International Journal of
INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING

ISSN:2147-6799 www.ijisae.org Original Research Paper

Fault Tolerance Enhancement Through Load Balancing Optimization in Cloud Computing

Bikash Chandra Pattnaik [1], Bidush Kumar Sahoo[2], Arabinda Pradhan [3], Satya Ranjan Mishra [4], Himadri Sekhar Tripathy [5], Payal Agasti [6]

Submitted:13/03/2024 **Revised:** 28/04/2024 **Accepted:** 05/05/2024

Abstract: In cloud computing, a significant demand for data requests in order to deliver on-demand services at the lowest possible cost. As a result, servers are essential to handle the cloud requests that are dispersed among several geographic zones. Due to less number of servers available in datacenter, some of them are overloaded and some servers are idle or underloaded. This results in requests failing and degrade the system performance. To solve this issue this paper proposed a Particle Swarm Optimization Based Fault Tolerance Load Balancing algorithm (PSOBFTLB). This algorithm is used to provide the flexible and reliability services to each cloud user and maintain the balance of load in each machine by checking the status. To verify our work, a series of experiments over multiple datasets are done by using the CloudSim simulator. According to the simulation results, the PSOBFTLB algorithm works better while using 5% more resources, reduces 15% of the execution time, 12% of the makespan time, 9% of the average response time, and 8% of the average waiting time. Overall, it increases 12% throughput by taking 10% more task is completed as compare with other algorithms such as DLBA and ACO-VMM algorithm.

Keywords: *Resource Utilization, Fault Tolerance, PSO, Load Balancing, Makespan*

1. Introduction

Cloud computing is growing in popularity as a comprehensive approach to computing. It provides the required services whatever the user wants. These services are on request basis or demand basis, less cost and easily available. Commonly, these services are offered by datacenter where it contains number of servers. Many companies, like Google, Amazon, Microsoft, and many more, have embraced cloud computing in recent years as a dependable and effective computing solution. Due to number of advantages of cloud computing, a greater number of requests are coming from user site. As compare to user request, there is not sufficient servers are available in datacenter. These servers are logically divided into several virtual machines (VMs) with the aid of the virtualization approach, which shows in Fig. 1 [1].

Along with several advantages of cloud computing, there is also some disadvantages of cloud computing and fault tolerance in load balancing is one of them. A system with fault tolerance can keep working even if one of its parts fails [2]. Commonly in cloud computing fault occurs to maintain the load balancing in VMs. As a result, load balancing in cloud computing is a difficult concept [3]. The workload in the cloud can vary periodically based on user demands, making it challenging to allocate these resources [4]. Fault tolerance in load balancing problem can be handled by an effective scheduling algorithm where it can uniformly distribute the entire task among available VM that all VM should be in balanced. Heuristic, meta-heuristic, and hybrid scheduling algorithms are the three types of scheduling algorithms that are available. Among these three types meta-heuristic is better to solve the problem in a limited time period of dynamic environment [5].

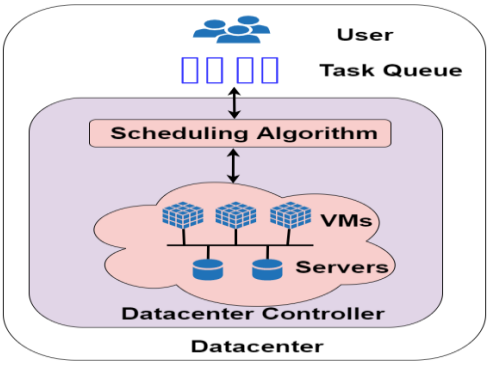

Fig. 1. Cloud computing model

Still, meta-heuristic scheduling algorithms have common drawbacks are: (1) it has slow convergence; (2) maximum time required for selection VM; (3) executes a large

1,3,4,5 Gandhi Institute for Education and Technology, Affiliated to Biju Patnaik University of Technology Rourkela, Odisha, India
ORCID ID: 0009-0000-6713-1492
ORCID ID: 0000-0002-3299-8990
ORCID ID: 0009-0003-2238-6338
ORCID ID:0009-0009-9019-0277
2 GIET University, Gunupur, Odisha, India
ORCID ID: 0000-0002-5044-0819
6Utkal University, Vanivihar, Bhubaneswar, Odisha, India
ORCID ID: 0009-0003-3142-8499
** Corresponding Author Email: arabindapradhan@giet.edu.in*

number of dynamically allocated tasks. Among various meta-heuristic scheduling, PSO technique is most popular due to its simplicity nature and less parameter is required to be finding the optimum result [6] and it is proposed by Kennedy and Eberhart [7]. There are various types of PSO method are available and Modified PSO (MPSO) is one of them [8]. Commonly in standard PSO, local optimum problems have been demonstrated that cause slow rate of convergence. This is why we concentrated on a modified PSO algorithm to maintain the load and proposed Particle Swarm Optimization Based Fault Tolerance Load Balancing scheduling (PSOBFTLB) algorithm. Proposed scheduling approach uses the linearly decreasing inertia weight parameter to solve local optimum problem and produces a better fitness function to choose the suitable VM and minimize the execution time.

Our work has contributed to the literature in the following ways: (1) Creating the PSOBFTLB method for fault tolerance load balancing, which focuses task migration, instead of migrating the entire overloaded VM. (2) Reduce makespan time and make the most use of available resources. (3) Verify the effectiveness of our approach by using the Jswarm and CloudSim program. The rest of this paper is structured as follows: The relevant task is displayed in Section 2, and the goal function is shown in Section 3. We go over the PSO approach in part 4. The PSOBFTLB model structure is described in Section 5. Performance evaluation is shown in Section 6. Section 7 contains the paper's conclusion and future directions.

2. Related Work

Numerous techniques have been created by different researchers to addresses different fault tolerance metrics in load balancing, including makespan time, resource utilization, etc. Such methods are based on PSO based scheduling method and VM migration (VMM) concept.

Modified central scheduler load balancing (MCSLB) method is proposed in [9]. The main role of this technique is to move VMs from the heavy load host to the lightest host based on VM migration. In order to minimize or prevent dynamic migration and attain optimal load balancing, [10] suggested a genetic algorithm-based scheduling technique for VM resource load balancing that takes system variance and historical data into account. Migration Management Agent (MMA) algorithm is proposed in [11] for dynamically balancing virtual machine loads. A two-stage genetic mechanism is proposed in [12] for the migration-based load balance of virtual machine hosts (VMHs) in cloud computing. This approach uses gene expression programming (GEP) to create symbolic regression models that forecast the loads of virtual machine heads (VMHs) during load balancing and characterize VM performance. In [13], a distributed virtual machine migration plan based on the Ant Colony Optimization (ACO) method is put forth. This method aims to minimize the number of migrations while achieving load balancing and reasonable resource use. In [14] proposed guaranteed fault-tolerant requirement load balancing scheme (GFTLBS). This scheme is used to migrates the VMs to maintain the load as balanced. A fault tolerance technique for handling server failures is developed in [15]. It involves relocating the VMs that are located on the downed server. Several load balancing techniques made for cloud computing environment is proposed in [16]. According to the study results, the majority of meta-heuristics outperform conventional heuristics in terms of results. In order to maintain a balanced load across VMs, a dynamic load balancing solution is suggested in [17]. In order to optimize VM utilization with a uniform load distribution, [18] proposed a hybrid method-based Deadline-constrained, Dynamic VM Provisioning and Load Balancing architecture that may reduce costs and makespan. Logarithm PSO based task scheduling algorithm is proposed in [19] which is used to reduce makespan time. A PSO-based scheduling technique was created in [20] and is utilized to increase resource utilization while reducing makespan. A clustering strategy is utilized in [21] to obtain the improved PSO (IPSO) algorithm, which is used to minimize the makespan time.

From the existing method, we discovered that the majority of researchers are attempting to balance the load across all VMs by using VMM technique, decrease makespan, and maximize resource consumption. VMM concept shows high cost that can affect the entire services. Instead of VMM, we use task migration concept by checking status of VM. This mechanism helps to avoid the fault occur when balance the load in VM.

3. Objective Function

Selection of suitable VM is depending on the current load and capacity of VM. Once the optimal VM has been identified, allocate the task to it in order to minimize execution time and enhance system efficiency. In order to manage these goals, the suggested scheduling first gathers all the data on incoming tasks and available virtual machines (VMs), including task number (T_t), task length (T_{length}), task file size (T_{fs}), VM number, CPU (VM_{cpu}), memory, MIPS (VM_{mips}) and bandwidth (VM_{bw}) [22].

3.1 Selection of VM

Initially, tasks are waiting in task queue and allocated to a VM on first come first serve order. The choice of VM at each iteration is determined by the VM's current load (VM_{load}) and capacity (VM_{capa}) which is shown in Eq. (1) and (2). The processing rate of VM (VM_{pr}) is calculated by using Eq. (3).

$$VM_{load} = \frac{T_t \times T_{length}}{VM_{pr}} \quad \ldots (1)$$

$$VM_{capa} = VM_{pr} + VM_{bw} \quad \ldots (2)$$

$$VM_{pr} = VM_{mips} \times VM_{cpu} \quad \ldots (3)$$

Based on the capacity and load of the VM; each VM has three states: balanced, underload, and overload. We have to take an assumption that if a VM load is under 25% of its capacity then it is under load, if a VM load is over 80% of its capacity then it is overloaded otherwise it is in balanced stage which is shown in Eq. (4) [23]. After checking the VM stages, the best VM is selected (VM_{sel}) for handle the task.

$$VM_{sel} = \begin{cases} VM_{load} < |VM_{capa} \times 25\%|, Underload \\ VM_{load} > |VM_{capa} \times 80\%|, Overload \\ Otherwise,\ Balanced \end{cases} \quad \ldots (4)$$

3.2 Makespan Time

After selecting the best VM from datacenter then task is allocated to be reducing the overall makespan time. Consequently, determine the expected execution time ($ET_{expected}$) which is representing as in Eq. (5).

$$ET_{expected} = \frac{T_{length}}{VM_{pr}} \quad \ldots (5)$$

Task allocation time (TA_{time}) is the amount of time required to allocate the task on to the VM. It is based on the task file size relative to the bandwidth of VM, as shown in Eq. (6).

$$TA_{time} = T_{fs}/VM_{bw} \quad \ldots (6)$$

Total execution time (ET_{total}) is the sum of the task allocation time and the expected execution time, as given by Eq. (7).

$$ET_{total} = ET_{expected} + TA_{time} \quad \ldots (7)$$

When more tasks are completed successfully at the datacenter, makespan time (MS_{time}) can decrease. It can be ascertained by calculating the maximum of the execution time as a whole using Eq. (8).

$$MS_{time} = max\{ET_{total}\} \quad \ldots (8)$$

3.3 Resource Utilization

The next optimization goal is to raise the value of(R_{util}), the resource usage, as it appears in Eq. (9). Resource usage is calculated as the total execution time of all tasks divided by the makespan.

$$R_{util} = \frac{ET_{total}}{MS_{time}} \quad \ldots (9)$$

3.4 Fitness Function

The fitness function (F_{func}) of the PSOBFTLB algorithm is now defined as Eq. (10), which provides a particle with an optimal solution and a better position.

$$F_{func} = MS_{time} + R_{util} \quad \ldots (10)$$

To collect the above information, by applying our proposed method scheduler can get the knowledge that which VM is now idle or busy. Then it transfers the extra task or next incoming task on selected VM. Instead of VMM technique we used task migration technique to reduce the cost of services.

4. Particle Swarm Optimization (PSO)

PSO is a swarm-based intelligence method [7] where each particle can find their personal best (P_{best}) and global best (G_{best}) to get the optimum result. Based on particle's fitness value their P_{best} and G_{best} can be evaluated. Also, their velocity and position can be modified. Every iteration updates each particle's location and velocity using the subsequent Eq. (11) and Eq. (12) [22].

$$X_i(t+1) = X_i(t) + V_i(t+1) \quad \ldots (11)$$

$$V_i(t+1) = \omega V_i(t) + c_1 r_1 \big(Pbest(t) - X_i(t)\big) + c_2 r_2 \big(Gbest(t) - X_i(t)\big) \quad \ldots (12)$$

5. Model of PSOBFTLB

The proposed approach organizes the number of upcoming tasks based on the fitness function relative to the available cloud resources. During the execution, the values for P_{best} and G_{best} are updated based on particle's fitness function. The optimal position of every particle is identified by comparing its current fitness value with its unique P_{best} value. In a similar manner, it considers the current fitness value of the population as a whole to calculate the best fitness value G_{best}. The scheduler periodically verifies the state of each VM and determines whether a job should be assigned to a VM or not. This process is repeated until the task queue is not empty. Proposed scheduling algorithm is modified two important parameters such as inertia weight and convergence rate. Both are used to improve the fitness function for getting an effective result as compare to existing fault tolerance load balancing method. Fig. 2 shows the proposed scheduling model.

5.1 Inertia Weight Parameter

Inertia weight parameter (ω) is an important concept which is used to avoid the local optimum problem. It regulates the particle's momentum and determines how much the previous flight direction will affect the new velocity [24]. According to early empirical research the value of ω is belong in to 0 to 1. In our proposed method, we used linearly decreasing inertia weight parameter which is represented in Eq. (13).

$$\omega = \omega_{max} - \frac{\omega_{max}-\omega_{min}}{k_{max}} \times k \quad \ldots (13)$$

Where ω_{max} particle swarm's maximum inertia weight. The particle swarm's minimal inertia weight is denoted by ω_{min}. k_{max} denotes the particle swarm's maximum number of iterations.

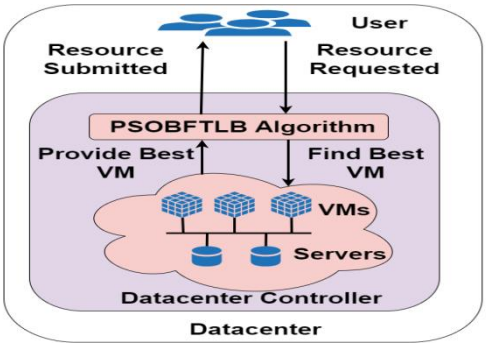

Fig. 2. PSOBFTLB model

5.2 Convergence Rate

In order to guarantee convergence to a stable point, Clerc and Kennedy analysed particle trajectories [25] which leads to a modification in the velocity equation, as shown in Eq. (14) and the constriction coefficient, χ, is determined using the formula in Eq. (15) where κ controls the rate of convergence.

$$V_i(t+1) = \chi V_i(t) + c_1 r_1\big(Pbest - X_i(t)\big) + c_2 r_2\big(Sbest - X_i(t)\big) + c_3 r_3\big(Gbest(t) - X_i(t)\big) \quad \ldots (14)$$

$$\chi = \frac{2\kappa}{|2 - \varphi - \sqrt{\varphi^2 - 4\varphi}|} \quad \ldots (15)$$

5.3 Pseudocode of PSOBFTLB

Input: VM set as $VM_m = \{VM_1, VM_2, \ldots, VM_p\}$ and Task set as $T_t = \{T_1, T_2, \ldots, T_q\}$
Output: Reduce makespan time while maximizing resource use.

1. **Begin Method (T, VM)**
2. Initialize ω, c_1, c_2, population size, iterations, r_1, r_2 and $Pbest$
3. **While** (T! = ∅)
4. **For** 1 to p
5. **For** 1 to q
6. Utilizing the PSOBFTLB algorithm, determine the F_{func}
7. To determine the optimal value, compare the current fitness value with P_{best}
8. **If** ($F_{func} \leq P_{best}$)
9. Set current F_{func} as new P_{best}
10. **End** of **if**
11. Compare all of your own best fitness to the global best fitness.
12. Find $G_{best} \curlyvee P_{best}$
13. **If** $P_{best} \leq G_{best}$
14. Assign P_{best}as global best
15. **Else**
16. Assign G_{best} as global best
17. Return G_{best}
18. **End** of **if**
19. Update iteration
20. Modify the particle's position and velocity
21. Continue doing so until achieve the best outcome
22. **End For**
23. **End For**
24. **End While**
25. **End Method**

6. Assessment of Performance

The suggested PSOBFTLB technique is compared with Dynamic Load Balancing Algorithm (DLBA) and VM migration strategy based on Ant Colony Optimization (ACO-VMM). The CloudSim tools and Eclipse Java Programming Environment are used to implement the suggested PSOBFTLB. Our PC was configured with an

Table 1. Properties of tasks, VMs, servers and PSO parameters.

Task range	10-50	VM range	3-5	Server range	5	Number of particles	50
Length	1000-6000	MIPS	250-300	MIPS	3000	c_1&c_2	2
File Size	300	Memory	256-512	Memory	512	r_1&r_2	[0,1]
		CPU	1-5	CPU	7	Maximum iteration	100
		Bandwidth	1000	Bandwidth	3000		

Intel (R) Core (TM) i3-7100 processor running at 2.40 GHz and 4 GB of RAM. Windows 10 was the 64-bit operating system. The simulation's parameters are displayed In Table 1, simulation results are displayed from Fig. 3 to Fig. 12, and the corresponding datasets are displayed from Table 2 to Table 11.

6.1 Execution time

Execution time is shown in Fig. 3 and Fig. 4. The dataset used in Fig. 3 is Table 2, which contains 1000 to 5000 tasks and 50 numbers of VMs. Similarly, Table 3 and Fig. 4 have 10 to 50 VMs and 5000 number of tasks. Initially, in Fig. 3 we have found that the proposed scheduling algorithm reduces approximate 7% to 15% execution time when the number of tasks increases from 1000 to 5000. Similarly, in Fig. 4 the proposed algorithm is reduced by approximate 5% to 11% of execution time as compared to its competitor.

6.2 Makespan time

Makespan time is shown in Fig. 5 and Fig. 6. Table 4 is the dataset for Fig. 5, which contains 1000 to 5000 tasks and 50 numbers of VMs. Table 5 is the dataset for Fig. 6, which contains 5000 number of tasks and 10 to 50 numbers of VMs. Initially, in Fig. 5 we have to found that proposed scheduling algorithm reduce approximate 9% to 12% makespan time when number of tasks is increases. Similarly, Fig. 6 reducing approximate 8% to 11 of makespan time as compared to its competitor.

6.3 Resource utilization

Fig. 7 shows the average resource utilization of 1000 to 5000 number of task on 50 numbers of VMs. From the figure we have find that the proposed algorithm can be increase the average resource utilization as around 2% to 4%. Table 6 shows the dataset for Fig. 7. Fig. 8 shows the average resource utilization of 10 to 50 numbers of VMs where 5000 number of tasks is taken. From this figure we have find that the proposed algorithm can be increase the average resource utilization as around 2% to 5%. Table 7 shows the dataset for Fig. 8.

6.4 Response time

Fig. 9 is the illustration of average response time for selected algorithm and find that minimum response time is 461 which is PSOBFTLB algorithm. Table 8 is the dataset for Fig. 9. Average response time of PSOBFTLB algorithm is reduced around 5% to 9% as compared to DLBA and ACO-VMM scheduling algorithm.

6.5 Task completed

The quantity of tasks finished by each scheduling strategy is displayed in Fig. 10. This result indicates that, in comparison to the current approach, our suggested algorithm improves the task completion count. This graph displays the algorithm's performance after 1000–5000 tasks are run over 50 VMs. Table 9 shows the dataset for Fig. 10. Total task completed of PSOBFTLB is improved around 6% to 10% as compare to DLBA and ACO-VMM scheduling algorithm.

6.6 Average waiting time

Table 10 and Fig. 11 shows data set and graph for average waiting time of all task on to VM for processing. From this information, we found that proposed scheduling is reduce approximate 5% to 8% waiting time as compare to DLBA and ACO-VMM algorithm.

6.7 Throughput

Fig. 12 demonstrate that, in comparison to previous baseline algorithms, the created PSOBFTLB has improved approximate 9% to 12% of throughput as compared to two mentioned techniques. Table 11 is the dataset for Fig. 12.

Table 2. Execution time in second with 50 numbers of VMs and 1000 to 5000 tasks.

Number of VM	Number of Task	PSOBFTLB	DLBA	ACO-VMM
50	1000	8574.71	8801.83	9185.23
50	2000	10851.45	11075.92	11713.47
50	3000	14383.75	15423.51	16673.54
50	4000	19714.70	21227.28	22852.35
50	5000	24237.32	25896.94	28137.16

Table 3. Execution time in second with 5000 tasks and 10 to 50 number of VMs.

Number of VM	Number of Task	PSOBFTLB	DLBA	ACO-VMM
10	5000	44598.68	45193.29	46101.42
20	5000	41687.34	42271.71	43259.35
30	5000	37465.12	38887.68	40311.92
40	5000	34981.43	36884.94	38872.26
50	5000	33102.42	34843.94	37115.36

Table 4. Makespan time of different algorithms with 50 number of VMs and 1000 to 5000 task sets.

Number of VM	Number of Task	PSOBFTLB	DLBA	ACO-VMM
50	1000	1684.13	1706.91	1666.91
50	2000	1769.51	1784.23	1773.47

50	3000	1772.54	1832.25	1892.97
50	4000	1314.18	1959.24	1963.91
50	5000	1935.94	2107.16	2178.26

Table 5. Makespan time of different algorithms with 5000 tasks and 10 to 50 number of VMs.

Number of VM	Number of Task	PSOBFTLB	DLBA	ACO-VMM
10	5000	2528.89	2614.42	2681.69
20	5000	2357.72	2408.35	2478.12
30	5000	2218.68	2264.97	2307.96
40	5000	2108.94	2177.26	2229.52
50	5000	1919.84	2074.46	2144.76

Table 6. Average resource utilization with 50 numbers of VMs and 1000 to 5000 number of tasks.

Number of VM	Number of Task	PSOBFTLB	DLBA	ACO-VMM
50	1000	0.71	0.703	0.698
50	2000	0.77	0.755	0.74
50	3000	0.81	0.79	0.775
50	4000	0.85	0.83	0.81
50	5000	0.86	0.84	0.823

Table 7. Average resource utilization with 5000 number of tasks and 10 to 50 number of VMs.

Number of VM	Number of Task	PSOBFTLB	DLBA	ACO-VMM
10	5000	0.69	0.66	0.65
20	5000	0.87	0.81	0.76
30	5000	0.91	0.89	0.85
40	5000	0.95	0.93	0.90
50	5000	0.96	0.94	0.91

Table 8. Average response time of different algorithms.

Load balancing parameter	PSOBFTLB	DLBA	ACO-VMM
Average response time	461	485	505

Table 9. Task completed of different algorithms with 1000 to 5000 number of tasks.

Number of Task	PSOBFTLB	DLBA	ACO-VMM
1000	990.14	910.84	885.38
2000	1990.17	1909.76	1828.81
3000	2987.49	2802.58	2709.78
4000	3991.41	3773.83	3662.22
5000	4991.58	4701.07	4508.45

Table 10. Average waiting time in second with 10 to 100 numbers of iterations.

Iteration	PSOBFTLB	DLBA	ACO-VMM
20	78	84	86
40	80	84	86
60	78	83	85
80	80	83	87
100	80	84	87

Table 11. Throughput value in seconds of different algorithms with 1000 to 5000 number of tasks.

Schedule algorithm	1000	2000	3000
PSOBFTLB	29.14	30.17	31.09
DLBA	27.48	27.94	28.68
ACO-VMM	26.58	27.21	27.78

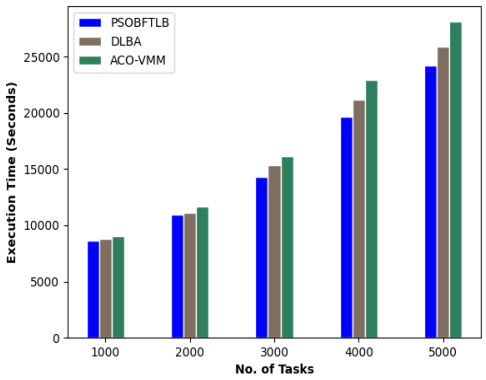

Fig. 3. Execution time of different tasks

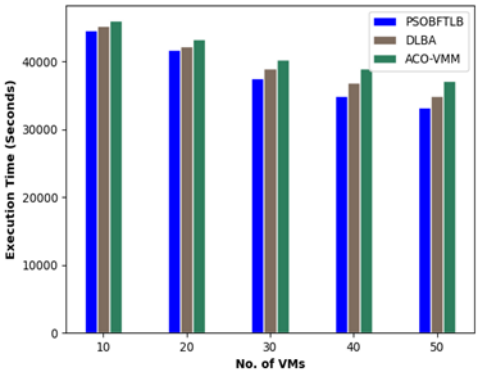

Fig. 4. Execution time of different VMs

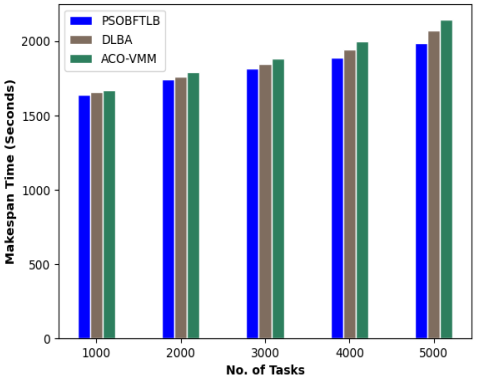

Fig. 5. Makespan time of different tasks

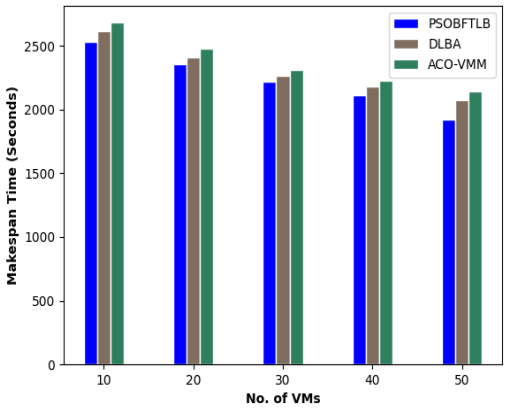

Fig. 6. Makespan time of different VMs

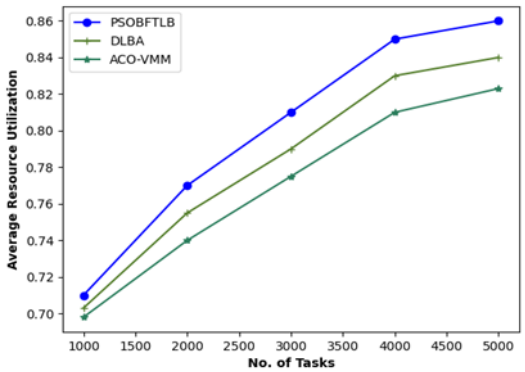

Fig. 7. Average resource utilization obtained for 1000 to 5000 tasks

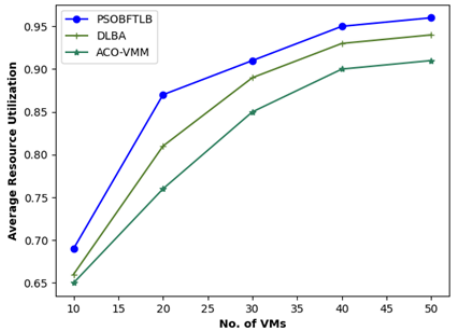

Fig. 8. Average resource utilization obtained for 10 to 50 VMs

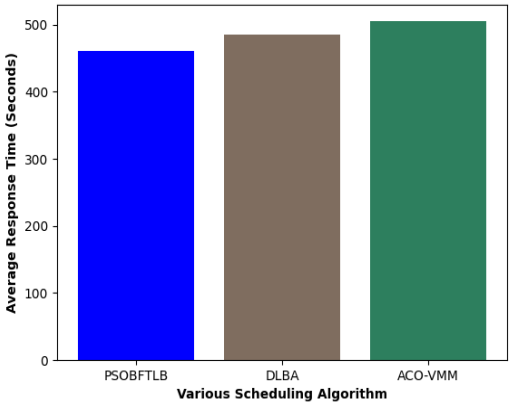

Fig. 9. Comparison average response time between various scheduling algorithm

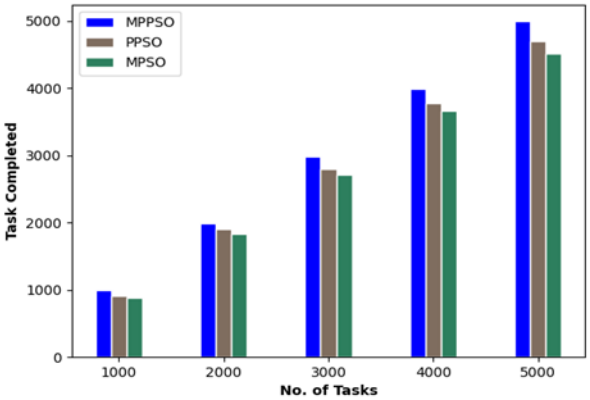

Fig. 10. Comparison of task completed between various scheduling algorithm

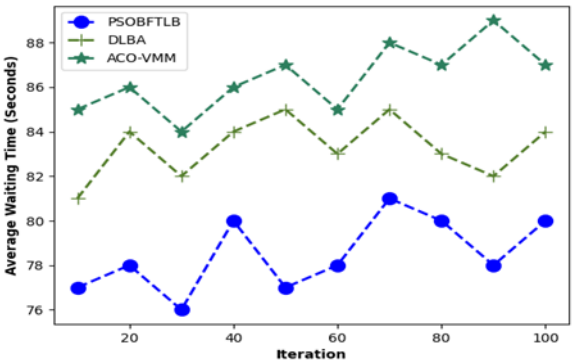

Table 11. Throughput value in seconds of different algorithms with 1000 to 5000 number of tasks

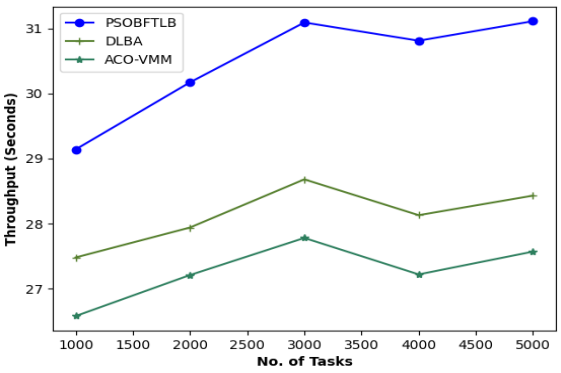

Fig. 12. Throughput value

11. Conclusion

Choosing the best resource in a cloud environment can be difficult because of the variety and complexity of new applications, intricate pricing schemes, and various provisioning plans. Thus, creating an effective scheduling method for cloud environments that enhances necessary QoS metrics within user-specified time frames is crucial. Several baseline scheduling methods, as well as their benefits and drawbacks in a cloud context, have been covered in this work. Using the PSOBFTLB method, the resource monitoring continuously verifies the state of VMs, the scheduler schedules tasks at VMs, and the controller uses an optimal task-to-resource mapping to provision and de-provision resources based on user demand. Performance of PSOBFTLB algorithm is evaluated using the CloudSim simulator and contrasted with baseline algorithms that are currently in use, including DLBA and ACO-VMM. Simulation results, shown in Table 2 to Table 11 and Fig. 3 to Fig. 12. In the future, we'll develop an autonomous resource provisioning strategy that uses a hybrid algorithm to maximize various QoS metrics within a user-defined deadline while maintaining the SLA.

The primary conclusions from the experimental study, in comparison to DLBA and ACO-VMM, are as follows:

- PSOBFTLB algorithm performs better in reducing approximately 15% execution time.
- The proposed algorithm reduces by approximate 12% makespan time.
- The proposed algorithm reduces by approximate 9% average response time.
- The proposed algorithm reduces 8% of overall waiting time.
- The proposed algorithm increases by approximate 5% average resource utilization.
- The proposed algorithm improves to complete a vast number of tasks by approximate 10%.
- The proposed algorithm improves by approximate 12% throughput.

References

[1] Pradhan, A., Bisoy, S. K., Kautish, S., Jasser, M. B., Mohamed, A. W.: Intelligent Decision-Making of Load Balancing Using Deep Reinforcement Learning and Parallel PSO in *Cloud Environment. IEEE Access, vol. 10, 76939-76952 (2022)*.

[2] Rehman, A. U., Aguiar, R. L., Barraca, J. P.: Fault-Tolerance in the Scope of Cloud Computing. *IEEE Access, vol. 10, pp. 63422-63441 (2022)*.

[3] Jena, U. K., Das, P. K., Kabat, M. R.: Hybridization of meta-heuristic algorithm for load balancing in cloud computing environment. *Journal of King Saud University- Computer and Information Sciences. Elsevier, pp 1-11 (2020)*.

[4] Pradhan, A., Bisoy, S. K., Sain, M.: Action-Based Load Balancing Technique in Cloud Network Using Actor-Critic-Swarm Optimization. *Wireless Communications and Mobile Computing, Wiley, Hindawi, Volume 2022, Article ID 6456242, 1-17 (2022)*.

[5] Kumar, M., Sharma, S.C., Goel, A., Singh, S. P.: A comprehensive survey for scheduling techniques in cloud computing. *Journal of Network and Computer Applications. Elsevier, Vol. 143, pp 1–33 (2019)*.

[6] Masdari, M., Salehi, F., Jalali, M., Bidaki, M.: A Survey of PSO-Based Scheduling Algorithms in Cloud Computing. *Journal of Network and Systems Management, 25(1), pp. 122-158 (2016)*.

[7] Kennedy, J., Eberhart, R.C.: Particle Swarm Optimization. Proceedings of the IEEE *International Conference on Neural Networks, Piscataway, NJ, USA, 1942–1948 (1995)*.

[8] Pradhan, A., Bisoy, S. K., Das, A.: A survey on PSO based meta-heuristic scheduling mechanism in cloud computing environment. *Journal of King Saud University –Computer and Information Sciences, Elsevier (2021)*.

[9] Wilcox Jr, T.C.: Dynamic load balancing of Virtual machines hosted on Xen. *Master's Thesis. Brigham Young University, USA (2009)*.

[10] Hu, J., Gu, J., Sun, G., Zhao, T.: A Scheduling Strategy on Load Balancing of Virtual Machine Resources in Cloud Computing Environment. *3rd Int. Symp. Parallel Architectures, Algorithms and Programming (PAAP), Dalian, China, pp. 89–96 (2010)*.

[11] Song, X., Ma, Y., Teng, D. A.: Load balancing scheme using federate migration based on virtual

machines for cloud simulations. *Mathematical Problems in Engineering, Volume 2015, Article ID 506432, pp 1-11 (2015).*

[12] Hung, L. -H., Wu, C. -H., Tsai, C. -H., Huang, H. -C.: Migration-Based Load Balance of Virtual Machine Servers in Cloud Computing by Load Prediction Using Genetic-Based Methods. *IEEE Access, Vol. 9 PP. 49760- 49773 (2021)*.

[13] Wen, W. -T., Wang, C. -D., Wu, D. -S., Xie Y. -Y.: An ACO-based Scheduling Strategy on Load Balancing in Cloud Computing Environment. *2015 Ninth International Conference on Frontier of Computer Science and Technology, IEEE, pp 364–369 (2015).*

[14] Yao, L., Wu, G., Ren, L., Zhu., Y., Lin, Y.: Guaranteeing Fault-Tolerant Requirement Load Balancing Scheme Based on VM Migration. *The Computer Journal, Vol. 57, No. 2, 225-232 (2014)*.

[15] Joshi, S. C., Sivalingam, K. M.: Fault tolerance mechanisms for virtual data center architectures. *Photonic Network Communications, 28(2), pp. 154-164 (2014).*

[16] Xu, M., Tian, W., Buyya, R.: A survey on load balancing algorithms for virtual machines placement in cloud computing. *Concurrency Computat: Practice and Experience, 29(12), e4123, pp. 1-16, (2017)*.

[17] Kumar, M., Sharma, S. C.: Dynamic load balancing algorithm to minimize the makespan time and utilize the resources effectively in cloud environment. *IJCA, Taylor & Francis, pp 1- 10 (2017).*

[18] Kaur, A., Kaur, B.: Load balancing optimization based on hybrid Heuristic- Metaheuristic techniques in cloud environment. *Journal of King Saud University- Computer and Information Sciences. Elsevier, pp 1-12 (2019).*

[19] Huang, X., Li, C., Chen, H., An, D.: Task scheduling in cloud computing using particle swarm optimization with time varying inertia weight strategies. *Cluster Computing, Springer, 23, 1137–1147 (2020).*

[20] Agarwal, M., Srivastava, G. M. S.: A PSO Algorithm Based Task Scheduling in Cloud Computing. *IJAMC. Volume 10, Issue 4, pp 1-17 (2019).*

[21] Saleh, H., Nashaati, H., Saber, W., Harb, H. M.: IPSO Task Scheduling Algorithm for Large Scale Data in Cloud Computing Environment. *IEEE Access, Volume 7, pp 5412-5420 (2019).*

[22] Pradhan A., Bisoy, S. K.: A novel load balancing technique for cloud computing platform based on PSO. *Journal of King Saud University –Computer and Information Sciences, Elsevier, Volume 34, Issue 7, pp 3988-3995 (2022).*

[23] Kumar, M., Sharma, S. C.: PSO-based novel resource scheduling technique to improve QoS parameters in cloud computing. *Neural Computing and Applications, Springer, 32, 12103–12126 (2020).*

[24] Eberhart, R. C., Shi, Y.: Comparing Inertia Weights and Constriction Factors in Particle Swarm Optimization. *Proceedings of the IEEE Congress on Evolutionary Computation, San Diego, USA (2000)*.

[25] Clerc, M., Kennedy, J.: The Particle Swarm-Explosion, Stability, and Convergence in a Multidimensional Complex Space. *IEEE Transactions on Evolutionary Computation, 6(1), pp 58–73 (2002)*.