EAI Endorsed Transactions
on Scalable Information Systems

Research Article EAI.EU

A New Hybrid COA-OOA Based Task Scheduling and Fuzzy Logic Approach to Increase Fault Tolerance in Cloud Computing

Manoj Kumar Malik[1,*], Dr. Vineet Goel[2] and Dr. Abhishek Swaroop[3]

[1]Assistant Professor, Bhagwan Mahavir University, Surat, Gujarat 395007, India.
[2]Professor, Bhagwan Mahavir University, Surat, Gujarat 395007, India
[3]Professor, Bhagwan Parshuram Institute of Technology, New Delhi, Delhi 110089-India.

Abstract

INTRODUCTION: Technology is made available to customers worldwide through a distributed computing architecture called cloud computing. In the cloud paradigm, there is a risk of single-point failures, in order to prevent errors and gain confidence from consumers in their cloud services, one problem facing cloud providers is efficiently scheduling tasks.
OBJECTIVES: High availability and fault tolerance must be offered to clients by these services. Fuzzy logic and hybrid COA-OOA are used in this study proposed fault-tolerant work scheduling algorithm. Jobs given by users and virtual machines are considered as input for this proposed approach.
METHODS: The given tasks are initially scheduled utilizing the FIFO order. Then, it is rescheduled utilizing the Hybrid Coati Optimization Algorithm (COA) - Osprey Optimization Algorithm (OOA) for scheduling the task based on priority.
RESULTS: This scheduled job is assigned to the VM for further execution. If the jobs are not executed successfully, then fault tolerant mechanism is carried out. Faults are recognized by employing fuzzy logic in this proposed approach.
CONCLUSION: This proposed approach attains 62 sec response time, 61 sec of makespan and 98% success rate. Thus, this proposed approach is the best choice for efficient task scheduling with fault tolerant mechanism.

Keywords: cloud computing; task scheduling; fault tolerant; COATI optimization; osprey optimization; fuzzy logic.

Received on 9 February 2024, accepted on 19 May 2024, published on 26 June 2024

doi: 10.4108/eetsis.6086

1. Introduction

The future of domestic and international computing technology rests on cloud computing. The quick growth of web-based applications and the rise of internet consumption have forced the development of cloud computing technology. In addition to its accessible expansion of grid computing, parallel, and outdated distributed, it provides consumers with pay-per-use solutions which give them access to scalable and dynamic virtual resources which they may utilize through the internet on demand [1]. Due to the accessibility of affordable and dependable resources at various times, it also offers a lifeline for many fresh start-ups. Utilizing many qualities such as heterogeneity, on-demand service, flexibility, and others, cloud computing technology meets all of a user's computing requirements [2]. Regarding job scheduling and resource allocation rules, it has the capacity to handle the volume of data. Workflow and task scheduling are two important methods in cloud computing which offer the assignment of responsibilities to the suitable resources to carry out [3].

[*]Corresponding author. Email: manojkumarmalik.bmu@gmail.com

A crucial component of cloud computing, efficient task scheduling is employed to achieve high performance in a cloud context. Workflow technology promises to provide solutions to those cloud computing problems particularly job scheduling and fault tolerance. Researchers are pursuing this approach to increase the cloud environment's efficiency [4]. The fraction of task requests sent to the virtual machine (VM) is chosen according to their runtime context using task scheduling. Several methods have been explored to increase the dependability of cloud computing. In order to provide fault tolerance, fault tolerant scheduling is developed, which distributes multiple copies of each work to various computer nodes. In the context of cloud computing, failures are to be anticipated [5]. The performance delivery of cloud resources is known to fluctuate. A fault-tolerant scheduling strategy that accounts for performance variances, resource fluctuations, and environmental breakdowns is crucial. Applications will inevitably experience an increasing number of component failures as they use cloud resources for extended periods of time. The execution of the tasks assigned to the failing components is impacted by task failures once they have occurred [6]. As a result, clouds need a fault-tolerant system. When a cloud scheduler exhibits fault tolerance, it means that it can protect and maintain the scheduled task delivery even in the event of cloud system failures.

One of the most important components in guaranteeing the resilience, reliability, and availability of operating applications and essential services in the cloud environment is fault tolerance. Various factors, such as RAM overload, power outages, server crashes, virtual machine failure, or a lack of bandwidth, might cause tasks to fail [7]. Redundancy in time or failure of hardware and software are also factors in fault tolerance. The majority of programs can be enhanced and optimized with the aid of the checkpoint technique, but at the cost of considerable latency [8]. Several jobs have employed a range of tactics, such as task recurrence and hardware redundancy, for the management of fault occurrence, eliminate delay, and keep stagnant entities [9]. The above-mentioned techniques have undesired characteristics and downsides, include the need for repeated execution, which takes more time and resources, particularly more electricity. Redundancy is still favoured in the infrastructure as a service cloud computing context since response speed is so crucial [10].

The most sophisticated algorithm for creating fabricated championships is the League Championship Algorithm (LCA) [11]. It employs a number of unrealistic or idealistic methods prior to developing the model-based computational intelligence algorithm (CIA) [12]. A comparison using state-of-the-art intelligence algorithms and simulation outcomes demonstrates an optimization strategy which quickly congregates global optimal. The functionality is prepared for scheduling with failure tolerance in the cloud computing environment and avoids localized trapping [13]. As a result, the cloud system developed a job scheduling method based on LCA for overall optimization [14]. Various fault-tolerant strategies have also been effectively implemented in real-world contexts these studies also examined the challenges and possibilities of the method. The primary flaw in all of these methods is that they do not simultaneously take into account the VM, SLA, and VM scheduling resources when outlining their assignment algorithm for multiuser scheduling [15]. SLA and VM scheduling must therefore be taken into account throughout the task-scheduling process to guarantee effective task processing and network stability. In this approach, hybrid task scheduling and fuzzy logic approach for increasing fault tolerance are designed.

The main contributions of this work are given below:

- A novel hybrid COA-OOA based task scheduling and fuzzy logic the design of cloud computing is a strategy to improve fault tolerance.
- Virtual machines and tasks given through clients are considered as input for this proposed approach.
- Hybrid Coati Optimization Algorithm (COA) and Osprey Optimization Algorithm (OOA) are utilized for scheduling the task in a queue.
- The fuzzy logic algorithm is designed using fuzzy rules is an expert-based fault detection system built on a cloud infrastructure.
- Fuzzy system is also utilized in this proposed approach in order to produce a suitable reaction that will raise the fault tolerance.

The remaining portions of the paper are arranged as follows: Section 2 reviews the various fault tolerant task scheduling algorithms. The proposed methodology and its architectures are described in Section 3. Experiments and results obtained using this methodology are provided in Section 4. Finally Section 5 concludes the entire work.

2. Literature review

In cloud computing, consumers receive transparent, automatically allocated resources whenever they need them. Failures during task execution are no longer unintentional; rather, they are a typical occurrence in cloud computing environments. Numerous techniques are developed for fault tolerant task scheduling. In that, few approaches are reviewed.

Malik et al. [16] recommended a unique Hybrid Grey Wolf and Ant Lion Model (HGW-ALM) with a lively standby replication (LSR) technique to improve the cloud computing paradigm. Additionally, the HGW-ALM model forecasts a host, and the selected host continues to operate utilizing the LSR approach if any of the hosts' capacity is insufficient for their task. The tolerant mechanism also effectively processes the checkpoint strategy. By reaching a high throughput of 6000 bps, this method produced improved results. However, this strategy has a long processing time.

A cooperative scheduling approach based on task admission and delay evaluation (CSADE) was created by Zuo et al. [17]. In order to schedule as many operations as possible to the edge while preserving QoS, a unique dynamic delay model in the source management advised in order to precisely anticipate the normal delay. The fault-tolerant system was activated and the scheduling approach was adjusted in response to resource node failures with emergency job scheduling. This CSADE can dramatically decrease the rate of QoS violations and average delay time. But this tactic doesn't yield better outcomes.

A novel Ant Colony Optimization (ACO) by familiarizing Reinforcement Learning (RL) was designed by Nalini and Khilar [18]. It was developed in combination with fault tolerance to achieve a short make-span and make the process of scheduling resistant to errors. When compared to ACO adoption alone, this strategy produces results that are around 60% better. But the system's efficiency was poor.

Saxena et al. [19] developed a Fault Tolerant Elastic Resource Management (FT-ERM). This method uses high-availability in servers and virtual machines to approach the problem discussed above from a different angle. Through experimentation with two real-world datasets, this framework was evaluated and contrasted with the most advanced. When compared to not using FT-ERM, VM migration is scaled back by 88.6% and service availability is increased by 34.47% respectively. However, this strategy took security into account.

Self-adaptive learning differential evolution based optimal PM selection method for fault tolerance (SALDEFT) was suggested by Karthikeyan et al. [20]. To raise the system's performance, robustness and effectiveness, the SALDEFT technique also includes five distinct mutation updating strategies. Subsequently, this strategies effectiveness and system performance were evaluated using fifteen benchmark functions. However, this strategy was seen to have a disadvantage in that the system's complexity rate increased. For a QFWMS study, Montage and Cybershake, two real-time scientific procedures, were taken into consideration. The QFWMS cuts cost by 6.19% and make-span by 8.86%. An energy-efficient fault-tolerant based scheduling architecture, however, was not taken into account in this case.

Salil Bharany et al, [21] studied different fault tolerance mechanism used for improving reliability and throughput in cloud computing. Also, this study analysed the performance of various machine learning and deep learning algorithm used for fault detection along with its challenges. Salil Bharany et al, [22] analysed the impact of higher energy consumption within cloud data centers and its influence on cloud environment. Techniques included in this study were VM Virtualization and Consolidation, Power-aware, Bio-inspired methods, Thermal-management techniques, and an effort to evaluate the cloud data center’s role in reducing energy consumption and CO2 footprints.

According to the literature, numerous systems are designed for fault-tolerant task scheduling in cloud computing environments. Based on the above-mentioned articles several significant problems are arises in effective fault tolerance task scheduling. High processing time [16], doesn't produce better outcomes [17], system performance was low [18], security was not considered [19], high complexity rate [20] and energy efficiency was considered [21]. In order to overcome these issues, hybrid optimization based task scheduling and fuzzy logic based fault tolerance approaches are developed in a cloud computing environment.

3. Proposed methodology

The term "cloud computing" refers to a distributed computer architecture that offers services to users globally. Exceptional fault tolerance and availability are required when providing these services to customers, but the cloud paradigm still leaves room for single-point failures. Because of this, one difficulty that cloud providers have is efficiently planning jobs to avoid errors and gain users trust in their cloud services. The FIFO merely queues processes based on the order in which they arrive in the ready queue; the first planned task will be carried out first. For work scheduling and fault tolerance solutions, a hybrid optimization methodology and fault-based methodologies are devised. The workflow of the proposed approach is given in Figure 1.

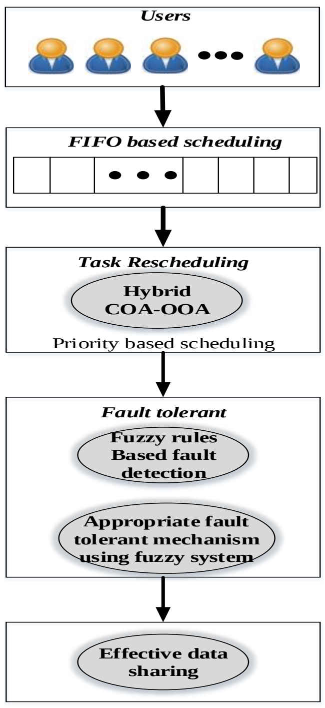
Users
FIFO based scheduling
Task Rescheduling
Hybrid COA-OOA
Priority based scheduling
Fault tolerant
Fuzzy rules Based fault detection
Appropriate fault tolerant mechanism using fuzzy system
Effective data sharing

Figure 1. Proposed Approach Workflow.

Several virtual machines (VM) and the various tasks submitted are first set up for the purpose of task scheduling and execution by various users. These submitted tasks are initially arranged in a queue based on the First in First Out order (FIFO). The tasks in a queue are then rearranged utilizing hybrid Coati Optimization (COA) -Osprey Optimization Algorithm (OOA) techniques to enhance task scheduling behaviour by taking into account variables like makes pan, cloud's execution time and cost. This hybrid technique uses the behaviour of both optimizations. In contrast to the OOA, which offers the best global optimum solutions, the COA offers the best local optimal solutions. A fuzzy logic approach is then used to identify the issues. The fuzzy fault detection system discovers any existing defects by abiding by the criteria. Following the identified issue with the necessary specifications, the fault tolerance mechanism then reacts properly to balance the system load. An expert developed the proposed rules based on a genuine cloud system and fuzzy rules for a fuzzy fault detection system. Three fuzzy values none, risk, and danger will be used as the system's output to decide whether or not a failure will occur in a physical node. Using a fuzzy system, an appropriate reaction will be generated to raise the tolerance against the fault after determining the system's status and locating the issue's source.

The n number of job requests $J_1, J_2, J_3, \dots J_n$ are the upcomings of multiple cloud users. These tasks need m number of computational resources $R_1, R_2, R_3, \dots, R_m$ in the form of virtual machines $vm_1, vm_2, vm_3, \dots, vm_m$ with various capabilities for processing task requests. With the goal of processing user applications fast and maximizing revenue for service providers through the effective use of computational resources, the best resources are chosen for end users based on their requirements and budget.

3.1. Hybrid COA-OOA for Task Scheduling

Initial FIFO order is used to group the user's submitted jobs in a queue. Then, employing a hybrid optimization technique, these jobs are rescheduled for improved results. The suggested framework is divided into three sections. Through cloud brokers, job schedulers and cloud users can communicate. The task scheduler's duty is to identify the ideal resource for carrying out a task. Users of the cloud can access resources at the data centre.

3.1.1. Coati Optimization

A COA is a recently proposed meta-heuristics method that mimics the traits of coatis that exist in nature. The fundamental goal of COA is to mimic two crucial coati behaviours: their approach when pursuing and hunting iguanas and their ability to avoid being pursued by hunters. Coatis, sometimes known as coatimundis, are omnivorous mammals that eat both small vertebrates and invertebrates as prey. Especially noteworthy is that the green iguana plays a big role in the coati's diet. Because they are arboreal animals, coatis usually forage for inguanas in trees and frequently hunt in packs. It's possible that the coati's hunting tactic entails some members of the group climbing trees to scare the iguana into leaping to the ground while others quickly attack it. Coatis are susceptible to attacks from hunters and huge raptors despite their successful predation strategies. The COA algorithm seeks to replicate the behaviours of coatis [23].

3.1.2. Osprey Optimization

Fish-eating raptors that are nocturnal and have a vast geographic range are the osprey, often known as the sea hawk, river hawk, or fish hawk. With a wingspan of 127–180 cm, ospreys weigh between 0.9 and 2.1 kg and measure 50–66 cm in length. Below are the osprey's physical characteristics:

- The upperparts are a rich glossy brown, the underpants are in pure white, and there are irregular brown streaks on a white breast.
- The head is white, but the mask which surrounds the eyes and goes to the corners along the neck is black.
- The colour of the translucent nictitating membranes is a light blue, while the irises on the eyes fluctuate in colour from golden to brown.
- The beak is black with a blue cere, while the feet are white with black claws.
- Ospreys have tiny tails and extended, slender wings.

Being a piscivorous bird, fish makes up around 99% of the osprey's diet. It often catches live fish that are between 25 and 35 cm long and 150 to 300 g in weight. Though, this captures fishes between 2 kg to 50 g in weight. Osprey can spot underwater objects thanks to their keen vision. The whereabouts of the fish underwater is discovered by the osprey when it is flying between 10 and 40 meters over the surface of the water. Subsequently, this moves towards fish, places its foot in the water and dives in to get it. After catching a meal, the osprey moves it to a nearby rock and starts to consume it [24]. The strategy of ospreys for fishing also transporting its catch to an appropriate area for consumption is a brilliant natural behaviour that can provide inspiration for developing a novel optimisation algorithm.

3.1.3. Steps involved in the Hybrid COA-OOA approach

This article presents a heuristic technique built on a Coati optimization and an Osprey optimization. The four steps in the proposed hybrid technique are initiation, fitness function, updating, and termination. In order to offer the best option, the updating component of the COA is replaced with the OOA method. Step by step process of this proposed hybrid approach is given below:

Step 1: Initialization

Various 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.

$$J = J_1, J_2, \dots, J_n \quad (1)$$

$$VM = VM_1, VM_2, \dots, VM_n \quad (2)$$

Where J indicates the number of jobs and VM indicates the number of virtual machines.

Step 2: 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.

$$Makespan = \sum_{i=1}^{n} F_{t_i} \quad (3)$$

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:

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

Step 3: Updation

Each iteration value is updated to find the best optimal value. Updation of each iteration solution is done by using both exploration and exploitation.

Ospreys are strong predators with excellent vision that enables find fish submerged. Once the have located the fish, they round it and hunt it down. The primary stage of the updating in the population of OOA depicted centred on the modelling of osprey's typical behaviour. By simulating the osprey's attack on fish, the osprey's location within the hunting area is significantly altered, which increases OOA's exploratory capacity in finding the optimal spot and avoiding local optima. Underwater fishes are the terms used in OOA design to describe the placements of other ospreys with higher objective function values for each osprey in the hunting area.

An attack on one of these fish occurs when the osprey, at random, finds its location. Applying (7) to the simulated movement of the osprey towards the fish yields an alternative position for the matched osprey. If the new position of the osprey raises the quantity of the objective function, it replaces its present position, as per (8).

$$x_{i,j}^{P1} = \begin{cases} x_{i,j}^{P1}, lb_j \le x_{i.j}^{P1} \le ub_j \\ lb_j, x_{i,j}^{P1} < lb_j \\ ub_j, x_{i,j}^{P1} > ub_j \end{cases} \tag{7}$$

$$X_i = \begin{cases} X_i^{P1}, F_i^{P1} < F_i \\ X_i,\ else \end{cases} \tag{8}$$

Where based on the initial OOA phase, X_i^{P1} represents the *i*th osprey's new position. Its objective function value is F_i^{P1}, and X_i^{P1} is its jth dimension.

An alternate position for the matching osprey is obtained by applying (7) to the osprey's simulated movement towards the fish. As stated in (8), the osprey's new location takes the place of its current location if it increases the quantity of the objective function. The OOA design first uses Eqn. (9) to establish a new random position for every member of the population that is "suitable for eating fish" in order to mimic this natural activity of ospreys. Then, if the value of the goal function is increased in this new site, it replaces the previous placement of the related osprey in accordance with Eqn. (10).

$$x_{i,j}^{P2} = \begin{cases} x_{i,j}^{P2}, lb_j \le x_{i.j}^{P1} \le ub_j \\ lb_j, x_{i,j}^{P1} < lb_j \\ ub_j, x_{i,j}^{P1} > ub_j \end{cases} \tag{9}$$

$$X_i = \begin{cases} X_i^{P2}, F_i^{P2} < F_i \\ X_i, else \end{cases} \tag{10}$$

Where, X_i^{P2} is the new position of the *ith* osprey based on the second phase of OOA, $x_{i,j}^{P2}$ is its *jth* dimension. F_i^{P2} is its objective function value.

Step 4: Termination

Once the best, most ideal answer has been found, the procedure is terminated. If the best is chosen, the process is complete; otherwise, proceed to step 2 again. Using this hybrid approach, the tasks are scheduled in the queue. The flow chart for the hybrid COA-OOA algorithm is given in Figure 2.

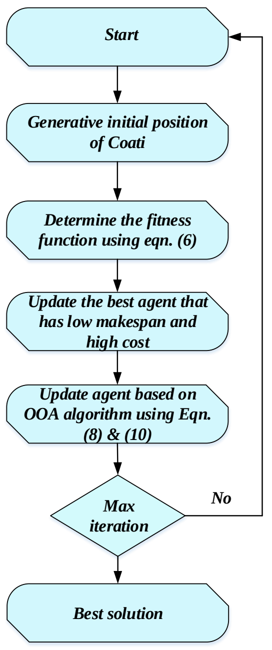
Start
Generative initial position of Coati
Determine the fitness function using eqn. (6)
Update the best agent that has low makespan and high cost
Update agent based on OOA algorithm using Eqn. (8) & (10)
Max iteration
No
Best solution

Figure 2. Flow chart of hybrid COA-OOA algorithm.

3.2. Fuzzy based Fault Tolerant Approach

Considering the variability of faults also failures in cloud computing, can lead to a virtual machine malfunction, which stops the machine from working. It's possible that a runtime fault will prevent a virtual machine (VM) from responding for running instructions for an undertaking or that a user request failure will cause a job to fail to execute. Time checking, migration, and task reprinting techniques used for rising mistake acceptability in the current study.

3.2.1. The Fuzzy Fault Detection System

A defect can be detected by the suggested fuzzy system based on the physical node's input parameters. The following are the system's input parameters.

- *Response time:* The response time average for all VMs on a physical node will be used to determine this metric.
- *Load density:* The ratio of assigned duties to all tasks is the value for this physical node attribute.
- *Throughput:* Million instructions per second (MIPS) is a measure of a virtual machine's processing power.

A fuzzy inference mechanism is employed in identifying the occurrence of fault, and each physical node's parameter values will be individually computed in accordance with the previously mentioned requirements. For better analysis, both the fuzzy models' input parameters for fault tolerance and fault detection in this work are normalized. By dividing the largest value, normalization is accomplished. Numerous tests are run on the cloud system under test in order to determine the maximum value for each parameter.

The fault detection input parameters procedure are collected by the monitoring element of the aforementioned system. The fuzzification component, which is accountable for fuzzifying the values of each input parameter in accordance with the identified fuzzy sets, must be used in order to use the inference system. The fuzzy system's output, which shows how reliable a certain node is translated into real numbers using the de-fuzzify component. This system's output, which is defined by three fuzzy values labelled "None," "Risk," and "Danger," determines whether or not a problem occurs in a physical node. In the context of the development of the suggested system for the independently physical evaluation of physical nodes, the fuzzy detection of the faults system's output reveals the cause of the error. If a failure occurs, the inputs required to produce the response as well as the result generated by this system will be received by the fault-tolerant expanding fuzzy system [25].

The single way to calculate the reaction time criterion will be discussed below because of the reasoning to determine which input parameters are comparable. The technique of determining how many more parameters to fuzzy is comparable to calculating the reaction time criterion. Three "Low," "Middle," and "High" fuzzy sets are used to describe the fuzzy value of reaction time. Response time is a fuzzy variable that can be represented as a rectangular function of membership along threshold values r, s, as well as t. The input variable rt indicates the result of this function. Eqns. (11) to (13), which specify the value of this criterion's degrees of membership function, are used.

$$\mu_{Low}(rt) = \begin{cases} 0 & if\ rt \geq s \\ \frac{s-rt}{s-r} & if\ r \leq rt < s \\ 1 & if\ rt < r \end{cases} \tag{11}$$

$$\mu_{Mid}(rt) = \begin{cases} 0 & if\ rt < r \\ \frac{rt-r}{s-r} & if\ r \leq rt < s \\ \frac{t-rt}{t-s} & if\ s \leq rt < t \\ 0 & if\ rt \geq t \end{cases} \tag{12}$$

$$\mu_{High}(rt) = \begin{cases} 0 & if\ rt < s \\ \frac{rt-s}{t-s} & if\ s \leq rt < t \\ 1 & if\ rt \geq t \end{cases} \tag{13}$$

For each output variable, there is a fuzzy set created during the aggregation procedure that needs to be defuzzed. Defuzzification is usually employed to isolate a precise value that accurately describes a fuzzy set. Here, local defuzzification is performed using the function $D_{centroid}$, which is provided by Eqn. (14), in conjunction with the fuzzy process used to calculate the membership functions.

$$D_{centroid}(rt) = \frac{\int_0^{\omega} rt\mu(rt)drt}{\int_0^{\omega} \mu(rt)drt} \tag{14}$$

The suggested guidelines created by a professional based on an actual cloud system for a problem detection system.

3.2.2. Increased fault tolerance through the fuzzy system

Using a fuzzy system, an appropriate reaction will be generated to raise the tolerance against the fault after assessing the system's status and locating the issue's source. The following are the system's input parameters.

State of Node: One of the three is represented by this parameter. fuzzy states "None," "Risk," or "Danger" of the physical node. Depending on the outcome of the fuzzy system's fault detection, this parameter's value will be established.

Lead time Job: Because of virtual machine collapse, runtime error, or demand error, current job execution may halt. Lead time task is the amount of time between the system's reaction time and its current time.

Re-execute unsuccessful: The quantity of ongoing work re-executions will be taken into account as the parameter's value if the fault's cause is a demand request since subsequent repetition raises system load.

VM throughput rate: When a VM fails, the total physical power of the physical node is determined without taking into account the fault machines and is then given to the fuzzy system purpose of decision-making. The ratio of the entire amount of calculations performed by the workload in the backlog compared to the total capacity all VMs that are now available is the value of this parameter.

The procedure of the arguments utilized for estimating the input variables becoming a fuzzy value is carried out as High fuzzy, Middle, and three Low sets. Use three strategies for the final product in this case: migration, timing verification, and task resubmission. The proposed rules are developed by an expert for an unclear error detection system, such as a fuzzy fault tolerance system, using an actual cloud platform. It is evident from the rules that response execution and the selection of nodes and virtual machines (VMs) for allocation and migration are carried out in a conventional way and are predicated on lower load densities. By use of a control message, this system notifies the user of the request output and deletes the active request for the scheduling mechanism. The execution buffer data for the task that failed on the VM is transferred to another VM via the migration approach at the migration output via the checkpoint, and the new VM responds in line with the data it obtains from the execution buffer. The checkpoint portion's duty is completed, and more data is transmitted to the cloud scheduling system [26].

It travels from the destructive physical node's queue to the physical node with the least amount of load, either in a single or multitasking migration mechanism. The number of migrations should be kept to a minimum because increasing the migration rate results in longer task response times and computational cost. The volume of tasks moved to the total tasks over time is the basis for calculating the migration rate. The suggested method's pseudocode is shown in Algorithm 1.

```
- Algorithm 1: Pseudocode for proposed task scheduling with fault tolerant
- Input: Number of jobs and VMs
- {
- VM=$VM_1, VM_2, \dots VM_n$
- J=$J_1, J_2, \dots J_n$
- # Task scheduling by hybrid COA-OOA
- {
-   Initialize tasks and VM
-  Calculate fitness using Eqn. (6)
-  Updation 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
- {
- Apply the Fault Tolerant Fuzzy System
- }
- End
```

4. Result and discussion

The focus of the cloud computing methodology, which is all about sharing data and computation, is on a scalable network of nodes, which includes end users, computers, data centres, and internet services. Task scheduling is a well-known combinatorial optimization issue that is essential to enhancing the efficiency of adaptable and dependable systems. Cloud-based application services handle massive amounts of data processing, such as social networking, web hosting, and content distribution. Processing a lot of data is necessary for these applications. In this paper, a hybrid COA-OOA based task scheduling and fuzzy logic for fault tolerance increase are developed. This proposed method is simulated using Cloudsim simulation tool version 3.0,3, Eclipse IDE, Version: 2021-06 (4.20.0).

Jobs given by various users and some, initially virtual computers are set up for the execution procedure. The jobs are first arranged in a queue utilizing FIFO order. These jobs are then rescheduled according to their priority, such as cost and expense. A hybrid COA-OOA approach is utilized as a task scheduling algorithm. The jobs are allocated to the VM for further processing. When tasks are not completed, fault tolerant mechanism is carried out. The rules state that the fuzzy fault detection system detects any existing faults and, based on the fault diagnosis and necessary parameters, the fault tolerance system responds appropriately to maintain system load balance. A fuzzy system will be utilised to develop a suitable reaction to enhance the tolerance towards the fault after assessing the system's status and identifying the source of the fault.

Table 1. Simulation parameters of the proposed work.

- Variables	- Specifications
- Total no. of VM	- 15
- Cost	- 0.5-2
- Total no. of tasks	- 100
- Architecture	- X64
- RAM	- 512 mb
- Host parameters	- 6821 MIPS
- Host MIPS	- 100000
- Task length	- 1000-3000 MIPS
- Bandwidth	- 2000 MIPS

Simulation parameters utilized in this proposed approach is given in Table 1. 100 number of jobs and 15 VMs are considered for this work.

Table 2. Simulation parameters of COA.

Parameters	Values
Search agent	30
Iteration	100
Upper Limit	50
Lower Limit	-50

Table 3. Simulation parameters of OOA.

Parameters	Values
Lower limit	-30
Upper Limit	30
Iteration	50
Search agent	50

Table 2 shows the modelling parameters of COA. Similarly, the simulation parameters utilized for the OOA approach is given in Table 3. Search agent, iteration, upper limit and lower limit are the simulation parameters utilized.

Scenario 1: In case of healthcare data monitoring huge number of task with varying importance level and length is simultaneously sent to cloud for processing. So, design of efficient task scheduling algorithm is essential for prioritizing the task and assigning based on its length, cost and execution time. Table 4 illustrates the task scheduling processing in healthcare applications.

Table 4. Process of Task Scheduling for Healthcare Data using proposed hybrid COA-OOA

No. of Task	T1	T2	T3	T4	T5	T6	T7
Length of task	100000	60000	5000	38000	12000	90000	76000
Makes pan	0.47	0.92	0.77	0.65	0.84	0.72	0.99
Cost	0.74	1	0.98	0.87	0.98	0.78	0.94
VM	VM6	VM4	VM1	VM3	VM1	VM5	VM2
ET	60	20	0.4	8	5	40	15
WT	4.5	2	0	0	0	3	`1.4

Performance of this healthcare oriented task scheduling algorithm is validated using some of the metric such as total cost, response time, resource utilization, success ratio and failure rate. Performance validated through varying the number of task is given in table 5.

Table 5. Performance validation of proposed model varying number of task

Number of task	**Total cost**	**Response time**	**Resource utilization**	**Failure rate**	**Success ratio**
500	72	42	5	15	40
1000	88	65	12	7	30
1500	105	77	18	20	55
2000	123	95	24	23	75
2500	137	102	43	35	24

Performance of the model get varied based on number of assigned task. Moreover resource utilization, response time and total cost is minimal for lesser number of task rather than higher one. Therefore, as number of task increases the performance get declined.

4.1. Comparison Analysis

The proposed Hybrid Task Scheduling and Fuzzy based Fault Tolerance (HTSFFT) approach is compared with existing algorithms called Effective Hybrid Fault Tolerant Task Scheduling (EHFTS), Dynamic Clustering League Algorithm (DCLCA) and Deadline Based Scheduling Algorithm (DBSA). Resource utilization, execution time, total cost, scheduling time, makespan, response time, success ratio, turnaround time, and failure rate. Waiting time and throughput are the performance statistics utilized to evaluate the proposed and existing approaches.

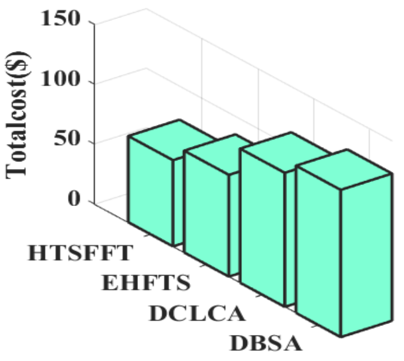

Figure 3. Total cost metrics comparison.

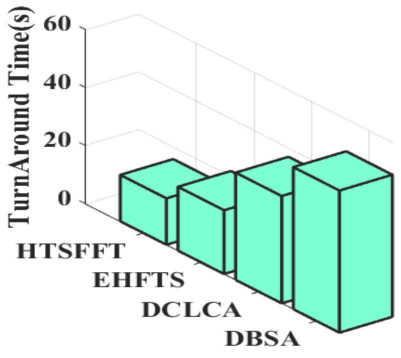

Figure 4. Analysis of Turnaround Time.

A comparison of Total cost statistic comparison is shown in Figure 3. The total cost utilized for the proposed HTSFFT algorithm is 72 $. It is low when compared to other existing approaches EHFTS, DCLCS and DBSA, which takes a total cost of 85 $, 112 $ and 123 $, respectively. Similarly, Figure 4 depicts the turnaround metric examination of various task scheduling algorithms. 16 sec, 22 sec, 37 sec and 49 sec are turnaround time taken by proposed HTSFFT and existing EHFTS, DCLCS and DBSA algorithms. This indicates that HTSFFT has a low cost and turnaround time when compared to other algorithms.

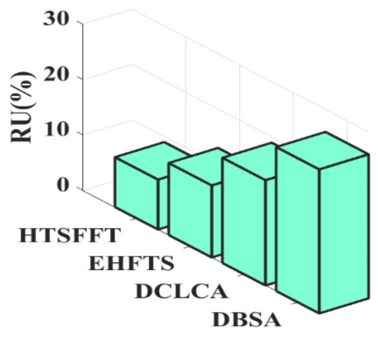

Figure 5. Resource utilization metric evaluation.

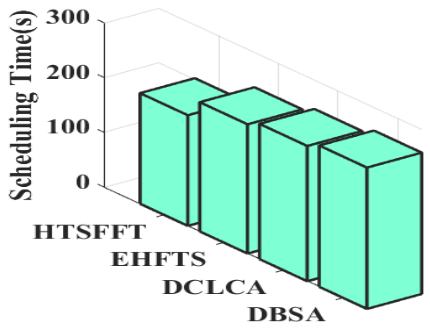

Figure 6. Examination of scheduling time.

Figure 5 depicts the resource utilization metric evaluation. Different task scheduling algorithms such as HTSFFT, EHFTS, DCLCS and DBSA has a resource utilization of 9%, 13%, 19% and 26%. This explains that the proposed approach uses low resources. Similarly, scheduling time of proposed and existing algorithms is illustrated in Figure 6. 202 sec, 236 sec, 247 sec and 258 sec are the scheduling time utilized by HTSFFT, EHFTS, DCLCS and DBSA algorithms. This indicates that the proposed HTSFFT has a low scheduling time.

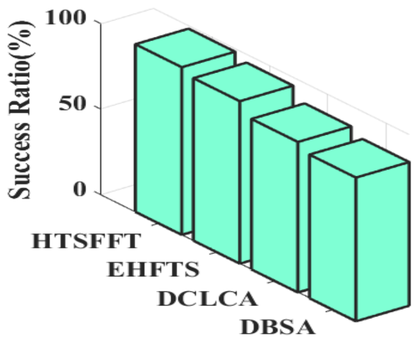

Figure 7. Success Ratio Comparison.

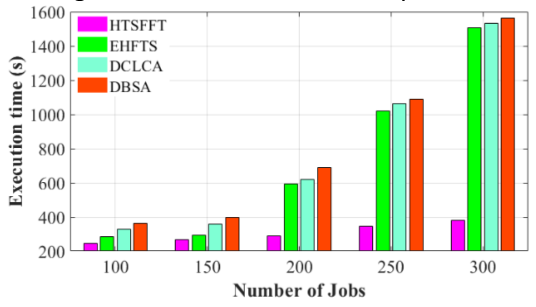

Figure 8. Analysis of Execution Time.

Figure 7 illustrates the success ratio metric analysis of proposed and existing algorithms. Various task scheduling algorithms named as HTSFFT, EHFTS, DCLCS and DBSA has a success ratio of 98%, 95%, 88% and 84%, respectively. Likewise, Execution time taken by different algorithms by changing the number of jobs is given in Figure 8. The execution time of the proposed HTSFFT is 247 sec. It is lower than other algorithms such as EHFTS, DCLCS and DBSA which has an execution time of 266 sec, 289 sec, 344 sec and 379 sec for 100 jobs. When amount of jobs increases the execution time taken by both proposed and existing algorithms are also get increased.

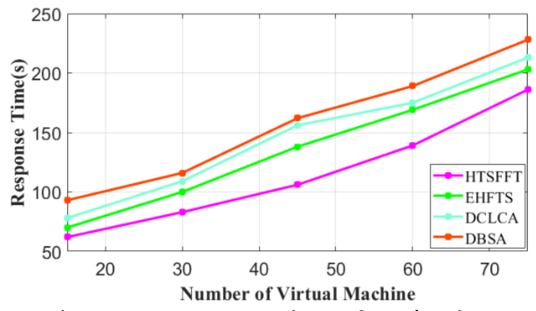

Figure 9. Response time of evaluation.

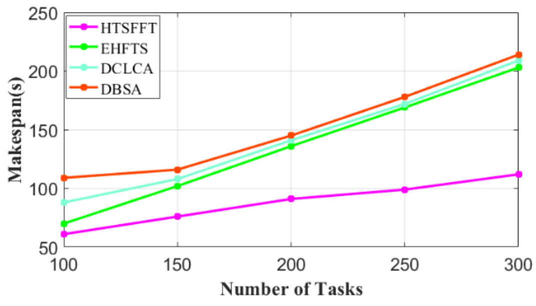

Figure 10. Examination of Makespan.

Response time comparison between the suggested and current approach are depicted in Figure 9. 62 sec, 70 sec, 78 sec and 93 sec are the response time taken by proposed HTSFFT and existing EHFTS, DCLCS and DBSA algorithms when a number of nodes considered is 100. The response time gets increased with the amount of tasks increases. Similarly, Figure 10 illustrates the Makespan evaluation of proposed and existing approaches. HTSFFT has a makespan of 61 sec, it is lower than existing EHFTS, DCLCS and DBSA algorithms which has a makespan of 70 sec, 88 sec and 109 sec, respectively.

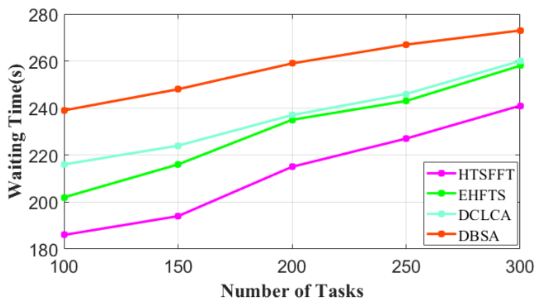

Figure 11. Waiting time metric comparison.

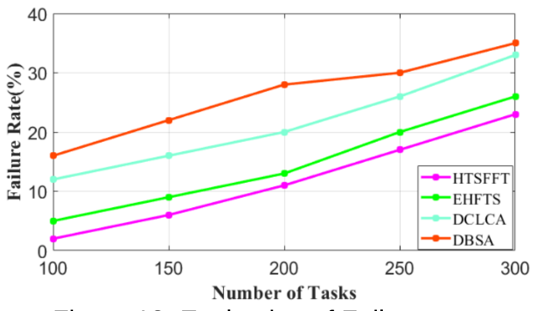

Figure 12. Evaluation of Failure rate.

Analysis of proposed and existing algorithms in terms of waiting time is shown in Figure 11. HTSFFT takes 186 sec of waiting time, while the existing EHFTS, DCLCS and DBSA takes 202 sec, 216 sec and 239 sec, respectively for 100 number of jobs. When the amount of jobs increase the waiting time will also get increased for both proposed and existing approaches. Figure 12 evaluates the failure rate of task scheduling algorithms. 2% is the failure rate of the proposed HTSFFT. But the failure rate produced by EHFTS, DCLCS and DBSA algorithms are 5%, 12% and 16% accordingly, it get increased when the number of nodes increases.

5. CONCLUSION

One of the quickly developing technologies that is affecting the IT sector and driving businesses to move their corporate infrastructures to cloud environments is cloud computing. It's challenging to schedule tasks and allocate resources in a cloud environment in a way that optimizes system performance by allocating them earliest and with the least amount of delay. In cloud computing, scheduling fault-tolerant jobs becomes an NP-hard issue because of the intricacy of the cloud, real-time task mapping with virtual machines, and virtual machines mapping with the host machine. In distributed computing structures, replication and resubmission are two of the most important and well-known multiple failure tolerance methods. Many algorithms based on replication or resubmission have been suggested over the past few years. Few of them, particularly in cloud systems, take into account these two strategies in tandem. An effective hybrid COA-OOA based fault tolerant and fuzzy logic based fault tolerance was developed. Different jobs submitted by users and several virtual machines were given as input in this proposed approach. The given jobs were initially arranged in a queue based on FIFO order. These tasks were then rescheduled based on the hybrid Coati optimization algorithm (COA)-Osprey optimization algorithm (OOA) by considering various parameters such as makespan, execution time and cost of the cloud. In this hybrid approach, the COA provides the best local solutions and OOA provides the best global solutions. The errors are then identified using the fuzzy logic approach. After determining the system's status and locating the fault's cause, a system that was fuzzy was utilised to provide a suitable reaction to increase the endurance against the fault. This proposed approach produces 62 sec response time, 61 sec of makespan and 98% success rate. Execution time achieved using proposed algorithm for 100 task is 380 sec. So, the jobs given by users were executed properly without any failure using this proposed method. Though the job are scheduled based on time and cost but the importance of job is not taken into account which is quite crucial for some application like healthcare which can be considered as future work.

Acknowledgement

Funding: The authors declare that no funds, grants, or other support were received during the preparation of this manuscript.
Conflict of Interest: The authors declared that they have no conflicts of interest to this work. We declare that we do not have any commercial or associative interest that represents a conflict of interest in connection with the work submitted.
Availability of data and material: Not applicable
Code availability: Not applicable

Author contributions: The corresponding author claims the major contribution of the paper including formulation, analysis and editing. The co-authors provides 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] Kanwal S, Iqbal Z, Al-Turjman F, Irtaza A, Khan MA. Multiphase fault tolerance genetic algorithm for vm and task scheduling in datacenter. Information Processing & Management. 2021 Sep 1;58(5):102676.

[2] Ghanavati S, Abawajy J, Izadi D. Automata-based dynamic fault tolerant task scheduling approach in fog computing. IEEE Transactions on Emerging Topics in Computing. 2020 Oct 26;10(1):488-99.

[3] Ali A, Iqbal MM, Jamil H, Qayyum F, Jabbar S, Cheikhrouhou O, Baz M, Jamil F. An efficient dynamic-decision based task scheduler for task offloading optimization and energy management in mobile cloud computing. Sensors. 2021 Jul 1;21(13):4527.

[4] Ali A, Iqbal MM, Jamil H, Akbar H, Muthanna A, Ammi M, Althobaiti MM. Multilevel central trust management approach for task scheduling on IoT-based mobile cloud computing. Sensors. 2021 Dec 24;22(1):108.

[5] Ali A, Iqbal MM. A cost and energy efficient task scheduling technique to offload microservices based applications in mobile cloud computing. IEEE Access. 2022 Apr 28;10:46633-51.

[6] Rezaeipanah A, Mojarad M, Fakhari A. Providing a new approach to increase fault tolerance in cloud computing using fuzzy logic. International Journal of Computers and Applications. 2022 Feb 1;44(2):139-47.

[7] Khaldi M, Rebbah M, Meftah B, Smail O. Fault tolerance for a scientific workflow system in a cloud computing environment. International Journal of Computers and Applications. 2020 Oct 2;42(7):705-14.

[8] Velliangiri S, Karthikeyan P, Xavier VA, Baswaraj D. Hybrid electro search with genetic algorithm for task scheduling in cloud computing. Ain Shams Engineering Journal. 2021 Mar 1;12(1):631-9.

[9] Manikandan N, Gobalakrishnan N, Pradeep K. Bee optimization based random double adaptive whale optimization model for task scheduling in cloud computing environment. Computer Communications. 2022 Apr 1;187:35-44.

[10] Malik MK, Joshi H, Swaroop A. An effective fault tolerance aware scheduling using hybrid horse herd optimisation-reptile search optimisation approach for a cloud computing environment. Cognitive Computation and Systems. 2023 Oct 10.

[11] Marahatta A, Xin Q, Chi C, Zhang F, Liu Z. PEFS: AI-driven prediction based energy-aware fault-tolerant scheduling scheme for cloud data center. IEEE Transactions on Sustainable Computing. 2020 Aug 11;6(4):655-66.

[12] Zheng, H., He, J., Huang, G., Zhang, Y., & Wang, H. (2019). Dynamic optimisation based fuzzy association rule mining method. International Journal of Machine Learning and Cybernetics, 10, 2187-2198.

[13] Liu, W. L., Gong, Y. J., Chen, W. N., Liu, Z., Wang, H., & Zhang, J. (2019). Coordinated charging scheduling of electric vehicles: A mixed-variable differential evolution approach. IEEE Transactions on Intelligent Transportation Systems, 21(12), 5094-5109.

[14] Kabir, M. E., Mahmood, A. N., Wang, H., & Mustafa, A. K. (2015). Microaggregation sorting framework for k-anonymity statistical disclosure control in cloud computing. IEEE Transactions on Cloud Computing, 8(2), 408-417.

[15] Li, Z. (2023). Exploring Significance of SPOC: A Path to Modernization of Music Cloud Computing. EAI Endorsed Transactions on Scalable Information Systems, 10(6).

[16] Malik MK, Singh A, Swaroop A. A planned scheduling process of cloud computing by an effective job allocation and fault-tolerant mechanism. Journal of Ambient Intelligence and Humanized Computing. 2022 Feb 1:1-9.

[17] Zuo L, He J, Xu Y, Zhang L. CSADE: a delay-sensitive scheduling method based on task admission and delay evaluation on edge–cloud collaboration. Cluster Computing. 2023 May 29:1-8.

[18] Nalini J, Khilar PM. Reinforced ant colony optimization for fault tolerant task allocation in cloud environments. Wireless Personal Communications. 2021 Dec;121(4):2441-59.

[19] Saxena D, Gupta I, Singh AK, Lee CN. A fault tolerant elastic resource management framework toward high availability of cloud services. IEEE Transactions on Network and Service Management. 2022 Apr 26;19(3):3048-61.

[20] Karthikeyan L, Vijayakumaran C, Chitra S, Arumugam S. Saldeft: Self-adaptive learning differential evolution based optimal physical machine selection for fault tolerance problem in cloud. Wireless Personal Communications. 2021 May;118:1453-80.

[21] Bharany, S., Badotra, S., Sharma, S., Rani, S., Alazab, M., Jhaveri, R. H., & Gadekallu, T. R. (2022). Energy efficient fault tolerance techniques in green cloud computing: A systematic survey and taxonomy. Sustainable Energy Technologies and Assessments, 53, 102613.

[22] Bharany, S., Sharma, S., Khalaf, O. I., Abdulsahib, G. M., Al Humaimeedy, A. S., Aldhyani, T. H., ... & Alkahtani, H. (2022). A systematic survey on energy-efficient techniques in sustainable cloud computing. Sustainability, 14(10), 6256.

[23] Ahmad Z, Nazir B, Umer A. A fault-tolerant workflow management system with Quality-of-Service-aware scheduling for scientific workflows in cloud computing. International Journal of Communication Systems. 2021 Jan 10;34(1):e4649.
[24] Dehghani M, Montazeri Z, Trojovská E, Trojovský P. Coati Optimization Algorithm: A new bio-inspired metaheuristic algorithm for solving optimization problems. Knowledge-Based Systems. 2023 Jan 10;259:110011.
[25] Dehghani M, Trojovský P. Osprey optimization algorithm: A new bio-inspired metaheuristic algorithm for solving engineering optimization problems. Frontiers in Mechanical Engineering. 2023 Jan 20;8:1126450.
[26] Nazari Cheraghlou M, Khademzadeh A, Haghparast M. New fuzzy-based fault tolerance evaluation framework for cloud computing. Journal of Network and Systems Management. 2019 Oct;27:930-48.