I.J. Image, Graphics and Signal Processing, 2025, 4, 68-86
Published Online on August 8, 2025 by MECS Press (http://www.mecs-press.org/)
DOI: 10.5815/ijigsp.2025.04.05

An Effective Hybrid HBA-MAO for Task Scheduling with a Hybrid Fault-Tolerant Approach in Cloud Environment

Manoj Kumar Malik*
Bhagwan Mahavir University, Surat, Gujarat-395007, India
Maharaja Surajmal Institute of Technology, C-4 Janak Puri, New Delhi-110058, India
E-mail: manojmalik@msit.in
ORCID: https//orcid.org//0000-0001-9477-5592
*Corresponding Author

Vineet Kumar Goel
Bhagwan Mahavir University, Surat, Gujarat-395007, India
Email: dean.engg@bmusurat.ac.in
ORCID: https//orcid.org//0000-0002-7889-1945

Abhishek Swaroop
Bhagwan Parshuram Institute of Technology, New Delhi, Delhi-110089, India
Email: asa1971@gmail.com
ORCID: https//orcid.org//0000-0003-0159-5694

Received: 23 February, 2024; Revised: 10 June, 2024; Accepted: 11 May, 2025; Published: 08 August, 2025

Abstract: "Cloud computing" refers to internet-based computing on demand and describes an incredibly scalable technology used by working-class and non-working individuals globally. Fault-tolerant task scheduling is an essential tool used by end users and cloud suppliers. Finding the best resource for the specified input task presents a key challenge for fault-tolerant task schedulers. The studies that have already been done have attempted to address each of these complex issues independently. Still, it is tricky to optimize resources and provide fault tolerance at the same time. In this paper, an effective hybrid HBA-MAO and hybrid fault-tolerant mechanism in cloud computing are designed to appropriate task scheduling in VMs without delay and failure. Various tasks submitted by users and virtual machines are taken as input for the proposed approach. Hybrid Honey Badger Optimization Algorithm (HBA) and Mexican Axolotl Optimization (MAO) are used in this proposed for priority based optimal task scheduling. These scheduled tasks are assigned to the VM for execution. A fault-tolerant mechanism is immediately carried out if the tasks are not completed successfully. The hybrid reactive and proactive fault-tolerant mechanism is used in this proposed approach for a high level of fault tolerance. The proposed approach attains better performance, like 70 sec of response time, 13% of resource utilization and 95% success rate. This approach uses resources efficiently by reducing resource consumption, so it is the best choice for fault-tolerant aware task scheduling.

Index Terms: Mexican Axolotl Optimization (MAO); Hybrid Honey Badger Optimization Algorithm (HBA); Cloud computing

1. Introduction

The world of collaborative computing is entering a new phase with cloud computing. It is a recent technology that has grown significantly [1]. With the help of a shared pool of processing and storage resources, cloud computing customers can access these services through the internet as needed using the pay-per-use model [2]. The internet serves as the basis of cloud computing and is essential to its operation because devices and servers are connected to one another. Cloud computing can store and access data online rather than on a computer's hard disc. Cloud computing has gained great popularity since it is readily accessible, has low expenses for upkeep, and provides constant access to computing resources from anywhere [3].

It is good to run programmes on virtual resources, especially virtual machines, to increase cost effectiveness and scalability. In reality, numerous applications across many domains, including astronomy, finance, and physics, struggle with the issue of constantly expanding data and complicated computations [4]. Cloud computing can efficiently address the needs of high-performance computing for certain scientific applications. However, resource failure has emerged as a key issue for cloud computing as the complexity of large-scale systems grows. There will be at least one daily failure for a cloud comprising 10,000 extremely stable servers [5]. Additionally, roughly 55 disc drives fail annually, and servers crash at least twice. What's worse is that data centres frequently use inexpensive commodity technology due to low production costs, which raises the likelihood of resource failure [6].

Tasks are assigned to computing instances using fault tolerance scheduling, which ensures that tasks be completed on time, even in the event of hardware or software faults. Fault tolerance is one of the most important elements in guaranteeing the stability, dependability, and availability of essential functions and operating applications in the cloud environment. Task failure can occur for various reasons, including RAM overflow, power outages, server crashes, virtual machine failure, insufficient bandwidth, etc. [7]. Failure of the hardware, software, or time redundancy affects fault tolerance. With the checkpoint approach, most applications may be enhanced and optimized, but at the cost of significant latency. To handle fault tolerance, several occupations have used a combination of hardware redundancy and task repetition. The aforementioned techniques have undesired characteristics and downsides, including the need for repeated execution, which takes more time and resources, particularly more electricity. Redundancy is still desired in the IaaS cloud context since response speed is crucial [8].

League Championship Algorithm, which is the initial intelligent system to generate a synthetic championship, first employs certain idealistic or irrational methods before developing an artificial intelligence process. Results of comparison with various intelligence algorithms, such as optimization and simulation, demonstrate an optimization process that quickly reaches the global ideal [9]. In the cloud, this functionality is ready for fault tolerance scheduling and avoids local trapping. As a result, the cloud system's LCA-based work scheduling was suggested for overall optimization. Other methods have been effectively used in real-world settings, and they have also investigated the difficulties and potential of the method. The primary flaw in these approaches is that they don't simultaneously propose task scheduling algorithms for multiple users while considering VM resources [10]. VM scheduling must be considered throughout the task scheduling process to guarantee efficient task processing and deliver reliability to the system. In this research, a hybrid optimization and fault tolerance strategy is proposed to pick the right multiuser VMs in accordance with user needs and to arrange the tasks on VMs in a suitable way avoiding any lag or loss.

The following summarizes the research's entire contribution:

- An effective hybrid HBA-MAO and hybrid fault-tolerant approach in cloud computing is developed to plan the VM tasks, avoiding lag or loss correctly.
- Several virtual machines and the tasks submitted by users are taken as input for the proposed method.
- Hybrid Honey Badger Optimization Algorithm (HBA) and Mexican Axolotl Optimization (MAO) are used in this proposed approach to boost the behaviour of task arrangement.
- A hybrid approach based on the replication impact heuristic is used in the proposed approach to offer excellent stage tolerance of fault.
- A hybrid strategy can be implemented utilizing the guidelines from reactive as well as proactive approaches.

The remaining sections are arranged as follows: Section 2 presents the most recent fault-tolerant work scheduling methodology. The proposed strategy and its framework are addressed in Section 3. Experiments and results are provided in Section 4, and Section 5 concludes the work.

2. Literature Review

Effective fault-tolerant scheduling of tasks in cloud computing is being created using various methods. The following section reviews some existing fault tolerance task scheduling techniques.

Kumari and Kaur [11] suggested a replication based fault tolerance method to raise the dependability of VM-based systems. This procedure selects the actual computers on which VM clones can be installed using a process aware of the data centre's topology. The simulation outcomes show that the proposed methodology offers greater reliability than the other approaches.

Edwin et al. [12] developed an efficient multiobjective optimized replication management. Different parts of dynamic, cost-conscious data replication through optimization were suggested, identifying the minimal amount of data replication information necessary to ensure data availability raises as the replication process scales. Future applications can improve this by increasing the data availability in each data centre with high replication factors that can be combined with other fault-tolerant approaches to increase dependability.

Tamilvizhi and Parvathavarthini [13] suggested an original viewpoint on using a fault-tolerant process. To successfully reduce data availability issues brought on by network traffic in the cloud server's cloudlets, it discusses the use of cloud servers, cloud selection, health monitoring for fault detection, and migration techniques to tackle faults as

they arise. Future research will focus on fault-masking techniques that will enhance tolerance even after a defect has occurred.

Dynamic Fault Tolerant VM Migration (DFTM), a novel technology, was developed by Sivagami and Easwarakumar [14]. It was created to guarantee the resiliency of the cloud data centre infrastructure using a sophisticated system for meeting virtual network demands. For the purpose of assessing route traffic, an integer linear programming model that takes into account all pertinent numerical elements and selects the best VM via the best path was created. An alternate switch identification technique then determines the new VM migration and routing. But it wasn't centred on rollback objectives.

Gupta et al. [15] suggested a clever artificial neural network and intuitive method influenced by nature. The fault-resistant aware ANN algorithm's major goals are proactive performance optimization and reliability testing. This model works better in a massive data centre.

A powerful and flexible fault-tolerant scheduling technique to enable error-free task scheduling was suggested by Sathiyamoorthi et al. [16]. For the best QoS, it considers the most important factors, like the failure rate and the present workload of the resources. Empirical results show that the recommended technique outperforms the benchmark solutions with regard to of balance of loads and fault tolerance. The performance, though, was poor.

The energy conscious fault tolerant dynamic planning scheme (EFDTS) was created by Marahatta et al. [17] to optimize resource and energy usage via a fault-tolerant strategy. To reduce the job rejection ratio brought on by machine failure and delay, replication was utilized for fault tolerance.

According to the literature, numerous systems are designed for fault tolerance task scheduling in a cloud environment. Based on the articles mentioned above, several significant challenges arise in effective fault tolerance task scheduling. Data availability in each data centre is low [12], doesn't provide improved fault tolerance [13], doesn't focus on rollback purposes [14] and low performance [16]. Thus an effective fault tolerance task scheduling is not achieved using the existing research. So, the proposed strategy focuses on hybrid optimization for task scheduling and a hybrid fault tolerance approach for effective fault tolerance in cloud computing.

3. Proposed Methodology

The field of distributed computing has entered a new phase with cloud computing. In recent years, this technology has grown significantly. Cloud computing enables consumers to pay-per-use access to these applications via the Internet as needed, thanks to a common pool of processing and storage resources. Therefore, the idea of job scheduling is considered to be a problem that needs to be resolved in the current study activity. An NP-hard combinatorial optimization problem is task scheduling in cloud computing. It is the technique of assigning tasks to VMs that are appropriate for them at a particular period. Tasks are assigned to the chosen virtual machines during task scheduling depending on user requests and requirements. A system with inefficient scheduling has low performance, whereas a system with an effective scheduling strategy has lower costs and higher performance. The workflow of the proposed approach is illustrated in Figure 1.

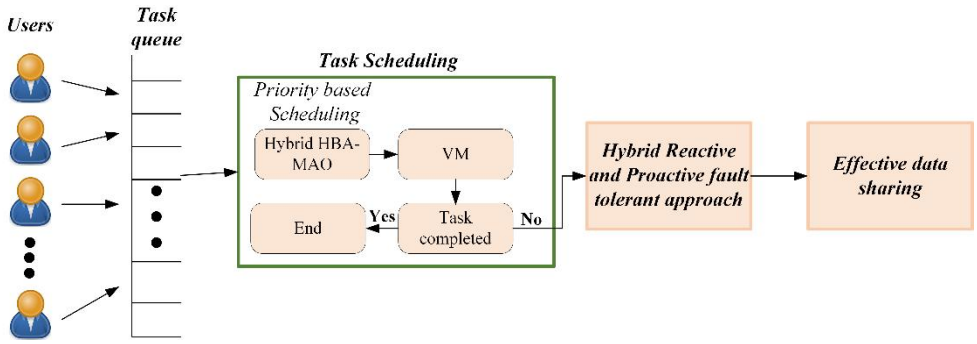

Fig. 1. Workflow of the proposed approach.

Several virtual machines and different tasks submitted by users are initially arranged for the scheduling and execution process. The tasks are initially scheduled based on its priority, such as makespan, running period and expense of the multi-cloud. Hybrid Honey Badger Optimization (HBA) and Mexican Axolotl Optimization (MAO) are used in this proposed approach to enhance the behaviour of task arranging. The best globally best practices are produced by this hybrid strategy, which combines the benefits of HBA and MAO. These scheduled tasks are assigned to the virtual machines for execution. A fault-tolerant mechanism is immediately carried out when a task is not completed successfully (i.e., task failed). This proposed technique adopts a hybrid strategy based on a replication impact heuristic to offer an elevated degree of fault tolerance. Guidelines from proactive (monitoring) and reactive (replication and resubmission) procedures can be used to create this hybrid strategy. This basic objective is to employ resources more effectively by lowering resource usage. This method accepts faults based on a heuristic estimate to decrease resource waste and boost VM efficiency.

3.1 Hybrid HBA-MAO for task scheduling

The unpredictable character of task scheduling makes it difficult to identify the best resource. Because it directly affects the income and flexibility of cloud resource providers, the makespan problem is considered here. It needs to be avoided, and the system's efficiency needs to be maximized by considering numerous factors. Three primary parts make up the suggested framework. 1. Task schedulers and users of clouds are connected through cloud brokers. 2. The job of the task scheduler is to locate the best resource for carrying out a task. 3. The data centre offers a resource to users of the cloud.

3.1.1 Honey Badger Optimization

The honey badger is a furry creature with fluffy black and white fur well-known for its fearlessness. This medium-sized dog hunts 60 different species, including deadly snakes, and is a fearless forager (60 to 77 cm in length and 7 to 13 kg in weight). It's a clever, tool-using species that loves honey. It likes to live solitary in self-made burrows and only contacts with other badgers during mating. The honey badger is divided into 12 recognized subspecies. There is no specific time to breed for honey badgers because cubs are born all year long. They are sufficiently courageous to face even greater predators when unable to escape, and they do it without hesitation. This animal also climbs trees to reach food sources like bird nests and beehives.

A honey badger locates its prey by travelling steadily and slowly while utilizing its capacity to recognize mice. Before catching the prey, it starts to locate the prey's basic position through digging. In an endeavour to obtain food, it can drill up to fifty holes in a day over a distance of at least forty kilometres. Despite being preferred by honey badgers, it is useless for locating beehives. Yet, a bird named Honey-guide can locate the hives but is unable to get honey from them. A partnership forms between the two due to these occurrences, in which the bird leads the badger to beehives and utilizes its powerful claws to aid in opening hives. Both then enjoy the results of their combined efforts [18]. The HBA offers strong exploration capabilities, rapid convergence, and adaptive search behaviour inspired by the foraging patterns of honey badgers, making it effective for identifying initial optimal task-resource mappings in dynamic environments. HBA handles the initial task allocation by exploring the search space efficiently.

3.1.2 Mexican Axolotl Optimization

The axolotl's way of life served as the inspiration for the metaheuristic algorithm known as the MAO. It draws inspiration from the axolotl's development, reproduction, and tissue repair processes and their aquatic lifestyle. The population of axolotls is split between men and females because they are sex-based animals. Also, take into account that axolotls can change their hue and that they do so to hide from predators by changing the colour of various body sections. Replication and Variety [19]. MAO algorithm enhances solution diversity and refinement through mechanisms inspired by axolotls' self-healing and gender-based evolutionary traits, which help avoid local optima and fine-tune scheduling decisions. MAO improves the solution through adaptive updates in the optimization loop. HBA-MAO combination ensures better resource utilization, reduced makespan and execution costs, and supports a fault-tolerant scheduling mechanism by dynamically reassigning failed tasks, thus achieving high success rates and reliability in cloud computing environments.

3.1.3 Steps involved in the hybrid HBA-MAO approach

In this, a heuristic algorithm based on a honey badger optimization with a Mexican axolotl optimization is presented. This proposed hybrid approach consists of four different steps such as initialization, fitness function, updation and termination. The updating portion of the HBA is replaced with the MAO approach to provide optimal solutions. Step by step process of this proposed hybrid approach is given below:

i. Initialization

Virtual machines that are dispersed around the data centre are the resources that are initialized in this first phase, together with a set of tasks.

$$T = T_1, T_2, \ldots, T_n \tag{1}$$

$$VM = VM_1, VM_2, \ldots, VM_n \tag{2}$$

ii. Objective Function

The major goal is to cut down on wait times and increase resource efficiency for suppliers of cloud services. The objective function is computed using the formula below:

Makespan: It is the entire duration of a task plan or the amount of time from the start of the tasks to the completion of processing for all of the tasks.

$$Makespan = \sum_{i=1}^{n} F_{t_i} \tag{3}$$

Where F is the task's completion time at time $t(n)$.

Execution time: The system's time spent carrying out a certain task is measured as the *completion* time of that task.

$$E_t = \frac{S_t}{P_{P_v}} \tag{4}$$

Where the task's dimension is S_t, its computational capability is P_{P_v}, and its running time is E_t on the virtual machine, VM(n).

Cost: Total *expense* incurred in planning a task.

$$Cost = \sum_{i=1}^{n} EC_{t_i,r_n} \tag{5}$$

Where EC_{t_i,r_n} denotes the expense of performing the task t_i on resource r_n.

The following is the fitness equation for the suggested algorithm:

$$Fitness\ function = Min\ (Makespan + Cost) \tag{6}$$

iii. Updation

Each iteration value is updated to find the best optimal value. Updation of each iteration solution is done by using the below expression:

$$M_{ji} = M_{ji} + M_{best,\,i} - M_{ji} * \alpha \tag{7}$$

$$F_{ji} = F_{ji} + F_{best,\,i} - F_{ji} * \alpha \tag{8}$$

iv. Termination

It is the last procedure, which happens once the best solution has been found.

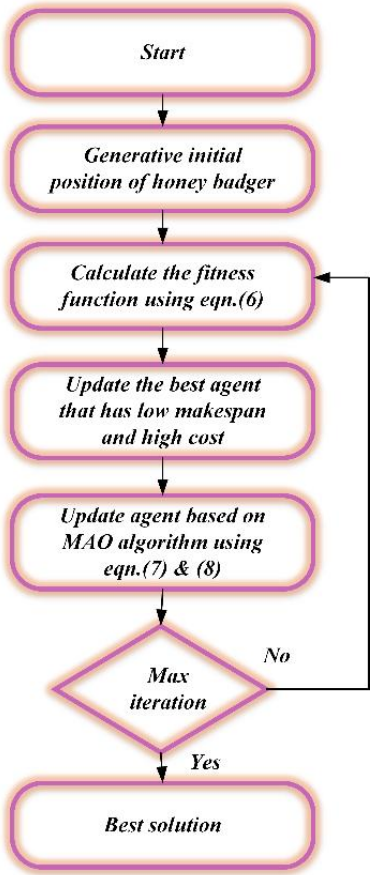

Fig. 2. Flow chart of hybrid HBA-MAO approach.

The hybrid HBA-MAO algorithm flow chart is given in Figure 2. Using this hybrid optimization method, the tasks are scheduled based on its priority and assigned to the VM for execution.

3.2. Hybrid Fault tolerant Approach

This considers hybrid architecture when proactive and reactive rules, such as replication and resubmission techniques, are implemented. This plan calls for increasing fault tolerance while minimizing resource and time loss. The three steps of this proposed architecture are pre-processing, execution, and resubmission.

A replica heuristic based on replication impact (RI), used to build a separate replica for each task, is calculated when the client delivers the task to execution. The scheduler then receives information to assign each replica to its corresponding VM. Likewise, add any changed data to the schedule buffer. The integration of the fault-tolerance mechanism with the scheduler is effectively formalized, ensuring seamless transitions between proactive and reactive strategies. The proactive phase, which involves assigning task replicas based on calculated replication impact (RI) scores to high-reliability VMs, ensures tasks are executed with minimal failure risks. If any replica fails, the system smoothly transitions to the reactive phase, where the task is resubmitted to a VM with the highest reliability score. This approach is governed by a well-defined threshold, ensuring that if the failure rate exceeds a set limit, the task undergoes retry attempts until either successful completion or a defined maximum number of retries. This structured methodology ensures high fault tolerance, efficient resource utilization, and optimized task execution, making it a reliable and robust solution for cloud computing environments. The hybrid fault-tolerant method process is shown in figure 3 below.

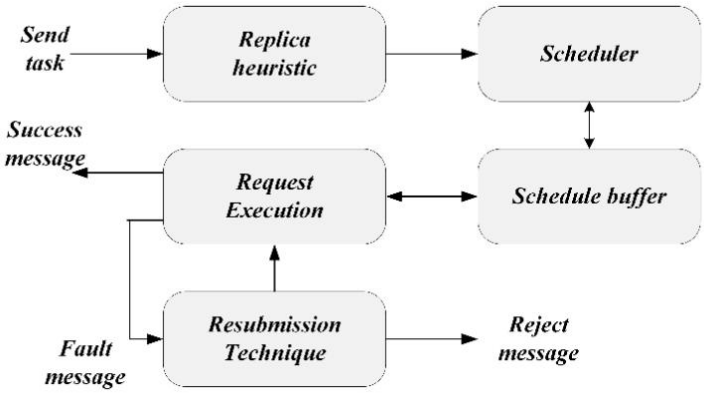

Fig. 3. Hybrid fault tolerant approach.

Numerous calculations, including those for reliability analysis, execution time, anticipated execution time, and job status, have been made during the execution phase. When every duplicate or a minimum of one of the exact same job has a pass status after request execution, the client receives a success signal. Next, the request for the execution phase of the following task in the schedule buffer is made. The execution phase does, however, communicate a fault tolerance mechanism (resubmission approach) if every duplicate of the identical task have a failed result. When a task fails, the fault tolerance algorithm selects a virtual machine (VM) with a high-reliability evaluation score and submits it for re-running. In the event that the task fails once more, the resubmitting cycle begins, and the task is uploaded another time to that exact VM. Send it again if they fail once more, but don't submit them more than once. If the client receives an alert rejecting the submission after the allotted number of attempts, the task itself may be at fault [20].

Algorithm 1: Pseudocode for proposed fault-tolerant tasks scheduling approach

Initialize the number of tasks submitted by users and the number of VM
{
VM= $VM_1, VM_2, ...VM_n$
T= $T_1, T_2, ...T_n$
# Task scheduling using hybrid HBA-MAO
{
Initialize tasks and VM as an input
Calculate fitness based on makespan and cost using eqn. (6)
Update the function based on MAO using eqn. (7)&(8)
Compute the fitness
If
{
fitness condition is satisfied
End
}
Assign the scheduled jobs to VM.
}

```
#Fault detection
{
If
{
Task completed in assigned VMs
End
}
Else
{
Fault identified
}}
#Fault tolerant mechanism
{
Hybrid proactive and reactive fault tolerant
{
 Replication
 Execution
 Resubmission
If
{
Any one of the replica executed
End
}
Else
{
Fault in task
}
}}
```

4. Results and Discussions

The environment of cloud computing is scattered. The world has been raised by its numerous aspects. These characteristics include making resources available to customers, speeding up complex calculations, and storing billions of terabytes of data on storage devices for high-performance applications. Cloud computing is the subject of criticism despite all these advantages. Utilizing approaches for fault tolerance, all of these tasks must be tolerated. A substantial amount of agreed-upon fault tolerance has been provided for accessibility and durability performance. The jobs on the VMs are suitably scheduled in this suggested strategy without any issues or delays utilizing a hybrid optimization technique and hybrid fault tolerant ways. This proposed method is simulated using Cloudsim simulation tool version 3.0,3, Eclipse IDE, Version: 2021-06 (4.20.0).

Tasks submitted by users and several virtual machines are initially arranged for execution procedure. The tasks are placed in the queue according to their priority, including their cost, makespan and expense. A hybrid HBA and MAO approach is used to improve task scheduling behaviour. The tasks are allocated to the virtual machines for further execution after scheduling. When a task is not completed successfully, the immediately fault-tolerant mechanism is carried out. A hybrid fault-tolerant approach is used in this proposed approach to provide a high level of fault tolerance. Reactive and proactive strategies may be employed to achieve this hybrid strategy. Reducing how much resource is used is the major objective in order to use resources more effectively.

Table 1. Simulation parameters of the proposed work.

Parameter	Specifications
Total no. of VM	15
Total no. of tasks	100
Cost	0.5-2
RAM	2 GB
Host MIPS	100000
Bandwidth	2000 MIPS
Architecture	X64
Task length	1000-3000 MIPS
Host parameters	6821 MIPS
CPU-intensive	50-70 tasks
I/O-intensive	10-30 tasks
Memory-intensive	10-30 tasks
Inter arrival time	1-5 sec
Short tasks	2-5 sec
Medium tasks	6-20 sec

Table 1 shows the simulation parameters used for this proposed approach. Totally 100 tasks and 15 VMs are considered for this approach. The cost assigned for each tasks, RAM, task length, bandwidth, host parameters and MIPS are given in this table.

Table 2. Simulation parameters of HBA.

Parameters	Values
Population size	30
Dimension	30
Number of iterations	1000
Lower limit	-10
Upper limit	10

Simulation parameters of the honey badger optimization algorithm are given in Table 2. Population size, dimension, number of iterations, lower limit and upper limit, are considered simulation parameters of this HBA for the proposed approach. Mexican Axolotl Optimization's simulation parameters are shown in Table 3.

Table 3. Simulation parameters of MAO.

Parameters	Values
Lambda value	0.5
Maximum number of functional evaluations	500
Damage probability	0.5
Population size	30
Regeneration probability	0.1
Tournament size	3

Table 4. Numerical analysis of the proposed method.

Input				Priority based Scheduling		Fault Detection	Replication based fault tolerance		Final Output
Tasks	Make span	Cost	VM	Sche-duled Tasks	Assigned VMs		Failed Jobs	VMs	
T0	0.82	0.91	VM0	T22	VM0	Success	T66	VM5	Success
T1	0.47	0.74	VM1	T24	VM6	Success	T53	VM10	Success
T2	0.96	0.98	VM2	T23	VM1	Success	T41	VM0	Success
T2	0.79	0.9	VM3	T34	VM2	Success	T15	VM1	Success
T4	0.92	0.96	VM4	T9	VM3	Success	T20	VM2	Success
T5	0.68	0.84	VM5	T67	VM4	Success	T81	VM3	Success
T6	0.97	0.99	VM6	T37	VM5	Success			
T7	0.77	0.88	VM7	T28	VM7	Success			
T8	0.89	0.94	VM8	T31	VM8	Success			
T9	0.45	0.72	VM9	T93	VM9	Success			
T10	0.84	0.92	VM10	T73	VM10	Success			
T11	0.47	0.74	VM11	T11	VM11	Success			
T12	0.74	0.87	VM12	T60	VM0	Success			
T13	0.99	1	VM13	T18	VM6	Success			
T14	0.74	0.87	VM14	T65	VM1	Success			
…	…	…		…		…			
T99	0.58	0.79		T13	VM1	Success			

Table 4 shows the numerical analysis of each step used in this proposed approach.100 tasks submitted by users, its corresponding makespan and cost and 15 VMs are considered for this proposed technique. These tasks are initially arranged based on its priority. Here, the T22 task comes first in the queue. Following that remaining jobs are arranged similarly. These scheduled tasks are assigned to VM for the execution process. T22 is assigned to VM0, T24 is to VM6, and all other jobs. In 100 tasks, six tasks, such as T66, T53, T41, T15, T20 and T81, failed to execute in the assigned VM. So fault-tolerant mechanism is carried out. In this proposed approach hybrid fault-tolerant mechanism is used. After the fault tolerance process, these failed tasks are executed in other VMs, such as T66 to VM5, T53 to VM10, and all other failed tasks. Finally, all tasks are executed successfully in any one of the VM.

4.1 Comparison Analysis

The proposed effective hybrid fault tolerant task scheduling (EHFTS) for appropriate task scheduling in VM without any delay and failure is compared with some existing techniques like hybrid Horse herd optimization algorithm and Reptile search algorithm (HOA-RSA), Heterogeneous Earliest Finish Time (HEFT), Checkpointing replication based on clustering heuristic (CRCH) and Maximum Effective Reduction (MER). Response time, total cost, makespan, turnaround time, waiting time, resource utilization, scheduling time, success ratio, failure rate and throughput are the performances used to compare proposed and existing techniques.

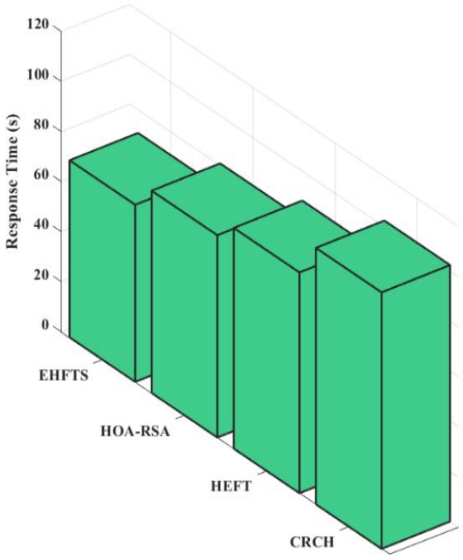

Fig. 4. Response time comparison.

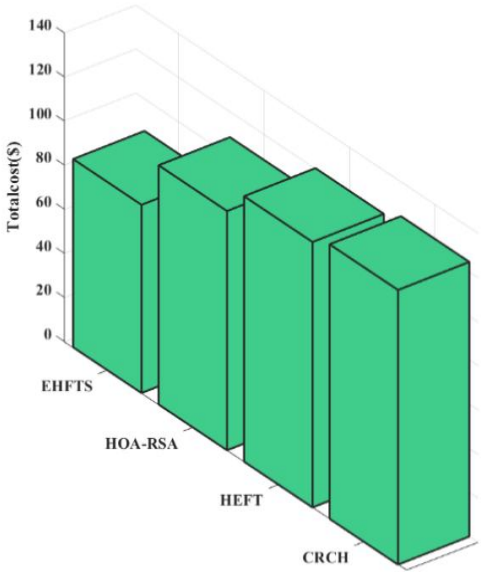

Fig. 5. Comparison of total cost.

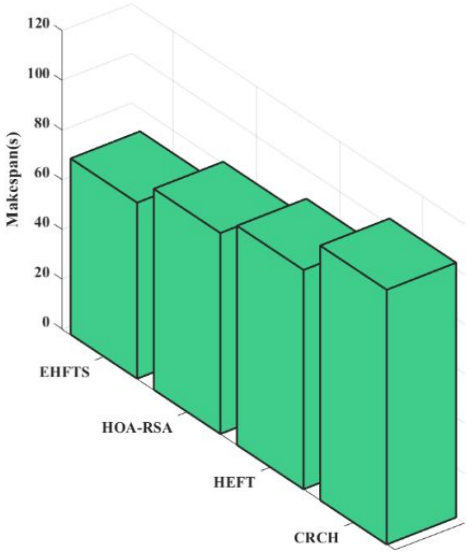

Fig. 6. Makespan metrics comparison.

Figure 4 shows the response time metrics comparison of proposed and existing approaches. The response time taken by the proposed effective hybrid fault tolerant task scheduling (EHFTS) is 70 sec. But the existing techniques, such as HOA-RSA, HEFT and CRCH, takes response time of 80 sec, 88 sec and 102 sec, respectively. It is high when compared to the proposed method. Likewise, the total cost metrics comparison of proposed and existing methods is illustrated in Figure 5. The total cost required by the proposed EHFTS is 85$. But the existing cost is 108$, 120 $ and 124$ for HOA-RSA, HEFT and CRCH techniques. It is higher than the proposed technique.

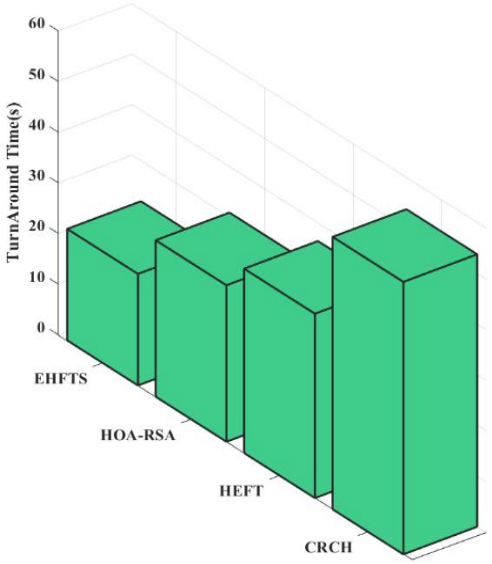

Fig. 7. Comparison of turnaround time.

The proposed and existing approach comparison in terms of makespan metrics is shown in Figure 6. The Makespan of the proposed EHFTS is 70 sec. It seems to be lower than other existing methods. Because it has a makespan of 80 sec, 88 sec and 102 sec for HOA-RSA, HEFT and CRCH, respectively. A comparison of turnaround time metrics for proposed and existing methods is depicted in Figure 7. HOA-RSA, HEFT and CRCH existing techniques have turnaround time of 30 sec, 36 sec and 53 sec, respectively. But proposed EHFTS has a 22 sec turnaround time. It is low when compared to the existing techniques.

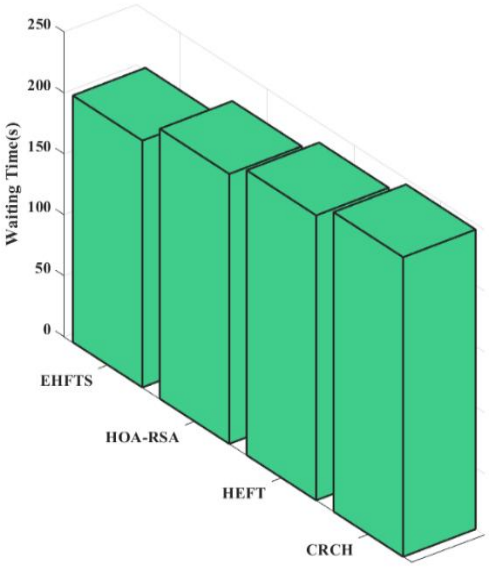

Fig. 8. Waiting time comparison.

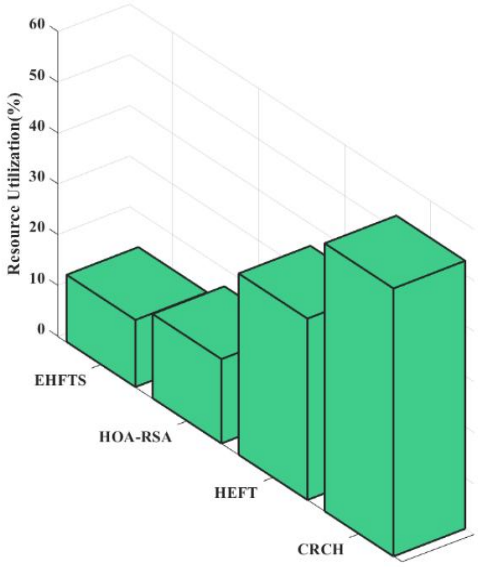

Fig. 9. Comparison of resource utilization.

Figure 8 illustrates the waiting time comparison of proposed and existing approaches. The waiting time taken by the proposed EHFTS is 70 sec. But existing techniques like HOA-RSA, HEFT, and CRCH has waiting time of 80 sec, 88 sec and 102 sec, respectively. It is high when compared to the proposed method. Similarly, the resource utilization plot for proposed and existing algorithms is shown in Figure 9. Resource utilization of the proposed EHFTS is 13%. But the existing HOA-RSA, HEFT and CRCH have resource utilization of 16%, 35% and 52%, respectively. The comparison of waiting time and resource utilization metrics indicates that EHFTS performs better than current techniques.

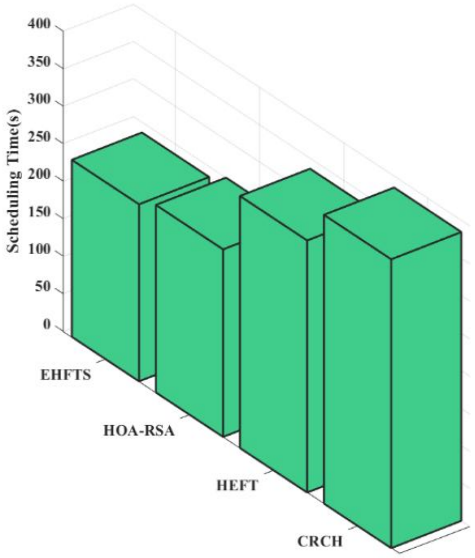

Fig. 10. Scheduling time comparison.

A comparison of scheduling time is depicted in Figure 10. The scheduling time taken by the proposed EHFTS is 236 sec. It is lower than the existing algorithms because existing algorithms have a scheduling time of 250 sec, 336 sec and 385 sec for HOA-RSA, HEFT and CRCH, respectively. Likewise, Figure 11 shows the success ratio metrics comparison of proposed and existing algorithms. 95% is the success ratio achieved by the proposed method. But existing techniques such as HOA-RSA, HEFT, and CRCH has 91%, 80% and 72%, respectively. It is lower than the proposed approach.

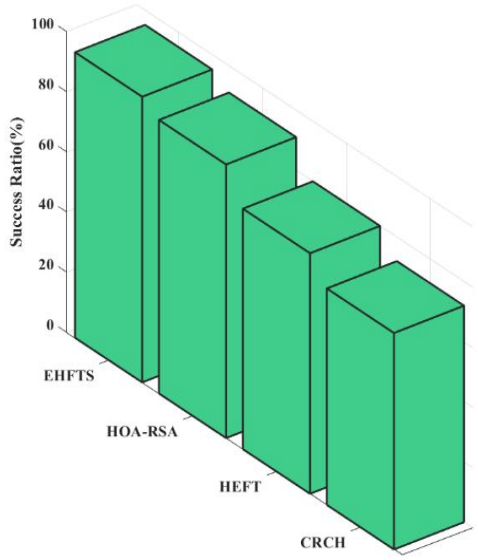

Fig. 11. Comparison of success ratio.

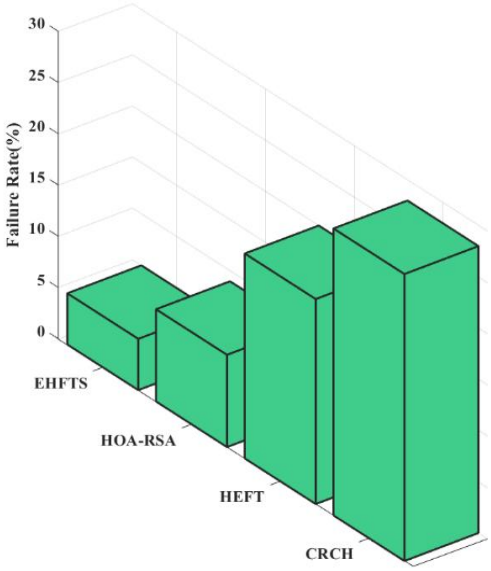

Fig. 12. Failure rate metrics comparison.

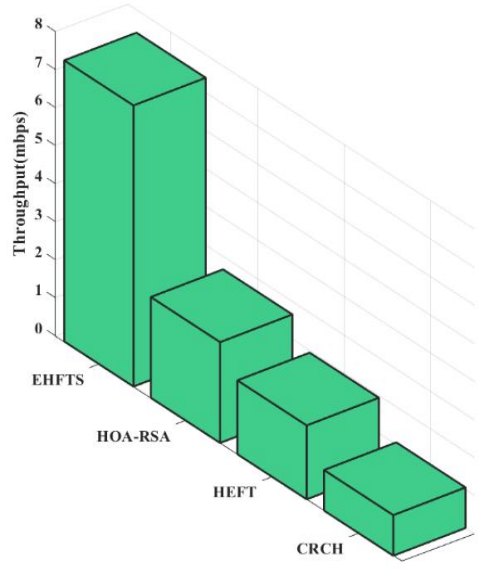

Fig. 13. Comparison of throughput metrics.

Figure 12 shows the failure rate metrics comparison of proposed and existing approaches. The failure rate of the proposed EHFTS is 5%. But existing techniques such as HOA-RSA, HEFT and CRCH have 9%, 20% and 28% failure rates, respectively. It is high when compared to the proposed method. Likewise, the throughput metrics comparison of proposed and existing methods is illustrated in Figure 13. The throughput value of the proposed algorithm is 7.3 mbps. But the existing methods have throughput values of 2.6 mbps, 1.9 mbps and 1 mbps for HOA-RSA, HEFT and CRCH techniques, respectively. It is lower than the proposed technique.

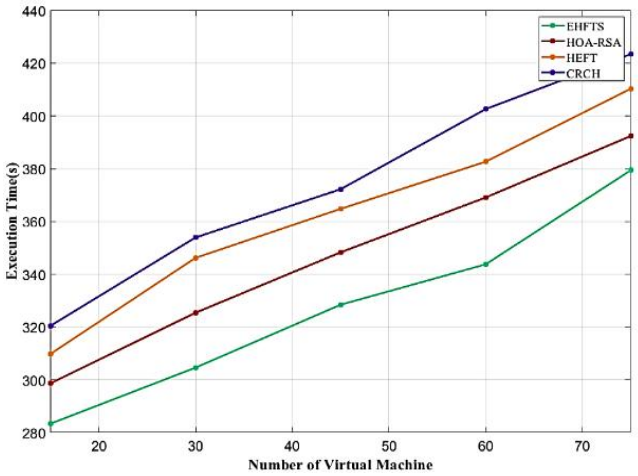

Fig. 14. Execution time comparison by varying no. of VM.

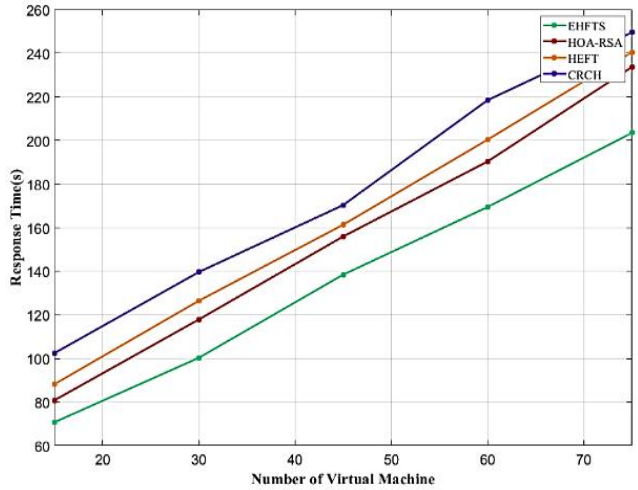

Fig. 15. Response time comparison for different no. of VM.

Execution time comparison for different quantity of VM is illustrated in Figure 14. VM's quantity is increased by 10, 20, 30, 40, 50, 60 and 70. In this proposed approach, 15 VMs are used. The execution time of the proposed approach is 250 sec for 20 VM. It increases when the number of VM increases, similarly to the existing techniques. However, the proposed algorithm takes a low execution time compared to existing techniques like HOA-RSA, HEFT and CRCH. Figure 15 shows the response time comparison of various number of VM. The response time of the proposed EHFTS is 80 sec, and existing techniques like HOA-RSA, HEFT and CRCH are 95 sec, 100 sec and 115 sec, respectively. It gets increased when the number of nodes increases. However, the proposed approach's response time is low compared to existing methods.

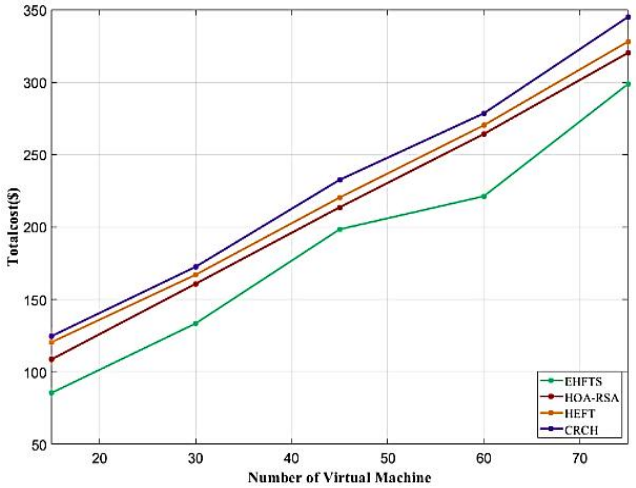

Fig. 16. Comparison of total cost by changing VM quantity.

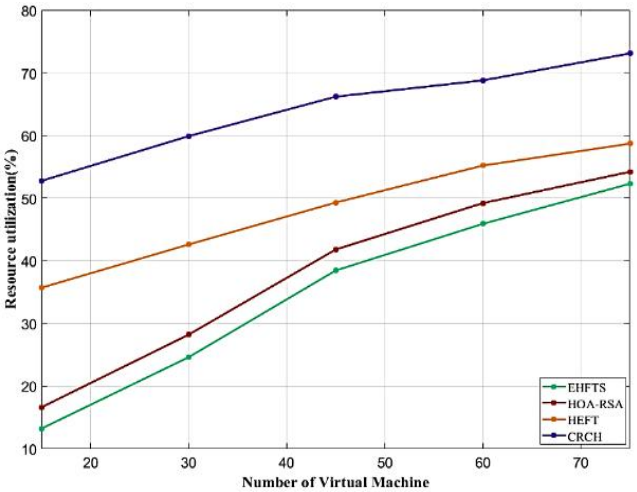

Fig. 17. Resource utilization metrics comparison by varying no. of VM.

A comparison of the total cost for different quantity of VM is depicted in Figure 16. The total cost required by the proposed EHFTS using 20 VM is 100 $. Likewise, existing HOA-RSA, HEFT and CRCH have total costs of 125 $, 140 $ and 145 $ for 20 VM. When the number of VM increases, the total cost increases for proposed and existing techniques. Resource utilization metrics comparison is shown in Figure 17. The resource utilized by the proposed EHFTS using 20 VM is 17%. Similarly, the existing HOA-RSA, HEFT and CRCH have 20%, 38% and 54% resource utilization. When the number of VM increases, resource utilization increases for both proposed and existing algorithms. However, the proposed approach is lower than other existing algorithms.

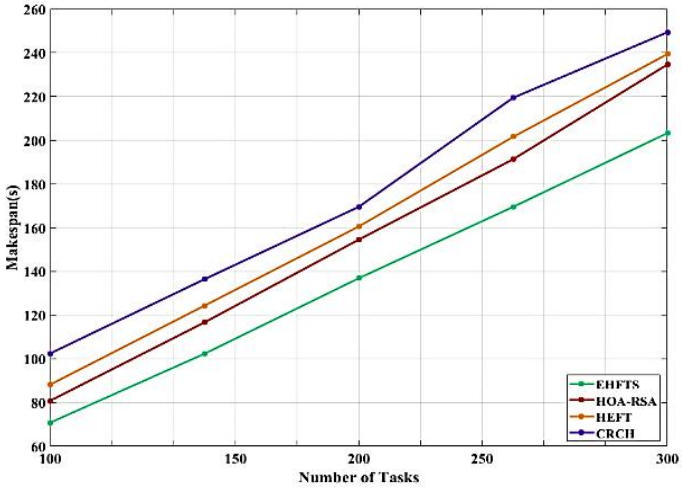

Fig. 18. Comparison of makespan metrics for various no. of tasks.

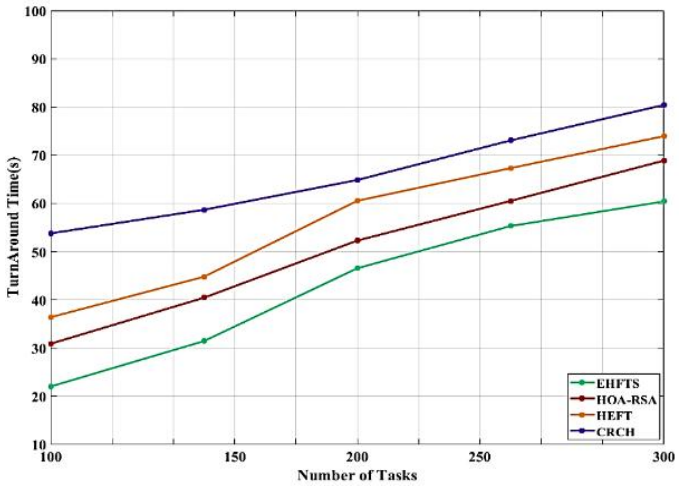

Fig. 19. Turnaround time comparison of different no. of tasks.

Makespan metrics comparison for various quantity of jobs is shown in Figure 18. Number of jobs considered for this comparison is 100, 200 and 300. The Makespan of the proposed EHFTS is 70 sec. Likewise, existing HOA-RSA, HEFT and CRCH have 80 sec, 90 sec and 102 sec, respectively, for 100 tasks. It gets increase when the number of nodes increases. But the proposed approach attains a low makespan in all scenarios. Likewise, Figure 19 shows the turnaround time metrics evaluation. The turnaround time of EHFTS is 22 sec. This get increases with task quantity increases for proposed and existing algorithms. However, the proposed algorithm takes a low turnaround time compared to existing algorithms.

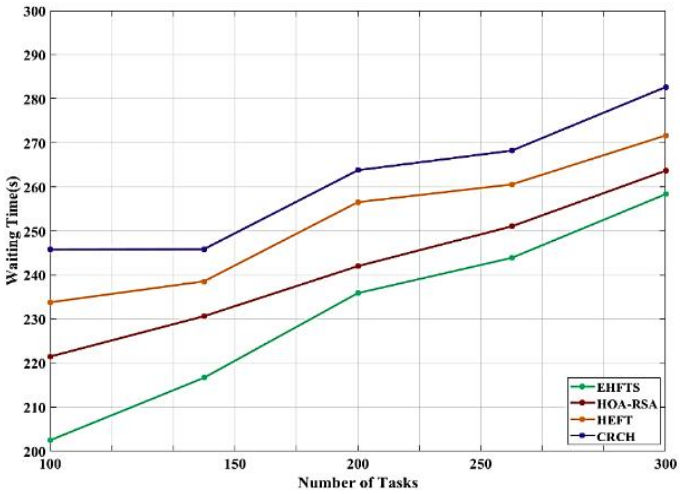

Fig. 20. Comparison of waiting time by changing no. of jobs.

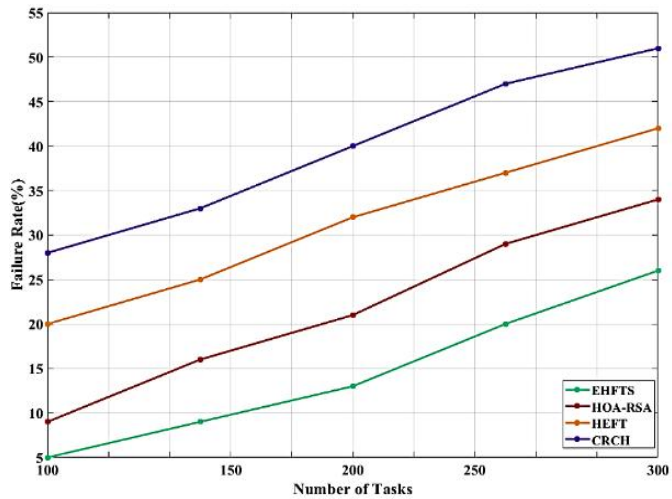

Fig. 21. Failure rate for different no. of jobs.

The evaluation of waiting time metrics in different number nodes is depicted in Figure 20. The waiting time of the proposed EHFTS is 220 sec, and existing HOA-RSA, HEFT and CRCH are 232 sec, 242 sec and 250 sec, respectively, for 150 tasks. Waiting time increased correspondingly with the number of tasks increases. It is the same for both proposed and existing algorithms. But the existing approaches takes a higher waiting time than the proposed algorithm.

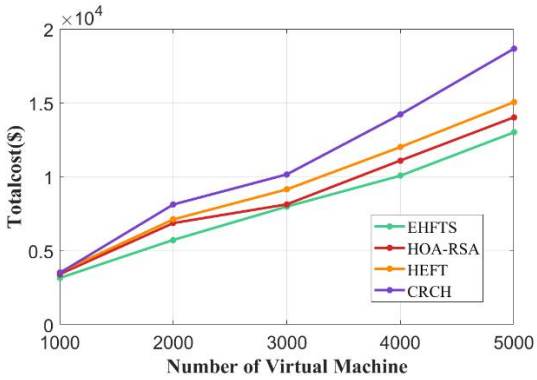

Fig. 22. Cost comparison for no. of. VMs.

Similarly, Figure 21 shows the failure rate comparison in various scenarios. The failure rate of the proposed EHFTS is 10%. Likewise, existing HOA-RSA, HEFT and CRCH have 17%, 17% and 34%, respectively, for 150 tasks. However, when the number of tasks increases, the failure rate increases for proposed and existing algorithms.

A comparison of the total cost for different quantity of VM is depicted in Figure 22. The total cost required by the proposed EHFTS using 1000 VM is 0.42×10^4 \$. Likewise, existing HOA-RSA, HEFT and CRCH have total costs of 0.38×10^4 \$, 0.40×10^4 \$ and 0.35×10^4 \$ for 1000 VM. When the number of VM increases, the total cost increases for proposed and existing techniques.

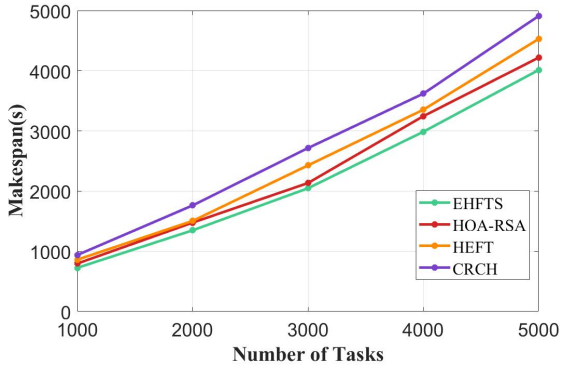

Fig. 23. Makespan comparison for varying no. of. Tasks.

Makespan metrics comparison for various quantity of jobs is shown in Figure 23. Number of tasks considered for this comparison is 1000-5000. The Makespan of the proposed EHFTS is 974 sec. Likewise, existing HOA-RSA, HEFT and CRCH have 890 sec, 920 sec and 952 sec, respectively, for 1000 tasks. The makespan increase when the number of tasks increases.

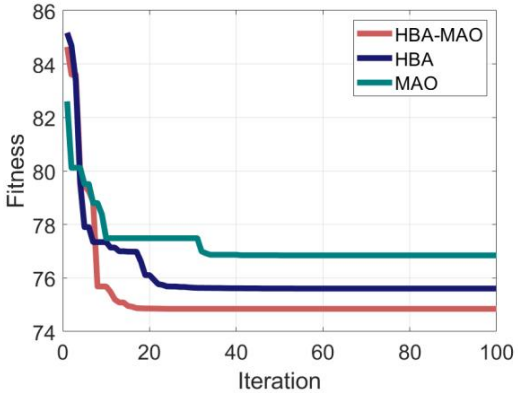

Fig. 24. Convergence graph for the proposed model.

Figure 24 shows the convergence graph for the proposed model. The fitness comparison for various iteration, the hybrid HBA-MAO, HBA and MAO methods are used. The proposed hybrid HBA-MAO perform better than HBA and MAO models.

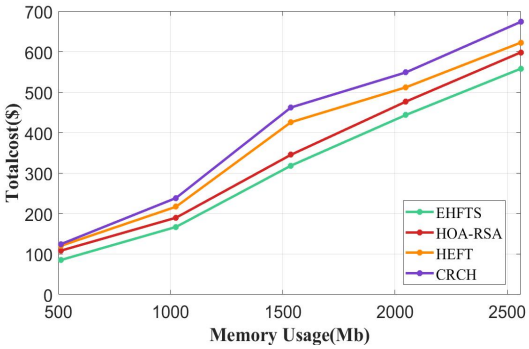

Fig. 25. Total cost comparison for different memory usage.

The evaluation of total cost in different memory usage is depicts in figure 25. The proposed EHFTS achieve 500 Mb memory in 100$ the existing methods like HOA-RSA, HEFT and CRCH achieve 120$, 138$ and 145$. The memory usage increases, the total cost increases for proposed and existing techniques.

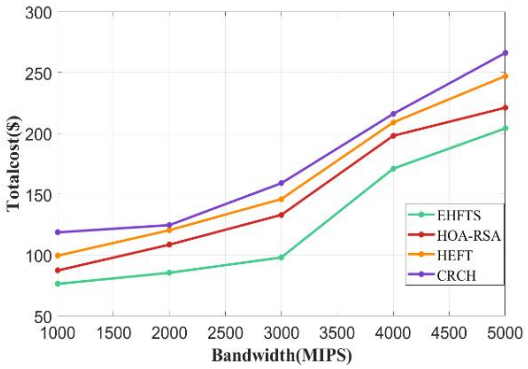

Fig. 26. Total cost comparison for different bandwidth.

The evaluation of total cost in different bandwidth is depicts in figure 25. The proposed EHFTS 1000 MIPS bandwidth achieve in 70$ the existing methods like HOA-RSA, HEFT and CRCH achieve 90$, 108$ and 125$. The bandwidth increases, the total cost increases for proposed and existing techniques.

Table 5. Statistical analysis for system performance in proposed model

Performance	F-statistic	P- value	Mean	Std	Min	Max
CPU usage	86.58	2.0201535104794102e-12	77.666667	2.516611	75	80
Memory usage	76.07	3.2149169695123357e-20	1024.666667	25.006666	1000	1050
Throughput	569.53	4.8727835099222685e-36	15.766667	0.251661	15.5	16.0
Bandwidth	469.70	5.9572770218302263e-53	102.333333	2.516611	100	105

Table 5 evaluate the statistical analysis for the proposed model. It has F-statistic, P- value, Mean, Std, Minimum, and Maximum. The CPU, memory, throughput and bandwidth parameters are used in this method. The proposed model delivers consistent and reliable performance across all evaluated parameters.

Table 6. Comparison of the proposed with various start of the art method

Methods	Makespan	Resource utilization	Throughput	Success rate
LBMPSO [21]	108.87 sec	61.5%	-	-
AdPSO [22]	5089 sec	-	1.6 tasks/sec	-
IWHOLF-TSC [23]	1054 sec	36.837%,	-	86%
HBA-MAO (Proposed)	70 sec	13%	7.3 mbps	95%

Table 6 illustrate the comparison of the proposed with various start of the art method for task scheduling algorithm. The existing methods are load balancing technique by using modified PSO (LBMPSO), Adaptive Particle Swarm Optimization (AdPSO) and Improved Wild Horse Optimization with Levy Flight Algorithm for Task Scheduling in cloud computing (IWHOLF-TSC). The proposed HBA-MAO perform better than existing methods.

5. Conclusion

In cloud computing, resources are automatically provided periodically and provided to users on demand in an open way. Failures during performing tasks are no longer unintentional; rather, they are a typical occurrence in cloud computing environments. While fault tolerance has received some attention recently, various intelligent planning strategies have been employed to address job scheduling concerns in the cloud. An effective hybrid HBA-MAO and hybrid fault tolerant strategy in the cloud was developed to arrange tasks in VM without any failure and delay. Different tasks submitted by users and several virtual machines were considered input for the proposed approach. These tasks were initially scheduled based on makespan, execution time and cost of multi cloud using hybrid honey badger optimization and Mexican axolotl optimization. Then the scheduled tasks were assigned to the virtual machines for execution. When a task is not completed successfully, the immediately fault-tolerant mechanism is carried out. A hybrid technique on the replication effect criterion was used in this proposal. Rules from reactive and proactive tactics were employed to carry out this hybrid strategy. This proposed approach was tested in the Cloudsim tool and attains better performance like 283 sec execution time, 70 sec of response time and makespan, 13% of resource utilization and 95% of success rate, 5% failure rate and 7 mbps than existing methods. Thus, using this approach, all tasks given by users are scheduled and executed without any failure. Further research may take into account flexible policies and robust methodologies.

Acknowledgement

Author contributions: The corresponding author claims the major contribution of the paper including formulation, analysis and editing. The co-authors provide guidance to verify the analysis result and manuscript editing.

Compliance with ethical standards: This article is a completely original work of its authors; it has not been published before and will not be sent to other publications until the journal's editorial board decides not to accept it for publication.

References

[1] S. A. Bello, L. O. Oyedele, O. O. Akinade, M. Bilal, J. M. Delgado, L. A. Akanbi, A. O. Ajayi, and H. A. Owolabi, "Cloud computing in construction industry: Use cases, benefits and challenges," Automation in Construction. vol. 122, pp. 103441, 2021.

[2] A. Khayer, M. S. Talukder, Y. Bao, and M. N. Hossain, "Cloud computing adoption and its impact on SMEs' performance for cloud supported operations: A dual-stage analytical approach," Technology in Society. vol. 60, pp. 101225, 2020.

[3] M. Jesi, A. Appathurai, M. Kumaran, and A. Kumar, "Load Balancing In Cloud Computing Via Mayfly Optimization Algorithm," Revue Roumaine Des Sciences Techniques — Série Électrotechnique Et Énergétique, vol. 69, no. 1, pp. 79-84, 2024.

[4] D. Saxena, I. Gupta, J. Kumar, A. K. Singh, and X. Wen, "A secure and multiobjective virtual machine placement framework for cloud data center," IEEE Systems Journal. vol. 16, no. 2, pp. 3163-74, 2021.

[5] S. Karthick, and N. Muthukumaran, "Deep Regression Network for the Single Image Super Resolution of Multimedia Text Image," 2023 IEEE 5th International Conference on Cybernetics, Cognition and Machine Learning Applications (ICCCMLA). pp. 394-399, 2023.

[6] Z. Li, V. Chang, H. Hu, H. Hu, C. Li, and J. Ge, "Real-time and dynamic fault-tolerant scheduling for scientific workflows in clouds," Information Sciences. vol. 568, pp. 13-39, 2021.

[7] B. Kada, and H. Kalla, "An efficient fault-tolerant scheduling approach with energy minimization for hard real-time embedded systems," InInternational Workshop on Distributed Computing for Emerging Smart Networks. pp. 102-117, 2019. Cham: Springer International Publishing.

[8] S. Paul, N. Chatterjee, and P. Ghosal, "A permanent fault tolerant dynamic task allocation approach for Network-on-Chip based multicore systems," Journal of Systems Architecture. vol. 97, pp. 287-303, 2019.

[9] M. Nanjappan, G. Natesan, and P. Krishnadoss, "An adaptive neuro-fuzzy inference system and black widow optimization approach for optimal resource utilization and task scheduling in a cloud environment," Wireless Personal Communications. vol. 121 no. 3, pp. 1891-1916, 2021.

[10] S. Kanwal, Z. Iqbal, F. Al-Turjman, A. Irtaza, and M. A. Khan, "Multiphase fault tolerance genetic algorithm for vm and task scheduling in datacenter," Information Processing & Management. vol. 58, no. 5, pp. 102676, 2021.

[11] P. Kumari, and P. Kaur, "Topology-aware virtual machine replication for fault tolerance in cloud computing systems," Multiagent and Grid Systems. vol. 16(2), pp. 193-206, 2020.

[12] E. B. Edwin, P. Umamaheswari, and M. R. Thanka, "An efficient and improved multi-objective optimized replication management with dynamic and cost aware strategies in cloud computing data center," Cluster Computing. vol. 22, pp. 11119-11128, 2019.

[13] T. Tamilvizhi, and B. Parvathavarthini, "A novel method for adaptive fault tolerance during load balancing in cloud computing," Cluster Computing. vol. 22, pp. 10425-10438, 2019.

[14] V. M. Sivagami, and K. S. Easwarakumar, "An improved dynamic fault tolerant management algorithm during VM migration in cloud data center," Future Generation Computer Systems. vol. 98, pp. 35-43, 2019.

[15] P. Gupta, P. Rawat, R. P. Tripathi, A. Mundra, S. Mundra, M. K. Goyal, M. Kaur, and R. Agarwal, "Nature inspired fault tolerant task allocation in cloud computing using neural network model," Journal of Intelligent & Fuzzy Systems. vol. 43, no. 2, pp. 1959-1968, 2022.

[16] V. Sathiyamoorthi, P. Keerthika, P. Suresh, Z. J. Zhang, A. P. Rao, and K. Logeswaran, "Adaptive fault tolerant resource allocation scheme for cloud computing environments," Journal of Organizational and End User Computing (JOEUC). vol. 33, no. 5, pp. 135-152, 2021.
[17] A. Marahatta, Y. Wang, F. Zhang, A. K. Sangaiah, S. K. Tyagi, and Z. Liu, "Energy-aware fault-tolerant dynamic task scheduling scheme for virtualized cloud data centers," Mobile Networks and Applications. vol. 24, pp. 1063-1077, 2019.
[18] R. El-Sehiemy, A. Shaheen, A. Ginidi, and M. Elhosseini, "A honey badger optimization for minimizing the pollutant environmental emissions-based economic dispatch model integrating combined heat and power units," Energies. vol. 15, no. 20, pp. 7603, 2022.
[19] C. S. Rao, A. Pandian, C. R. Reddy, F. Aymen, M. Alqarni, and M. M. Alharthi, "Location determination of electric vehicles parking lot with distribution system by Mexican AXOLOTL optimization and wild horse optimizer," IEEE Access. vol. 10, pp. 55408-55427, 2022.
[20] E. AbdElfattah, M. Elkawkagy, and A. El-Sisi, "Hybrid Fault Tolerance Approach Based on Replication Impact Heuristic for Cloud Computing," In2019 29th International Conference on Computer Theory and Applications (ICCTA). pp. 60-67, 2019.
[21] A. Pradhan, and S.K. Bisoy, "A novel load balancing technique for cloud computing platform based on PSO," Journal of King Saud University-Computer and Information Sciences. vol. 34, no. 7, pp. 3988-3995, 2022.
[22] S. Nabi, M. Ahmad, M. Ibrahim, and H. Hamam, "AdPSO: adaptive PSO-based task scheduling approach for cloud computing," Sensors, vol. 22, no. 3, p. 920, 2022.
[23] G. Saravanan, S. Neelakandan, P. Ezhumalai, and S. Maurya, "Improved wild horse optimization with levy flight algorithm for effective task scheduling in cloud computing," Journal of Cloud Computing, vol. 12, no. 1, p. 24, 2023.

Authors' Profiles

Manoj Kumar Malik is an Assistant Professor at Maharaja Surajmal Institute of Technology (MSIT), Delhi, and a research scholar at Bhagwan Mahavir University, Surat. With a strong inclination toward both academics and administration, he has successfully published 16 research papers in reputed national and international journals. Known for his organisational skills and leadership abilities, he actively contributes to the smooth functioning of institutional activities while mentoring students in their academic growth. His balanced approach to teaching, research, and administration reflects his commitment to holistic academic development and excellence.

Vineet Kumar Goel is currently serving as the Director-Principal of Bhagwan Mahavir College of Engineering & Technology and Dean Academics at Bhagwan Mahavir University, Surat. With over 23 years of academic and administrative experience, he holds a Ph.D. in Mechanical Engineering from NIT Kurukshetra and has contributed extensively to research in CAD, composite materials, and manufacturing technologies. Dr. Goel has published numerous papers in reputed journals, served as Editor-in-Chief of the International Journal of Technical Research, and actively promotes innovation through initiatives like the Student Startup & Innovation Policy (SSIP) in Gujarat.

Abhishek Swaroop is a dedicated faculty member in the Department of Computer Science and Engineering at Bhagwan Parshuram Institute of Technology (BPIT), Delhi. With over a decade of academic experience, he specializes in areas such as artificial intelligence, machine learning, cloud computing, and cybersecurity. Known for his student-focused teaching style and research-oriented mindset, he has guided numerous academic projects and published scholarly work in reputed journals. Prof. Swaroop is committed to blending theoretical knowledge with practical application, fostering innovation, and promoting ethical, impactful research within the academic community.

How to cite this paper: Manoj Kumar Malik, Vineet Kumar Goel, Abhishek Swaroop, "An Effective Hybrid HBA-MAO for Task Scheduling with a Hybrid Fault-Tolerant Approach in Cloud Environment", International Journal of Image, Graphics and Signal Processing(IJIGSP), Vol.17, No.4, pp. 68-86, 2025. DOI:10.5815/ijigsp.2025.04.05