International Journal of Computer Networks and Applications (IJCNA)
DOI: 10.22247/ijcna/2022/214505 Volume 9, Issue 4, July – August (2022)

RESEARCH ARTICLE

# Multi-Objective Fault Tolerance Model for Scientific Workflow Scheduling on Cloud Computing

S. Anuradha
Department of Computer Science, Shri Sakthikailassh Women's college, Salem, Tamil Nadu, India
anuradhasphd@gmail.com

P. Kanmani
Department of Computer Science, Thiruvalluvar Government Arts College, Namakkal, Tamil Nadu, India
kanmanip2022@gmail.com

Received: 21 June 2022 / Revised: 23 July 2022 / Accepted: 03 August 2022 / Published: 30 August 2022

**Abstract – Cloud computing is used for large-scale applications. Therefore, a lot of organizations and industries are moving their data to the cloud. Nevertheless, cloud computing might have maximum failure rates because of the great number of servers and parts with a high workload. Reducing the false in scheduling is a challenging task. Hence, in this study, an efficient multi-objective fault detector strategy using an improved Squirrel Optimization Algorithm (ISOA) in cloud computing is proposed. This method can effectively reduce energy consumption, makespan, and total cost, while also tolerating errors when planning scientific workflows. To increase the detection accuracy of failures, the Active Fault Tolerance Mechanism (PFTM) is used. Similarly, the reactive fault tolerance mechanism (RFTM) is used for processor failures. The efficiency of the proposed approach is analysed based on various measurements and performance compared to other approaches.**

**Index Terms – VM Failure, Overloaded, Under Load, Squirrel Optimization Algorithm, Pro-Active Fault Tolerance, Reactive Fault Tolerance, Scheduling, Migration.**

## 1. INTRODUCTION

Scientific workflow planning and mapping have long been a problem in the research of cloud computing. These optimization problem tasks need to be planned. This decreases the expense of execution time with the necessary quality assistance boundary. Cloud computing is considered a model for empowering helpful, network admittance to a common arrangement of underlying administrations [1]. Optimal planning of virtual resources has become a top priority for cloud service providers (CSPs) because it can save millions of dollars every year [2]. Cloud computing services are divided into three types [3].It provides quick access to computer resources like networks, servers, storage, and applications. Scientific workflows are an excellent form of workflow which is used in astrology, biometrics, and gravitational waves [4,5]. The continuous form of organizing the set of computational tasks and the dependent tasks is called the workflow. Large-scale scientific workflows need versatile information and PC assets to carry out science workflows in the cloud. Work process-based applications can therefore be determined, implemented, and verified in the cloud [6,7].

Logical applications are tended to by the work interaction as a plan of exercises and datasets. There are two kinds of workflow like business workflow and scientific workflow. A business workflow is an assortment of exercises and cycles related with a business. Scientific workflows are illustrative of scientific applications that rely upon other work that was perplexing in activity.

Formalizing and organizing complex scientific cycles is worked with by scientific workflows. Moreover, it accelerates some scientific discoveries. Scientific workflows in the cloud can be specified, modified, implemented, and failure handled and monitored. Science applications can meet the need of the application by mapping it with VM images. Used in various fields: astronomy, bioinformatics, seismology, gravitational wave physics, and marine science workflow [8,9].

The use of virtual machines (VMs) has the potential to reduce inefficient resource allocation and excessive overhead. A VM can create a configuration environment that is independent of one resource and allows multiple environments to be used on the same resource. An efficient multi-objective fault detector technique based on the improved squirrel optimization algorithm (ISOA) can be used in cloud computing.

This method will effectively reduce energy consumption and overall cost while at the same time tolerating errors when planning scientific workflows. To increase the accuracy of diagnosing failures, an active fault tolerance mechanism can be used. Similarly, the reaction fault tolerance technique can be used for processor failures. Cloud computing provides an excellent space for transition between performance and cost. The proposed approach main contribution is listed here;

RESEARCH ARTICLE

- Multi-objective function based fault tolerance scientific workflow scheduling on cloud is proposed.
- To avoid processor failure, reactive fault tolerance is used. This strategy is used to assign a fresh VM to execute the workflow task.
- The proactive fault tolerance strategy is introduced to avoid overloaded VM failure. This strategy is used to migrate the data task on overloaded VM and avoid further allocation on overloaded VMs.
- For the scheduling process, ISOA is introduced. This algorithm is assign the task to VM based on a multi-objective function.
- The efficiency of the recommended approach is compared with different metrics and different metrics.

The organization of the paper is presented here, in section 2 literature survey is presented and the workflow model is given in section 3. In section 4 mathematical model of multi-objective function is presented and a clear explanation of the proposed approach is presented in section 5. The experimental results are presented in section 6 and the conclusion part is given in section 7.

## 2. LITERATURE REVIEW

A lot of researchers had developed fault tolerance-based workflow scheduling on the cloud. Some of those works are analyzed here;

Heyang Xu et al. [10] introduced cloud workflow scheduling considering error recovery in the multi-objective optimization Approach. For analysis, cloud sources consider the probability of failure during operation. The authors explored the problem of workflow planning in the clouds. The authors aimed to develop multi-objective optimization (MOF) model. In this approach, the purpose of the first and second upgrades is not only to reduce the overall completion time but also to reduce the overall processing cost. To reduce the cost, a heuristic algorithm termed Min-min-based time and cost transfer (MTCT) was developed. The efficiency of this approach was compared with existing methods.

Zulfiqar Ahmad et al. [11] introduced managing fault-tolerant and data-intensive scientific applications using cloud computing. Cluster-based, fault-tolerant, and data-intense (CFD) planning is provided in cloud environments of scientific applications. In this methodology, the information power of the errands of scientific workflows with bunch based, issue open minded components were alluded to by the CFD technique. In this approach, the montage science workflow was considered a simulation. Zhongjin Li et al. [12] introduced the Task Replication method in the fault-tolerant scheduling cloud for scientific workflow. In this approach, for scientific workflow in the context of cloud computing by fault-tolerance scheduling (FTS) algorithm was proposed. The proposed FTS instructions ensured that the task was successfully implemented in the presence of internal failure or external failure in terms of task copy. With this approach, the authors sought to reduce workflow costs with time constraints through internal and external failures. The results showed that the FTS algorithms could only ensure the successful implementation of work.

KokKonjaang and Lina Xu [13] introduced cloud computing by using a multi-objective workflow optimization strategy (MOWOS). In this approach, the authors proposed the MOWOS. The proposed approach was used to reduce administrative costs and accomplish workload tasks. The proposed algorithm had three sub algorithms such as the MaxVM selection algorithm and the MinVM selection algorithm. This algorithm was significantly increased in its performance compared to the algorithm of HSLJF and SECURE.

Yuandou Wang et al [14] introduced MOF and deep-Q-network-based multi-agent for Scheduling. In this approach, the authors proposed a multi-agent reinforcement learning system by using the deep-Q-network model. The authors considered the Markov game model. Furthermore, the decentralized DQN-based MARL framework was developed based on the decentralized DQN. The proposed DQN-based MARL framework was a combination of the traditional DQN algorithm for reinforcement learning. The authors have demonstrated that the proposed method works better than basic algorithms namely, NSGA-II, MOPSO, and GTBGA.

Prem Jacob and Pradeep [15] introduced Cuckoo particle swarm optimization for multi-objective optimal task scheduling in cloud environments. The designed MOF is based on the costs of resources and tasks. In this approach, CPSO work was done for task scheduling. This method reduced the performance costs and user costs, and it relatively minimize the make-span value of the tasks. From this approach, the proposed CPSO algorithm has reached the minimum maturity violation rate compared to other methods.

Amandeep Varma and SakshiKushal [16] introduced a scientific workflow based on hybrid MOPSO. This method deals with workflow planning problems with multiple conflicting objective functions in IaaS clouds. The proposed heuristic performance was compared with another method. The heuristic provided better coordination and regular spacing between solutions compared to others. Neeraj Arora and Rohitash Kumar Banyal [17] introduced a scientific workflow scheduling based on a hybrid HPSOGWO algorithm which is the hybridization of PSO and grey wolf optimization (GWO). The summary of the survey is given in table 1.

**RESEARCH ARTICLE**

| References | Algorithm | Proposed | Workflow Model | Findings |
|---|---|---|---|---|
| [10] | Min-min based time and cost transfer (MTCT) | This paper investigates the problem of workflow scheduling in clouds considering fault Recovery. | Montage, CyberShake, Epigenomics, and LIGO | This method does not find all types of faults. |
| [11] | fault-tolerance and data-intensive (CFD) scheduling | An efficient resource allocation on the cloud is proposed | Montage and CyberShake | This work does not consider the energy consumption of scheduling |
| [12] | fault-tolerant scheduling (FTS) algorithm | Fault-Tolerant Scheduling for Scientific Workflow with Task Replication Method in Cloud is proposed | Montage, LIGO, SIPHT, and CyberShake | They only focus on scheduling and they attained high energy consumption |
| [13] | Multi-objective workflow optimization strategy (MOWOS) | An efficient workflow scheduling is proposed | Montage, Cybershake, LIGO Inspiral, and SIPHT | This method has computation complexity |
| [14] | Deep-Q-Network-Based Multi-Agent Reinforcement Learning | Multi-objective workflow scheduling is proposed | CyberShake, Epigenomics, Inspiral, Montage, and Sipht | It takes maximum time |
| [15] | Cuckoo Particle Swarm Optimization | Multi-objective optimal task scheduling is proposed | Dynamic task | This method attained better results compared to other methods. |
| [16] | Hybrid Particle Swarm Optimization | Multi-objective workflow scheduling is proposed | Montage, EpiGenomics, cybershake, LIGO, and SIPHT | The proposed optimization algorithm easy to falls on local optimum |
| [17] | Combination of particle swarm optimization and grey wolf optimization | Efficient scientific workflow scheduling on cloud | Montage, Cybershake, Epigenomics, SIPHT, and LIGO | It takes maximum time |

Table 1 Summary of Literature Survey

### 2.1. Problem Definition

Due to the rapid advancement of the cloud, many companies and industries are using cloud applications. Cloud computing enables sharing of geographically distributed resources, which solves data-intensive and large-scale computing applications. Managing resources belonging to different organizations is complex as each organization has its own policies.

As the availability of resources varies dynamically, and due to the heterogeneity of resources, the chances of errors increase in a phased computing environment. Therefore, an efficient fault detection and fault handling mechanism is essential for a cloud computing environment. Nowadays, lot of researcher had developed task scheduling mechanism. But they not focused on faults tolerance strategy. Therefore, in this paper, an efficient fault tolerance based scheduling is proposed in this paper.

## 3. MODEL OF WORKFLOW

The structure of cloud computing contains of n number of physical machines (PM) denoted as $P = \{P_0,...,P_{i-1}, P_i\}$. Each PM is divided into some VMs denoted as $V = \{V_0,...,V_{i-1},V_i\}$ .The workflow model is generally represented by the directed acyclic graph (DAG) $WF = (T, E)$, where, $T = \{t_0, t_1,...,t_i,...,t_{n-1}\}$ is represented as available tasks and $E = \{(E_{t_i,t_j}) | t_i, t_j \in T\}$ is represented as edge.

Each task has a weight value that indicates the execution time $(t_i)$. The weights assigned to the edges express the amount of data transferred between tasks. The immediate predecessor $\Pr e(t_i)$ is calculated using equation (1).

**RESEARCH ARTICLE**

$$Pre(t_i)=\{t_j \mid (t_j,t_i)\in U\} \quad (1)$$

The immediate successor $Succ(T_i)$ is evaluated using equation (2).

$$Succ(t_i)=\{t_j \mid (t_i,t_j)\in U\} \quad (2)$$

The sample DAG is given in figure 1. In figure 1, a task with no predecessors is represented as $T_{entry}$ and a task with no successors is represented as $T_{exist}$ .

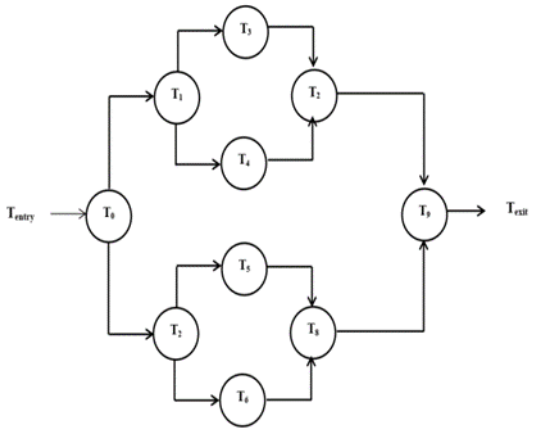


Figure 1 DAG Model

## 4. DESIGN OF MULTI-OBJECTIVE FITNESS MODEL

As the fitness function is designed to evaluate the solution also it quantitatively measures how to fit a given solution fits in solving the problem. In this paper, for fitness evaluation, the multi-objective function is presented. The single objective function is only focused on a single parameter. This affects the overall performance of the system. Therefore, the multi-objective function is designed in this paper. The proposed MOF is designed using three parameters namely, total execution cost, Makespan, and energy. From the parameter, it's understood the proposed multi-objective function has a minimization problem. The proposed fitness function is given in equation (3).

$$Fitness=\min[\alpha_1\times\cos t+\alpha_2\times makespan+\alpha_3\times Energy] \quad (3)$$

The first parameter of the objective function is cost. Total execution cost (TC) is measured based on how much cost is utilized for the scheduling process. The user wants to utilize the resources in a VM and the user has to pay the cost for the task. If the task $W_i$ is scheduled $VM_j$, the total CPU and memory cost is included in the payment. The CPU cost is calculated using equation (4).

$$C_{cost}(j)=C_{base}\cdot T_{ij}\cdot C_j \quad (4)$$

Where;

$T_{ij}$ → The execution time of the task $W_i$ on $VM_j$

$C_j$ → Overall quantity of CPU

$C_{base}$ → CPU base cost

The memory cost is calculated using equation (5)

$$M_{cost}(j)=M_{base}\cdot T_{ij}\cdot M_j \quad (5)$$

Where;

$M_j$ → Total amount of memory

$M_{base}$ → Memory base cost

The overall execution cost of all tasks is evaluated using equation (6)

$$TC=\sum_{i=1}^{N}C_{cost}(j).X_{ij}+\sum_{i=1}^{N}M_{cost}(j).X_{ij} \quad (6)$$

Where;

$X_{ij}$ → Assignment matrix

The second parameter of the fitness function is Makespan. The key parameter of task scheduling is makespan. The objective of scheduling is to minimize the total completion time for scheduling the entire task within the available resources. The makespan is calculated using equation (7);

$$Makespan=CT_{Max}[i,j]$$
$$i\in T,\ i=1,\ldots,n \qquad j\in VM,\ i=1,\ldots,k \quad (7)$$

$CT_{max}$ is calculated using equation (8)

$$CT_{\max}=\sum_{i=1}^{N}\sum_{j=1}^{k}(T_{ij}\cdot X_{ij}) \quad (8)$$

Where;

$CT_{\max}$ → Maximum completion

$n$ → Task count

$k$ → VM count

The third parameter of objective functions **is** Energy. A good scheduling system uses a small amount of energy to plan the entire task from the available resources. The total energy consumption during scheduling is given in equation (9).

$$E_{total}=E_{busy}+E_{idle} \quad (9)$$

**RESEARCH ARTICLE**

Energy consumption is calculated using Equation (10) when all resources are busy.

$$E_{busy}=\sum_{i=1}^{n} K\times V_{i,pj,s}^{2}\times f_{rj,s}\times T_{i,j}=\sum_{i=1}^{n} P_{dynamici}\times T_{i,j} \quad (10)$$

Where;

$T_{i,j}$ →Time to execute the task $n_i$ using resource $r_j$

$V_{i,pj,s}$ → Task $n_i$ scheduled to be $r_j$ based on voltage $v_s$

$f_{rj,s}$ →Frequency of resource $r_j$ having voltage level s

Dynamic power loss $(P_{dynamic})$ is evaluated using equation (11).

$$P_{dynamic} = K\times V_{j,s}^{2}\times f_s \quad (11)$$

Where;

$K$ →Constant parameter associated with dynamic power

$V_{j,s}^{2}$ →Supply voltage

$f_s$ → Relevant frequency of $V_{j,s}^{2}$

Energy consumption is calculated using Equation (12) when all resources are idle.

$$E_{idle}=\sum_{j=1}^{p} K\times V_{ilowest}^{2}\times f_{jlowest}\times T_{jidle} \quad (12)$$

Where;

$V_{jlowest}$ →Lowest voltage of resources

$f_{jlowest}$ →Lowest frequency of resources

$T_{jidle}$ → Idle period of resource $r_j$

## 5. PROPOSED MODEL

The main aim of the presented technique is to optimally schedule the task on cloud resources while tolerating the faults. An improved squirrel optimization algorithm with proactive and reactive techniques is used to create a multi-objective function to achieve this concept. The overview of the presented technique is given in Figure 2. The proposed approach contains four phases monitoring phase, analyzer phase, planner phase, and executer phase. The monitoring phase collects the task to the user and collects the information on resources. Then, the analyzer uses ANFIS to predict a load of resources based on the monitor's feedback. The predicted values are then sent to the scheduler component, and a workflow-scheduling algorithm is executed accordingly. To schedule, a multi-objective ISO algorithm is used, which combines pro-active and reactive techniques. Finally, at the executor stage, workflow tasks are assigned to the appropriate VMs.

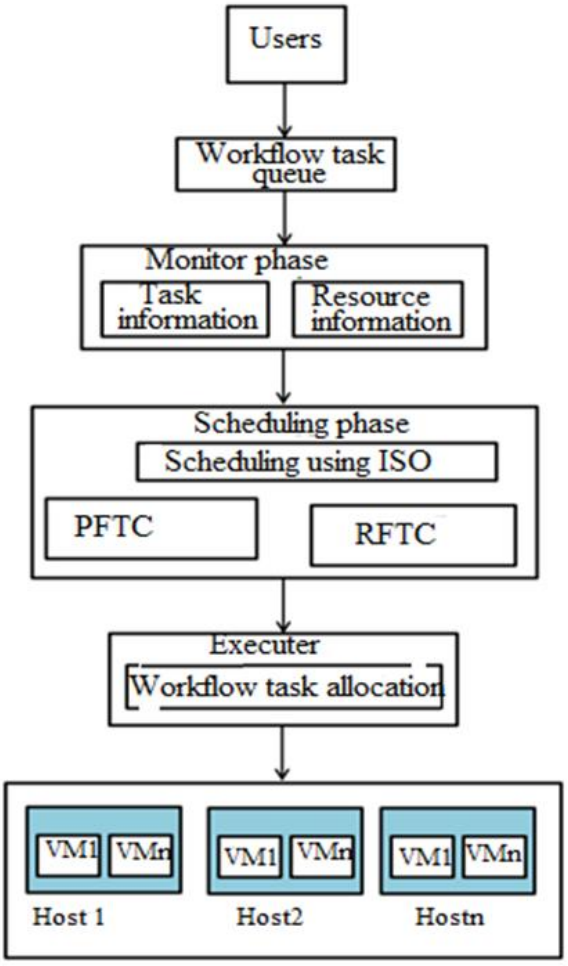



Figure 2 Overall Structure of Proposed Methodology

### 5.1. Monitor Phase

The monitor phase is to gather input information from the user and collect the resource load information from the data center. It comes with two sensors, a user sensor, and a resource sensor. The purpose of the user sensor is to collect information on the user's workflow tasks such as the size of the task, type of task and request rate, etc. The resource sensor is to gather information on the resource load rate such as the load of the CPU, network traffic, and memory. Monitoring data are stored in the knowledge base, which is used for analysis.

**RESEARCH ARTICLE**

5.2. Analyzer Phase

The analyzer phase is used to analyze and predict future VM load for error recognition in VMs. At this point, the analyzer gets the data from the monitor phase, and based on the data, the future load is predicted.

This technique is used to distribute the over-loaded VM to the under-loaded VM to avoid VM failure and increases the optimal performance. In this paper, ANFIS is used for prediction. During prediction, the time series data i.e., CPU load is trained by ANFIS. Here, two input variables and one output are used. The ANFIS description is explained below;

Step 1: Data collection: The research data are collected from the user's request. These data are considered as the time series i.e., the data are known at a point y = T to forecast the series of future values y = T + P. initially, for prediction, the data is trained based on the user data.

Step 2: for forecasting, "one-step-ahead" framework is designed. In this step, initially, we map D to a Δ shaped time series, is $(y(k-(Z-1)\Delta),\ldots,y(k-\Delta),y(k))$, to a forecasted future value $y(T+p)$. In the presented forecast model, the requested number of each layer is collected and the values D = 11 $\Delta = P = 1$ are used to estimate the number of services requested at the next time interval.

We predict $y(k)$ from the ten past values of the time series, that is, $y(k-10), y(k-9),\ldots.$, and $y(k-1)$. Therefore the format of the training data is given in (13)

$$[y(k-10),\ y(k-9),\ldots,y(k-1);\ y(k)] \quad (13)$$

Each input package is split into two sets such as training and testing.

Step 3: Create ANFIS forecast model: To generate the ANFIS model, initially, the input data are partitioned using a subtractive clustering approach. Then based on the clusters the fuzzy interface system is generated. The steps included in ANFIS are explained below;

- The universe of discourse for input variables should be partitioned as follows: Initially, the variables are splinted based on the minimum maximum method
- Set membership function for output variable: For attained the output, the Gaussian membership function is used. Consider, two-period inputs $(y_{k-1}, y_{k-2})$ and, three linguistic intervals $(i=1,2,3)$ which are grouped by using subtractive clustering in every input variable. Based on the input and output variable, the fuzzy rules are generated using equation (14).

$$If\ y(y_{k-1})=U_i\ \ and\ \ x(y_{k-2})=V_i\ \ then\ \ f_i(y_k)=u_i\,y+v_i\,x+r_i \quad (14)$$

Where;

$y(y_{k-1}), x(y_{k-2}) \rightarrow$ Linguistic variables,

$U_i, V_i \rightarrow$ linguistic values (high load, middle load, and low load),

$f_i(y_k) \rightarrow i^{th}$ Output value

$u_i, v_i, r_i \rightarrow$ Parameters $(i=1,2,3)$

- Create a fuzzy inference system: First, the input member function and the output member function are derived from the steps above. Secondly, based on the linguistic values (Ui, Vi) fuzzy if-then rules are generated.
- Train parameter of fuzzy inference system: In this step, the training data are trained based on the least-square method and back-spread slope descent method. The training process is iterated 60 times.

Step 4: Forecast testing datasets: When the forecasting sample reaches the stop criterion, the FIS parameters are determined from step 3; the training forecast model is then used to predict $y_k, y_{k-1}, \ldots,$ respectively, for the target training and the test datasets.

Step 5: Refine forecasts by the different statistical evaluation criteria: The forecast test datasets are obtained from step 4. To change the forecasts with less error in the training database, we use different statistical evaluation criteria.

5.3. Fuzzy Scheduler Phase

The fuzzy scheduler gets data from the analyzer phase. The data contains information on the future load of each resource. The scheduler phase operates based on three fuzzy if-then rules which are given in table 2.

Table 2 Fuzzy Scheduling Rules

| Fuzzy scheduler component rules |
|---|
| If <load>is forecasted as <Normal> |
| Then<"no change"> in the task |
| Call ISO algorithm |
| If < load> is forecasted as <high> |
| Then "chances for error occur" |
| Call PFTC strategy to minimize load |
| If < load> is Forecasted<very high> |
| Then "Fault happened" |
| Call RFTC to migration process |

**RESEARCH ARTICLE**

The first rule says; that if the future resource loads are forecasted as a normal state by the analyzer phase, then the fuzzy scheduler calls the ISO algorithm to schedule the workflow by considering makespan, energy, and the total cost.

The first rule says; that if the future resource loads are forecasted as high state, then the fuzzy scheduler phase invites the PFTC to prevent the fault event. In this PFTC, the adaptive weight adjustment algorithm is utilized to balance the load.

The second rule says; that if the future resource loads are forecasted as too high, then the fuzzy scheduler phase calls the RFTC, which is used to minimize the server failure. If one of the available active VMs fails, that error indicates that the action must be identified and an advance migration algorithm must be enabled.

5.3.1. Scheduling Using Improved Squirrel Optimization Algorithm (ISOA)

Scheduling is an important process for tolerating the fault and reducing the energy consumption of the cloud. To achieve this concept, the ISO algorithm is presented. The SOA algorithm is a bio-logical inspiration optimization algorithm, which is developed based on the food search behavior of the flying squirrels. The most fascinating reality about flying squirrels is that they don't fly, rather they utilize a special technique for locomotion for example "Gliding" which is viewed as enthusiastically modest, permitting little mammals to cover enormous distances rapidly and proficiently. This method provides efficient search space exploration as the seasonal surveillance level is linked. Furthermore, there are three types of trees in the forest, such as the common tree, the oak tree, and the hickory tree, which preserve population diversity and enhance algorithmic studies. Further, to improve the performance of SOA, opposition-based learning (OBL) is adapted to SOA. The OBL strategy improves the searching ability of SOA. The step-by-step process of the fault-tolerant-based task scheduling process is explained in table 3.

Table 3 Solution Encoding Format

| **No. of task / No. of VM** | **$T_1$** | **$T_2$** | **$T_3$** | **$T_4$** | **$T_5$** | **$T_6$** | **$T_7$** | **$T_8$** | **$T_9$** |
|---|---|---|---|---|---|---|---|---|---|
| VM1 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 |
| VM2 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 |
| VM3 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 |

Step 1: Solution encoding: Initialization of the parameter used in this paper is done in this section. Count of workflow tasks, amount of VMs, population size, CPU capacity, maximum iteration, and ISOA parameters. The solutions are then randomly generated. The solution consists of VM and task. The solution encoding format is given in table 1.

In table 1, there are nine tasks and three VMs are considered. In the table, "1" represent, the corresponding task assigned to VM, and the "0" represent, the task is not assigned to the VM. Only one VM is assigned a task.

Step 2: Opposite solution generation: Based on initial solutions, opposite solutions are developed. By creating opposite solutions, search efficiency is increased. Where, $S \in [a,b]$ is the real number. The opposite solution $\overline{S}$ is estimated by using equation (15);

$$\overline{S} = a + b - S \quad (15)$$

Step 2: Fitness Evaluation: The fitness value of the solutions is essential in determining which solutions to select to create the next iteration. For fitness, in this paper, MOF is designed which is based on the makespan, total processing cost, and power consumption. A good scheduling system should reduce the fitness function. The fitness function is given in equation (16).

$$Fitness = \min[\alpha_1 \times \cos t + \alpha_2 \times makespan + \alpha_3 \times Energy] \quad (16)$$

Step 3: Solution updation: The solutions are updated after the fitness calculation. Solutions will be updated as they progress through the following phases.

Sort, declare, and random selection process: Each solution's fitness values are sorted ascending after being determined. In the hickory nut tree, the best location (i.e. minimum exercise) is considered to be. Acorn nuts in trees and hickory nuts are the next two best solutions, as well as the squirrel that tries to go towards them. The left is thought to be in ordinary trees on the squirrel. The announcement states that some squirrels are chosen at random to travel to the hickory nut tree. Due to the presence of hunters on squirrels of all kinds, the algorithm is designed to take into account the probability that the predator is present $(P_{prob})$.

Generate new locations: based on the hunting probability ( $p_{prob}$) the squirrel locations are updated. The following norms are used for updation process.

Norm 1: Flying squirrels on the acorn nut trees ( $S_A^i$ ) will more often than not move towards the hickory nut tree which is given in condition (17).

**RESEARCH ARTICLE**

$$S_A^{i+1} = \begin{cases} S_A^i + g_{dist} \times g_{const} \times \left(S_H^i - S_A^i\right) & ; r_1 \geq p_{prob} \\ Rand & ; otherwise \end{cases} \quad (17)$$

Norm 2: Flying squirrels lying on the typical trees ($S_N^i$) probably travel towards acorn nut trees which are given in equation (18).

$$S_N^{i+1} = \begin{cases} S_N^i + g_{dist} \times g_{const} \times \left(S_A^i - S_N^i\right) & ; r_2 \geq p_{prob} \\ Rand & ; otherwise \end{cases} \quad (18)$$

Norm 3: Squirrels flying over normal trees ($S_N^i$) try to move towards hickory nut trees given in equation ( 19).

$$S_N^{i+1} = \begin{cases} S_N^i + g_{dist} \times g_{const} \times \left(S_H^i - S_N^i\right) & ; r_3 \geq p_{prob} \\ Rand & ; otherwise \end{cases} \quad (19)$$

The above equations, random gliding distance is represented as $g_{dist}$; $(g_{const} = 1.9)$ represents the gliding constant, $i$ and $i+1$ refers the new and following iteration; $S_A^i$, $S_H^i$ and $S_N^i$ denotes the location of the flying squirrel. Also, $r_1$, $r_2$ and $r_3$ are arbitral value among [0, 1].

*Aerodynamics of gliding*: The flying squirrel at the standard angle of landing at the horizontal angle is defined as the lift-to-drag ratio (also called the rate of glide), which is calculated using equation (20)

$$G = \frac{E}{d} = \frac{1}{\tan \psi} \quad (20)$$

The above equation $\psi$ indicates the glide angle, which is calculated using equation (21).

$$\psi = \arctan\left(\frac{d}{E}\right) \quad (21)$$

The small value $\psi$ helps increase the glide path length of the squirrels. Also, the lift force equation is represented in (22).

$$E = \frac{1}{2\phi E_{coef} T^2 L} \quad (22)$$

Further, frictional drag is calculated using equation (23)

$$d = \frac{1}{2\phi d_{coef} T^2 L} \quad (23)$$

Where $\phi = (1.204 kgm^{-3})$ indicates the mass of air; $T = 5.25 ms^{-1}$ represents speed; $L = 154 cm^2$ specifies the outside area of the body; $E_{coef}$ and $d_{coef}$ describes the lift coefficient and drag coefficients respectively.

*Seasonal monitoring condition*: This is because seasonal change plays an important role in the search for food for flying squirrels. Here are some of the steps involved in mathematically expressing seasonal monitoring are given in table 4.

Table 4 Seasonal Monitoring Condition

calculate seasonal constant ($c_s^{(i)}$) by,

$$c_s^{(i)} = \sqrt{\sum_{j=1}^{z} \left(S_{A,j}^i - S_{H,j}^i\right)^2}$$ , where $i = 1,2,3$ is the iteration count

Find low seasonal constant using,

$$c_0 = \frac{10 e^{-6}}{(365)^{\frac{2.5i}{i_{\max}}}}$$, where $i_{\max}$ is the maximum iteration

***// Analyze seasonal monitoring condition***

if $\left(c_s^i \leq c_0\right)$

Perform random relocation

Else $\left(c_s^i > c_0\right)$

Flying squirrels themselves explores for the best food source

5.3.2. Random Relocation During the Winter Season

Observations indicate active squirrels that can explore their ideal food source according to their seasonal position. The random migration strategy is also being used to make sure that squirrels surviving but unexplored find their optimal food source. Using the Levy distribution, random relocation occurs here (24):

$$S_N^* = S_{\min} + levy(m) \times \left(S_{\max} - S_{\min}\right) \quad (24)$$

Where $levy(y)$ indicates the levy distribution which is calculated using equation (25).

$$levy(y) = 0.01 \times \frac{u_p \times \delta}{\left|u_q\right|^{1/\chi}} \quad (25)$$

**RESEARCH ARTICLE**

In this paper, we used the constant $\chi = 1.5$ and random number $u_p$ and $u_q$ distributed normally within [0,1]. Also, $\delta$ is calculated using equation (26).

$$\delta = \left( \frac{\Gamma(1+\chi) \times \sin\left(\pi\chi/2\right)}{\Gamma\left(\frac{1+\chi}{2}\right) \times \chi \times 2^{\left(\chi-1/2\right)}} \right)^{\frac{1}{\chi}} \quad (26)$$

Where: $\Gamma(y) = (y-1)!$

*Termination condition*: The termination condition is performed the best solution is reached or when the maximum number of repetitions is reached. At the end of ISOA, the optimal scheduled task are found, which are used for scheduling.

5.3.3. PFTC

Due to the high load on the VMs, the process on the VM layers fails, which is considered a fault. The ultimate objective of PFTC is to avoid incoming workload to the fault zone and it does not assign new loads to overloaded VMs. The fault detection rules of the proactive controller are explained below;

- If a VM is recognized as having an additional load by the analyzer phase, a higher load situation will occur. In this scenario, the PFTC calculates the exact load weight of the VM and priority parameter for the target VM relative to the current CPU usage status. By changing the weight value, the load balancer sends low-responsibility solicitations to this VM to guarantee that the CPU is secure.
- The situation under load is recognized when the usage of CPU resources is reduced at regular intervals (i.e., the number of resource-driven workflow tasks is small and often inactive). By increasing the weight and priority value of idle VMs to meet higher workflow requirements, the PFTC reduces the possibility of other VMs failing due to failures on idle VMs.

5.3.4. RFTC

In the proposed work, the fuzzy scheduling stage calls on the RFTC to tolerate the error if the analyst level considers the property load to be exceptionally overestimated. This indicates that present dynamic VMs is vulnerable, which should be differentiated as a fault reaction, and fault tolerance cycles will be quickly placed in the program, in these ways, a displacement mechanism will be implemented. VM migration is a strategy to work with development providers to oversee the production of cloud assets [41]. A direct VM migration is the transfer of an overloaded VM to another VM without interrupting the new operation of utility services. The resource utilization is calculated using equation (27).

$$Utilization_{vm_i} = \frac{Total\ Req_{Band}}{Total\ Mips_{Host}} + \frac{Total\ Req_{Band}}{Total\ Bandwidth_{Host}} + \frac{Total\ Req_{RAM}}{TotalRAM_{Host}} \quad (27)$$

The total VM utilization is calculated using equation (28)

$$Total\ VM\ \ utilized = Utilization_{VM_i} - Capacity_i \quad (28)$$

The overall migration time is calculated using equation (29)

$$Total\ Migration\ Time = Time\ taken\ to\ copy\ the\ VMs + Total\ data\ migrated \quad (29)$$

Time taken to copy the VMs is calculated using equation (30).

$$Time\ taken\ \ to\ copy\ \ the\ VMs\ \ = Temporary + Host_{load} \quad (30)$$

Temporary is calculated using equation (31)

$$Temporary = \frac{\frac{\sum_{i=1}^{n} VM_i^{CPU}}{Total\ \ Mips\ \ of\ \ Host} + \frac{\sum_{i=1}^{n} VM_i^{RAM}}{Total\ \ RAM\ \ of\ \ Host} + \frac{\sum_{i=1}^{n} VM_i^{BAN}}{Total\ \ Bandwidth\ \ of\ \ Host}}{3} \quad (31)$$

Total data migration is calculated using equation (32)

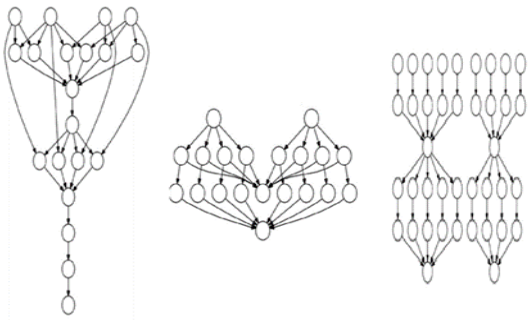
$$Total\ \ data\ \ migrated\ \ = Maximum\ \ threshold\ \ - Host_{load} \quad (32)$$

5.4. Executer Component

Once scheduling process is completed, the executor component is assigning the task to the corresponding CVMs. The actual execution of the actions determined by the scheduling components is the responsibility of this component.

## 6. RESULTS EVALUATION

The results obtained from the presented approach are explained in this section. The suggested technique is implemented using JAVA and Windows 10 operating system. Montage, cyber shake, and LIGO workflows are used in experimental analysis. The workflow system is shown in Figure 3.



(a) (b) (c)

Figure 3: Experimental Used Workflow Models (a) Montage, (b) CyberShake, and (c) LIGO

RESEARCH ARTICLE

6.1. Experimental Analysis Based on Energy Consumption

Energy consumption is a very important parameter for scheduling and resource allocation on the cloud. We properly allocate the task on resources means; we can reduce energy consumption. In this work, the MOF has focused on energy consumption also. The comparative analysis based on energy for different workflow models is presented in this section.

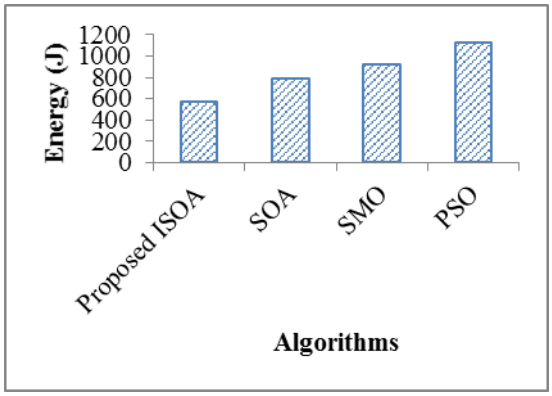



Figure 4 Comparative Analysis Based on Energy for Montage Workflow

In figure 4, for Montage workflow, the efficiency is discussed using on energy. The total energy consumed by the execution of all tasks is called energy consumption. A good scheduling system should have minimum energy consumption. When analysing figure 4, our proposed approach is consumes less amount of energy of 574J for scheduling Montage workflow which is 785J for SOA-based scheduling, 925J for SMO-based scheduling, and 1127J for PSO-based scheduling. Due to the OBL strategy with SOA, in this paper, our proposed approach has achieved minimal energy consumption compared to other methods. Because the OBL strategy increases the searching ability of SOA.

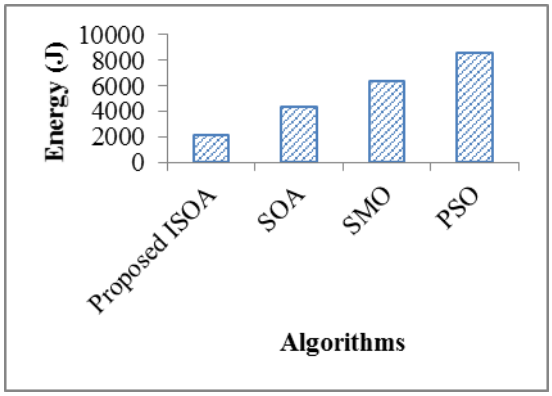



Figure 5 Comparative Analysis Based on Energy for CyberShake Workflow

In figure 5, energy of Cybershake workflow is analysed. In this figure, we analyse the different methods namely SOA, SMO, and PSO with our approach. According to figure 5, the proposed ISOA-based scheduling has achieved better results and the PSO-based scheduling has achieved the worst output compared to other methods. In figure 6, the comparative analysis results of proposed against existing based on energy consumption are analyzed. Here, the proposed approach achieved better results than already.

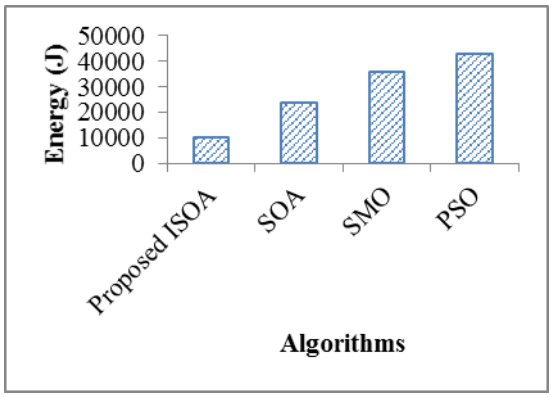



Figure 6 Comparative Analysis Based on Energy for LIGO Workflow

6.2. Experimental Analysis Based on Cost

One of the important parameter of scheduling is cost. The cost is varies for task and VM. In this section, the efficiency of recommended technique is analysed in terms cost. A good scheduling system should complete its execution at a minimum cost. The outcome came from three workflows are described below:

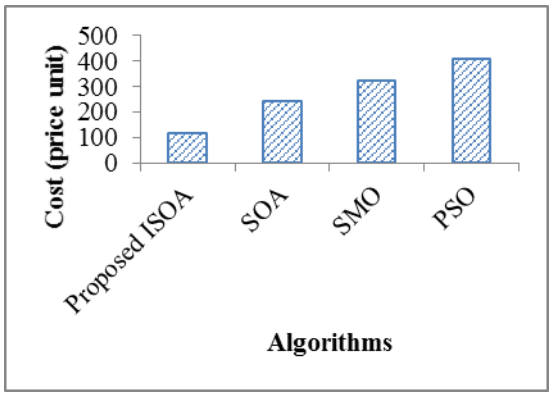



Figure 7: Experimental Results Based on Cost Using Montage Workflow

In figure 7, ISOA performance is analyzed in terms of cost using montage workflow. Any user who wants to schedule their task on resource, pay some amount. The resource cost is related to CPU and memory. When analyzing figure 7, our proposed approach is takes the minimum cost of 118$ for scheduling montage workflow which is 241$ for SOA-based montage workflow scheduling, 321$for SMO-based montage

RESEARCH ARTICLE

workflow scheduling, and 408 for PSO-based montage workflow scheduling. The effectiveness of ISOA is analyzed based on cost using Cybershake work scheduling given in figure 8. As expressed in Figure 8, our recommended model achieved improved results compared to the other method. Similarly, in figure 9, the proposed approach takes minimum cost for scheduling LIGO workflow compared to other methods.

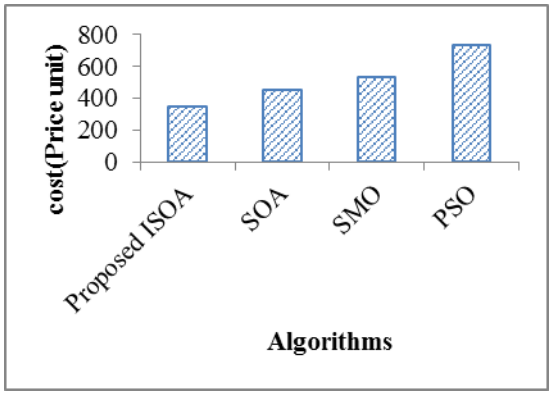



Figure 8 Experimental Results Based on Energy Using Cybershake Workflow

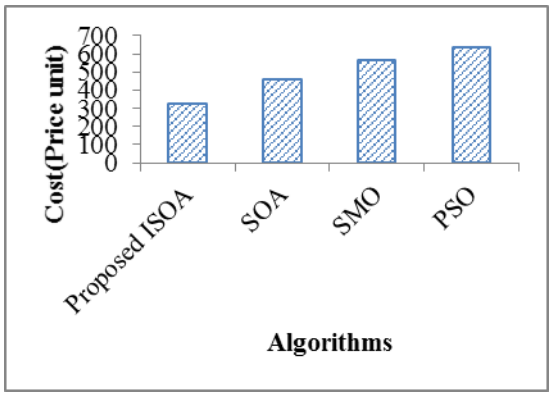



Figure 9 Experimental Results Based on Cost for LIGO Workflow

6.3. Comparative Analysis Based on Makespan

The results obtained by various workflow model based on makespan is analyzed in this section.

Figure 10 illustrates the makespan of the proposed and other algorithm using Montage workflow. When analyzing figure 10, ISOA based scheduling approach attained the makespan of 124s which is 186s for SOA-based scheduling, 224s for SMO-based scheduling, and 267s for PSO-based scheduling. The graph shows that the makespan of ISOA-based scheduling is lesser than the SOA, SMO, and PSO-based workflow scheduling. This is due to the adaptation of ISOA. In figure 11, the effectiveness of the proposed approach is analyzed based on makespan using Cybershake workflow. For the scheduling process, makespan is an important parameter. The task flow affects the makespan of the scheduling. Compared to another method, ISOA attained the minimum makespan. Figure 12 illustrates the makespan using the LIGO workflow. According to figure 12, the ISOA technique takes a minimum makespan of 4157s. From figures 10-12, we understand, that the ISOA-based workflow scheduling technique takes a minimum makespan of 124s, 387s, and 4157s for Montage, Cybershake, and LIGO respectively.

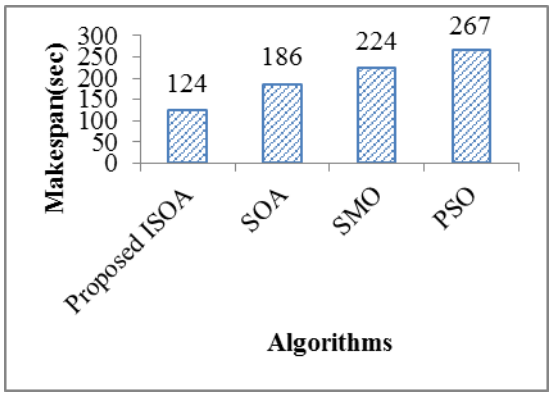



Figure 10 Comparative Analysis Based on Makespan for Montage Workflow

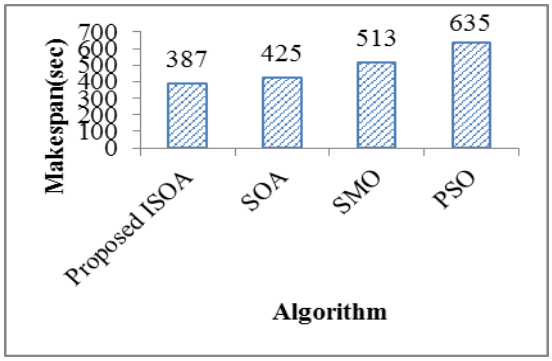



Figure 11 Comparative Analysis Based on Makespan for Cybershake Workflow

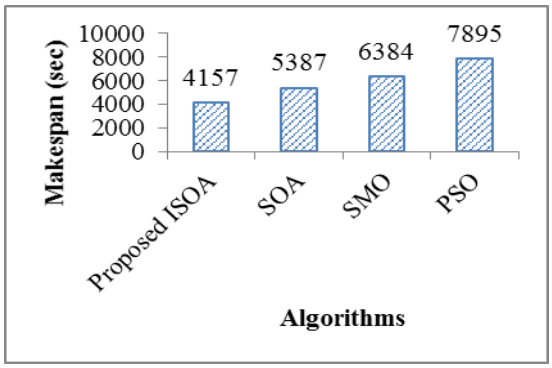



Figure 12 Comparative Analysis Based on Makespan for LIGO Workflow

**RESEARCH ARTICLE**

6.4. Comparative Analysis Based on Fault Ratio

In this section, the recommended approach efficiency is discussed based on fault ratio. A good scheduling system should have minimum faults. The following graphs clearly show the outcome of the proposed method.

Table 4 Comparative Analysis Based on Fault Ratio for Montage Workflow

| Number of tasks | Proposed ISOA | SOA | SMO | PSO |
|---|---|---|---|---|
| 25 | 4 | 5 | 6 | 7 |
| 50 | 6 | 8 | 10 | 12 |
| 100 | 8 | 10 | 12 | 15 |
| 500 | 10 | 12 | 14 | 17 |
| 1000 | 12 | 14 | 16 | 20 |

Table 5 Comparative Analysis Based on Fault Ratio for Cybershake Workflow

| Number of tasks | Proposed ISOA | SOA | SMO | PSO |
|---|---|---|---|---|
| 25 | 5 | 7.3 | 10.1 | 10.8 |
| 50 | 5.5 | 8 | 11.3 | 11.9 |
| 100 | 7.5 | 9.5 | 12.1 | 12.8 |
| 500 | 8.2 | 10.2 | 13.5 | 14.2 |
| 1000 | 10 | 11.5 | 14.3 | 15.1 |

Table 6 Comparative Analysis Based on Fault LIGO for Montage Workflow

| Number of task | Proposed ISOA | SOA | SMO | PSO |
|---|---|---|---|---|
| 25 | 5.5 | 6 | 7.5 | 7.9 |
| 50 | 4.3 | 5.5 | 7 | 7.2 |
| 100 | 10 | 12 | 12.3 | 13 |
| 500 | 11.2 | 12.8 | 13 | 13.8 |
| 1000 | 13.5 | 16 | 16.8 | 17.5 |

In table 4, a comparative analysis based on the error rate is given. The error rate gradually increases as the number of tasks is maximized. The minimum failure rate increases the system performance. In this proposed approach, to reduce the failure rate, we use CPU load prediction using ANFIS and we replace the faulty VM with new VMs. Similarly, the approach proposed in table 5 and 6 achieved a better error rate compared to the other methods. It is clear from the results that the suggested model has yielded the best results compared to other methods. This is due to the ANFIS-based CPU load prediction; ISOA-based scheduling, and reactive and pro-active fault tolerant strategy.

## 7. CONCLUSION

In this paper, a multi-objective fault tolerance model based on scientific workflow scheduling on the cloud has been proposed. The fault was tolerated based on ANFIS-based CPU prediction, the faulty VM was replaced by a new VM with, a proactive and reactive fault tolerance strategy. Similarly, the workflow schedule has been done based on the ISOA. For scheduling multi-objective function has been designed by considering energy cost and makespan. The mathematical model of each algorithm has been clearly explained. The experimental result has been carried out based on three workflows namely, Montage, Cybershake, and LIGO. The effectiveness of the recommended approach has been analysed based on cost, makespan, energy, and the fault ratio.

## REFERENCES

[1] S.M.Jaybhaye and Vahida Z. Attar, “A Review on Scientific Workflow Scheduling in Cloud Computing”, Proceedings of the 2nd International Conference on Communication and Electronics Systems (ICCES 2017) IEEE Xplore Compliant - Part Number: CFP17AWO-ART, ISBN: 978-1-5090-5013-0

[2] Bhaskar Prasad Rimal, Martin Maier, “Workflow Scheduling in Multi-Tenant Cloud Computing Environments”, IEEE Transactions on parallel and distributed systems , Vol 28, No. 1, Jan 2017, pp 290-304.

[3] G. Natesan and A. Chokkalingam, “Multi-Objective Task Scheduling Using Hybrid Whale Genetic Optimization Algorithm in Heterogeneous Computing Environment,” Wireless Personal Communications, 2019.

[4] G. Juve and E. Deelman, “Scientific Workflows in the Cloud,” Grids, Clouds and Virtualization, pp. 71–91, 2011

[5] S. Saeedi, R. Khorsand, S. Ghandi Bidgoli, and M. Ramezanpour, “Improved many-objective particle swarm optimization algorithm for scientific workflow scheduling in cloud computing,” Computers and Industrial Engineering, vol. 147, no,June, p. 106649, 2020

[6] Juve, Gideon, Ann Chervenak , EwaDeelman, ShishirBharathi, Gaurang Mehta, and Karan Vahi, “Characterizing and profiling scientific workflows”, Future Generation Computer Systems, Vol. 29, Issue 3 , 2013: 682-692.

[7] Fan Zhang, Junwei Cao, Kai Hwang,Keqin Li, and Samee U. Khan, “Adaptive Workflow Scheduling on Cloud Computing Platforms with Iterative Ordinal Optimization”, IEEE Transaction on cloud computing , Vol 3, No. 2, April/June 2015, pp 156-168.

[8] Tongyi Zheng and Weili Luo, “An Improved Squirrel Search Algorithm for Optimization”, Hindawi Complexity Volume 2019, Article ID 6291968, 31 pages

[9] Yong Zhao, IoanRaicu, Shiyong Lu, Wenhong Tian, Heng Liu, Enabling scalable scientific workflow management in the Cloud”, Future Generation Computer Systems, 23 October 2014.

[10] Heyang Xu, Bo Yang, Weiwei Qi and Emmanuel Ahene, “A Multi-objective Optimization Approach to Workflow Scheduling in Clouds Considering Fault Recovery”, KSII Transations on internet and information system vol. 10, NO 3, Mar. 2016.

[11] Ahmad, Z.; Jehangiri, A.I Ala’anzy, M.A. Othman, M.Umar, A.I. “Fault-Tolerant and Data-Intensive Resource Scheduling and

**RESEARCH ARTICLE**

Management for Scientific Applications in Cloud Computing", Sensors 2021, 21, 7238.
[12] Zhongjin Li, Jiacheng Yu, Haiyang Hu, Jie Chen, Hua Hu, Jidong Ge and Victor Chang, "Fault-Tolerant Scheduling for Scientific Workflow with Task Replication Method in Cloud", The 3rd International Conference on Internet of Things, Big Data and Security (IoTBDS 2018), pages 95-104 ISBN: 978-989-758-296-7
[13] J. KokKonjaang and Lina Xu, "Multi-objective workflow optimization strategy (MOWOS) for cloud computing", Journal of Cloud Computing: Advances, Systems and Applications, (2021) 10:11
[14] Yuandou Wang, Hang Liu, Wanbo Zheng, Yunni Xia, Yawen Li, Peng Chen, KunyinGuo, and Hong Xie, "Multi-Objective Workflow Scheduling With Deep-Q-Network-Based Multi-Agent Reinforcement Learning", Special Section on Mobile Service Computing with Internet of Things, February 11, 2019.
[15] T. Prem Jacob, K. Pradeep, "A Multi-Objective Optimal Task Scheduling in Cloud Environment Using Cuckoo Particle Swarm Optimization", Wireless Personal Communications · November 2019 DOI: 10.1007/s11277-019-06566-w.
[16] Amandeep Varma, SakshiKushal, "A hybrid multi-objective Particle Swarm Optimization for scientific workflow scheduling", Parallel computing volume 62, February 2017, Pages 1-19.
[17] Neeraj Arora, Rohitash Kumar Banyal, "HPSOGWO: A Hybrid Algorithm for Scientific Workflow Scheduling in Cloud Computing", (IJACSA) International Journal of Advanced Computer Science and Applications, Vol. 11, No. 10, 2020.

Authors

**S. Anuradha** is a Research scholar from Periyar University, Salem. Presently I am working as Assistant Professor in Department of Computer Science at Shri Sakthikailassh Women's College, Salem. My research interest includes Fault Tolerence, Load Balancing and Optimization.

**Dr. P. Kanmani** received her Ph.D from Mother Terasa Women's University, Kodaiknal in 2014. Presently she is working as Assistant Professor in Department of Computer Science at Thiruvalluvar Govt. Arts College, Rasipuram. Her research interest includes Mobile Computing, Fault Tolerence.

**How to cite this article:**

S. Anuradha, P. Kanmani, "Multi-Objective Fault Tolerance Model for Scientific Workflow Scheduling on Cloud Computing", International Journal of Computer Networks and Applications (IJCNA), 9(4), PP: 438-450, 2022, DOI: 10.22247/ijcna/2022/214505.