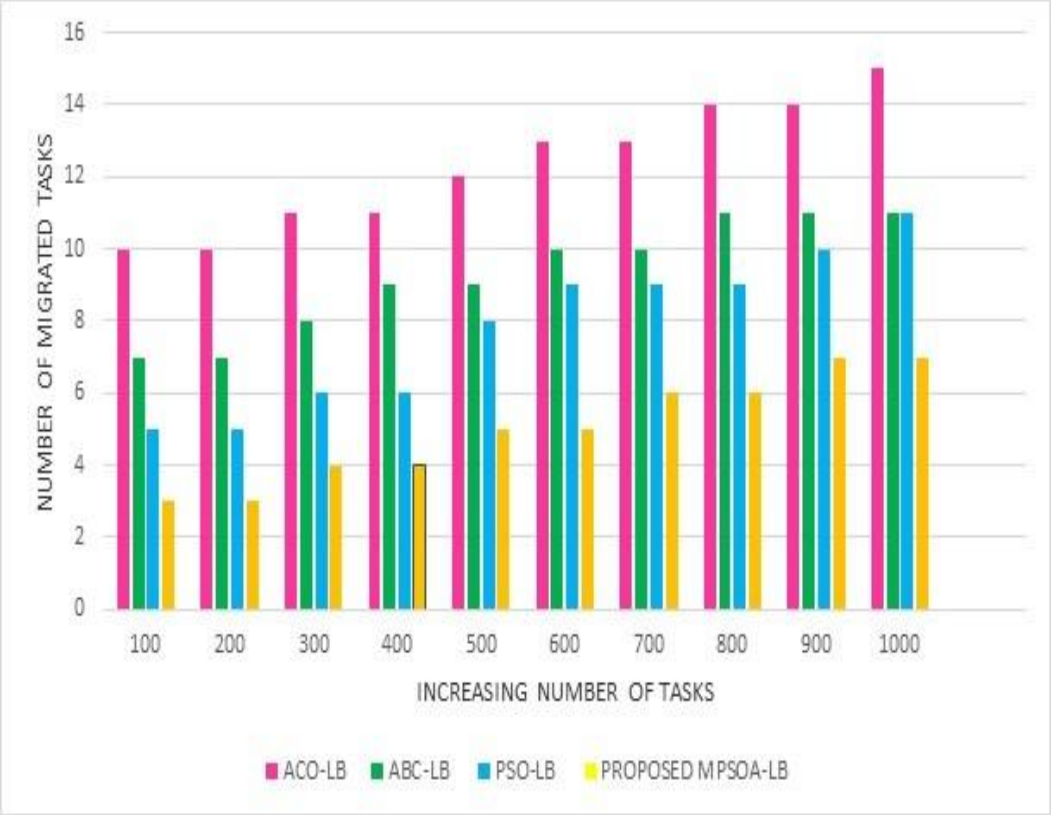

Figure 13: Migrated Tasks under Increasing Number of Tasks (VMs=6)

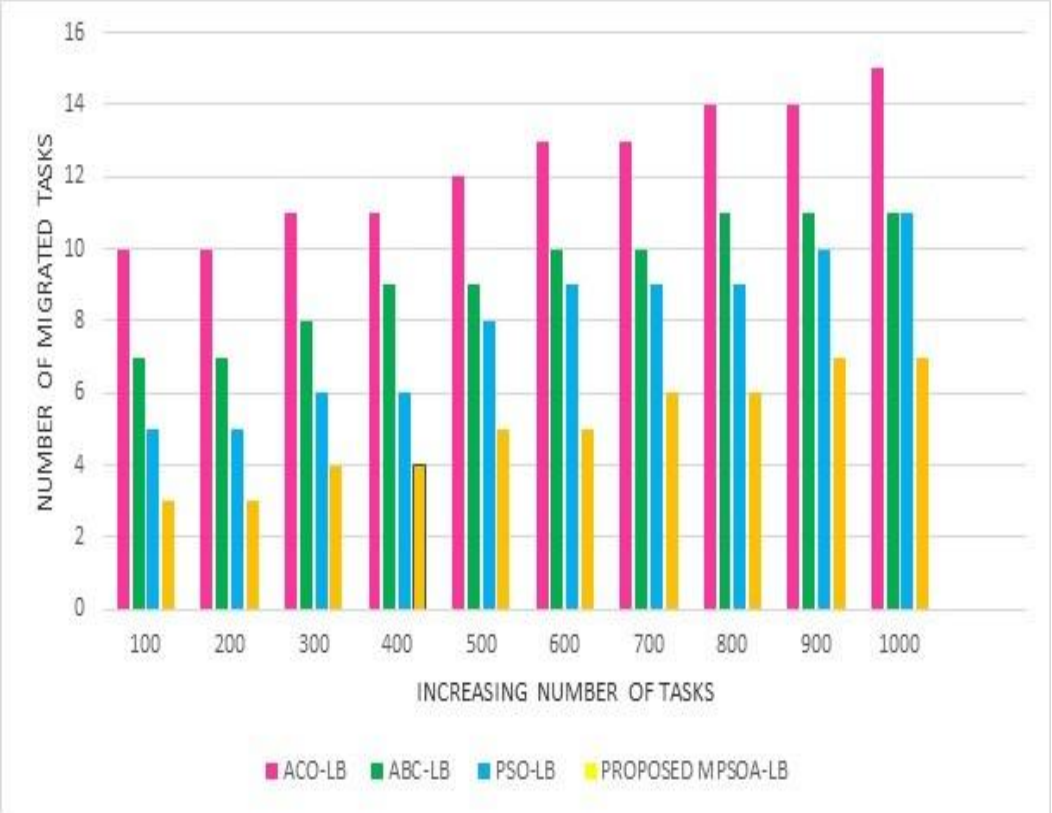

Figure 14: Migrated Tasks under Increasing Number of Tasks (VMs=8)

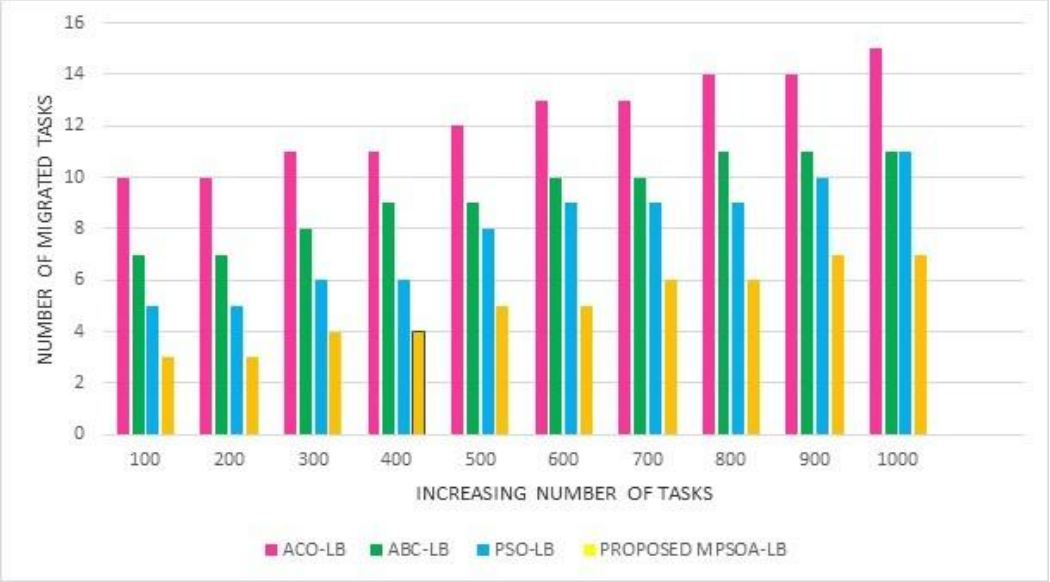

Figure 15: Migrated Tasks under Increasing Number of Tasks (VMs=10)

The efficiency of the MPSOA-LB scheme in minimizing task migration with varying task counts for 6 and 8 virtual machines (VMs) is shown in Figures 13 and 14, respectively. The results show a significant reduction in task migration across all task counts, regardless of the quantity of VMs, when contrasted to the baseline methods. This improvement is attributed to the multi-perspective PSO mechanism, which effectively evaluates the accessibility of VMs and hosts within the cloud environment. For 6 VMs, the MPSOA-LB scheme achieves a reduction in migrated tasks by 5.36%**,** 6.86%**,** and 7.12% equated to the ACO-LB, ABC-LB, and PSO-LB schemes, respectively. With 8 VMs, the reductions are 3.92%**,** 4.65%**,** and 5.32% compared to the same baselines. Similarly, Figure 15 highlights the functioning of the MPSOA-LB scheme with 10 VMs. The scheme continues to effectively reduce the quantity of migrated tasks even with an increased quantity of VMs, leveraging its ability to accurately detect overloaded VMs in the cloud setting. With 10 VMs, the no. of migrated responsibilities is reduced by 4.21%**,** 5.28%, and 6.54% when equated to the ACO-LB, ABC-LB, and PSO-LB approaches.

STATISTICAL ANALYSIS OF THEMPSOA-LB APPROACH

Tables 2-4 provide a statistical evaluation of the MPSOA-LB scheme, considering key metrics such as make-span, system imbalance degree, and the quantity of migrated tasks before and after load balancing. The analysis assesses the performance of MPSOA-LB concerning the baseline ACO-LB, ABC-LB, and PSO-LB schemes, demonstrating its effectiveness and overall performance improvements across all metrics.

Table: Make span (Seconds) of the MPSOA-LB Before and After Load Balancing

Tasks	**MPSOA-LB**		**ACO-LB**		**ABC-LB**		**PSO-LB**	
	Before	**After**	**Before**	**After**	**Before**	**After**	**Before**	**After**
100	12.65	6.79	18.23	9.45	21.38	10.15	23.34	11.65
200	21.56	8.28	23.45	11.35	25.68	12.12	26.78	13.36
300	23.48	9.56	25.46	12.12	29.78	13.46	31.24	14.68
400	29.46	10.76	28.56	12.82	30.24	13.48	32.56	14.98
500	39.42	12.36	30.62	13.12	32.86	14.78	34.68	15.68
600	45.68	14.13	34.42	15.62	36.76	16.78	38.94	17.24
700	59.84	15.14	35.18	16.86	38.82	17.12	39.12	18.92
800	68.74	17.14	36.18	18.12	40.64	17.86	41.34	19.78
900	74.58	20.46	38.92	21.56	41.72	18.36	43.86	19.98
1000	79.84	22.42	38.98	26.65	42.84	18.45	44.18	20.12

Table: Degree of Imbalance in the MPSOA-LB Before and after load balancing

Tasks	MPSOA-LB		ACO-LB		ABC-LB		PSO-LB	
	Before	After	Before	After	Before	After	Before	After
100	1.63	0.34	1.78	0.38	1.88	0.44	1.98	0.45
200	1.67	0.37	1.83	0.45	1.96	0.52	2.25	0.56
300	1.69	0.39	1.89	0.48	2.12	0.59	2.36	0.69
400	1.72	0.43	1.97	0.54	2.18	0.65	2.42	0.74
500	1.78	0.47	2.12	0.58	2.23	0.72	2.46	0.78
600	1.84	0.49	2.19	0.63	2.28	0.78	2.32	0.83
700	1.89	0.52	2.24	0.67	2.32	0.83	2.38	0.87
800	1.94	0.57	2.36	0.69	2.38	0.87	2.42	0.92
900	1.98	0.62	2.42	0.74	2.42	0.92	2.46	1.12
1000	2.12	0.65	2.46	0.78	2.46	1.12	2.32	0.83

The results authorise that the MPSOA-LB program demonstrates superior performance in relations of make-span system inequity, and the number of migrated tasks, both before and after the load-balancing process, when compared to the baseline ACO-LB, ABC-LB, and PSO-LB schemes. This underscores the improved efficiency and effectiveness of the MPSOA-LB approach in optimizing load balancing and resource allocation.

Table 4: Tasks Migrated in MPSOA-LB Before and After Load Balancing

Tasks	MPSOA-LB		ACO-LB		ABC-LB		PSO-LB	
	Before	After	Before	After	Before	After	Before	After
100	7.34	3.21	8.21	3.98	8.88	4.24	9.34	4.78
200	7.38	3.26	8.28	4.04	9.02	4.56	9.46	4.84
300	7.45	3.39	8.38	4.08	9.12	4.58	9.54	4.92
400	7.54	3.45	8.46	4.14	9.19	4.64	9.58	4.95
500	7.59	3.48	8.56	4.19	9.28	4.69	9.65	4.98
600	7.64	3.56	8.64	4.24	9.34	4.78	9.78	5.12
700	7.68	3.68	8.78	4.28	9.46	4.84	9.86	5.19
800	7.76	3.79	8.89	4.34	9.54	4.92	9.94	5.24
900	7.84	3.88	8.96	4.39	9.58	4.95	9.96	5.29
1000	7.89	3.94	8.98	4.45	9.65	4.98	9.98	5.34

The results reveal that the average improvement of the make-span achieved using the MPSOA-LB load balancing scheme before and after the balancing process is 4.12%, 4.98%, and 5.42% when compared to ACO-LB, ABC-LB, and PSO-LB schemes, respectively. The system imbalance was also reduced by an average of 4.16%, 4.86%, and 5.42% after applying the MPSOA-LB scheme compared to the same benchmarks. In addition, the amount of work migrated during load balancing was drop off by an mean of 5.21%, 5.98%, and 6.36% compared to the ACO-LB, ABC-LB, and PSO-LB approaches, respectively. The results of this study show that the MPSOA-LB scheme can improve resource allocation, reduce system imbalance, and

increase system performance. From Table 5-9, it can be observed that the MPSOA-LB scheme outperforms the other schemes in terms of execution cost, energy consumption, communication time, performance time, and resource consumption across multiple iterations of implementation. The results clearly show that the MPSOA-LB scheme is superior to the benchmarked ACO-LB, ABC-LB, and PSO-LB. In particular, the execution cost under different iterations in the MPSOA-LB scheme is reduced significantly with an average reduction of 3.21%, 4.38%, and 5.62%, compared with the benchmark approaches.

Table5: Execution Cost of the MPSOA-LB Scheme for Various Tasks

Task	MPSOA-LB	ACO-LB	ABC-LB	PSO-LB
100	0.0812	0.0887	0.0896	0.0821
200	0.0814	0.0892	0.0898	0.0824
300	0.0816	0.0894	0.0913	0.0927
400	0.0819	0.0895	0.0916	0.0929
500	0.0821	0.0898	0.0919	0.0933
600	0.0824	0.0899	0.0922	0.0936
700	0.0826	0.0913	0.0928	0.0939
800	0.0829	0.0916	0.0932	0.0942
900	0.0832	0.0925	0.0938	0.0946
1000	0.0835	0.0928	0.0945	0.0949

Table 6: Energy Consumption (Joules) of the MPSOA-LB for Various Tasks

Task	MPSOA-LB	ACO-LB	ABC-LB	PSO-LB
100	0.1713	0.1967	0.2134	0.2232
200	0.1719	0.1976	0.2138	0.2238
300	0.1723	0.1978	0.2142	0.2245
400	0.1728	0.1981	0.2145	0.2254
500	0.1732	0.1983	0.2154	0.2259
600	0.1735	0.1986	0.2158	0.2265
700	0.1738	0.1989	0.2162	0.2269
800	0.1742	0.1992	0.2165	0.2273
900	0.1746	0.1995	0.2168	0.2278
1000	0.1756	0.1998	0.2175	0.2287

Table 7: Communication Time (Seconds) of the MPSOA-LB for Various Tasks

Task	MPSOA-LB	ACO-LB	ABC-LB	PSO-LB
100	0.2212	0.2476	0.2712	0.2934
200	0.2218	0.2479	0.2724	0.2945
300	0.2221	0.2484	0.2729	0.2954

400	0.2227	0.2487	0.2734	0.2965
500	0.2229	0.2488	0.2745	0.2976
600	0.2232	0.249	0.2756	0.2984
700	0.2238	0.2494	0.2764	0.2992
800	0.2245	0.2496	0.2775	0.2998
900	0.2268	0.2497	0.2779	0.3012
1000	0.2278	0.2499	0.2785	0.3042

Table 8: Execution Time (Seconds) of the MPSOA-LB for Various Tasks

Task	MPSOA-LB	ACO-LB	ABC-LB	PSO-LB
100	0.3121	0.3345	0.3632	0.3894
200	0.3126	0.3412	0.3645	0.3865
300	0.3129	0.3421	0.3654	0.3869
400	0.3132	0.3428	0.3659	0.3874
500	0.3138	0.3438	0.3665	0.3886
600	0.3143	0.3448	0.3672	0.3892
700	0.3147	0.3512	0.3678	0.3895
800	0.3154	0.3528	0.3682	0.3897
900	0.3158	0.3556	0.3698	0.3898
1000	0.3321	0.3675	0.3689	0.3899

It is shown that the MPSOA-LB scheme outperforms the benchmarked ACO-LB, ABC-LB, and PSO-LB schemes in terms of communication time, implementation time, and resource utilization for fluctuating iterations. The communication time is reduced by 4.04%, 4.86%, and 4.58% on average. It also minimizes the execution time by 4.86%, 5.86%, and 6.54% respectively, and resource utilization by 5.21%, 6.12%, and 6.86% respectively to the above-mentioned methods.

Table 9: Resource Utilization (in%) of the MPSOA-LB for Various Tasks

Tasks	MPSOA-LB	ACO-LB	ABC-LB	PSO-LB
100	0.2124	0.1986	0.1786	0.1612
200	0.2458	0.1975	0.1972	0.161
300	0.2465	0.1968	0.1962	0.1604
400	0.2469	0.1956	0.1942	0.1602
500	0.2474	0.1948	0.1934	0.1589
600	0.2479	0.1932	0.1912	0.1574
700	0.2564	0.1922	0.1902	0.1565
800	0.2568	0.1912	0.1814	0.1554
900	0.2570	0.1904	0.1812	0.1548

1000	0.2589	0.1902	0.1806	0.1542

CONCLUSIONS

The proposed work has thoroughly described the MPSOA LB load balancing scheme that utilizes the advantages of multi-objective PSO to provide a robust solution for dynamic and flexible task allocation to hosts and VMs. It reduces mean response times and execution times and achieves this across task loads and instruction lengths. More specifically, the scheme demonstrated better performance in terms of mean response time reduction of 8.12%, 8.54%, and 9.65% compared to ACO-LB, ABC-LB, and PSO-LB, respectively, for task volumes varying from 100 to 1000. For practicable instruction lengths between 2000 and 20000, the MPSOA-LB scheme also improved response time by 7.14%, 8.21%, and 9.32% over the same baseline schemes. The MPSOA-LB always performed better in terms of execution time. For 1000 task volumes, the execution time was reduced by 7.82%, 8.43%, and 9.42% as compared to ACO-LB, ABC-LB, and PSO-LB, respectively. A second important feature of the MPSOA-LB approach is that it reduces the number of migrated tasks by 4.21%, 5.32%, and 6.65% as the number of VMs increases from 2 to 10. Moreover, as the task numbers increased from 100 to 1000, the scheme decreased the number of migrated tasks by 3.24%, 4.21%, and 5.42% relative to the same baseline schemes. From these findings it is clear that the task allocation using the MPSOA-LB load balancing scheme is more efficient and yields an overall better system performance.

REFERENCES

[1] S. K. Gorva and L. C. Anandachar, "Effective Load Balancing and Security in Cloud Using Modified Particle Swarm Optimization Technique and Enhanced Elliptic Curve Cryptography Algorithm," *International Journal of Intelligent Engineering & Systems*, vol. 15, no. 1, pp. 78–88, 2022.

[2] W. Kimpan, "Multi-Objective Task Scheduling Optimization for Load Balancing in Cloud Computing Using Hybrid Artificial Bee Colony Algorithm with Reinforcement Learning," *IEEE Access, vol. 10, pp. 28987–29002, 2022*

a. Pradhan and S. K. Bisoy, "A Novel Load Balancing Technique for Cloud Computing Platform Based on QMPSO Algorithm," Journal of King Saud University– Computer and Information Sciences,vol. 34, no. 2, pp. 182–191, 2022.

[3] S. H. Fattahiand M. G. Sadiq. (2020)."Multi-Objective Load Balancing in Cloud Computing using Improved Particle Swarm Optimization." International Journal of Cloud Computing and Services Science(IJ-CLOSER), 9(1), 23-34.

[4] P. Vanathi and R. P. Srivastava. (2021). "Dynamic Load Balancing in Cloud Computing using Multi-Objective Particle Swarm Optimization." *Journal of King Saud University - Computer and Information Sciences.*

[5] Ahmed, E., & Abdelrahman, S. (2019). "Optimal Load Balancing Based on PSO for Cloud Computing Environment." *International Journal of Cloud Computing and Services Science* , 8(1), 1-16.

[6] S. K. Agarwal, R. B. Patel, and H. B. Raghuwanshi. (2019). "A Multi-Objective Particle Swarm Optimization Approach for Load Balancing in Cloud Computing." *Journal of Information Processing Systems*, 15(4), 896-906.

[7] Ali, O. H., & Khan, F. (2018). "A Hybrid Load Balancing Algorithm Based on Multi-Objective Particle Swarm Optimization for Cloud Computing." *The Computer Journal*, 61(6), 813-825.

[8] A.P. Shameer and A.C. Subhajini (2017) Optimization Task Scheduling Techniques on Load Balancing in Cloud Using Intelligent Bee Colony Algorithm." International Journal of Pure and Applied Mathematics ", Volume 116 No. 22 2017, 341-352

[9] A.P. Shameer and A.C. Subhajini (2019) Quality of Service Aware Resource Allocation Using Hybrid Opposition-Based Learning-Artificial Bee Colony Algorithm. "Journal of Computational and Theoretical Nanoscience Vol. 16, 588 594, 2019