

# Hybrid fault tolerant cost aware mechanism for scientific workflow in cloud computing

Chaya T. Doddaiiah<sup>1</sup>, Mohamed Rafi<sup>2</sup>

<sup>1</sup>Department of Computer Science and Engineering, University BDT College of Engineering, Davangere, India

<sup>2</sup>Department of Studies in Computer Science and Engineering, University BDT College of Engineering, Davangere, India

## Article Info

### Article history:

Received Jan 30, 2023

Revised Sep 5, 2023

Accepted Sep 17, 2023

### Keywords:

Cloud computing

Cost minimization

Fault-tolerant

Hybrid cost-aware fault tolerant

Resource management

## ABSTRACT

Cloud computing provides solutions for diverse commercial and academic applications which is the primary goal. Scientific workflows are used in the cloud-computing environment to analyse large-scale scientific applications. For scientific workflows, many data is required, and a single scientific workflow that includes hundreds of stages, depending on the application's time restrictions, task failures, money limits, incorrect task organization, and task management issues can all hinder the implementation of scientific methods. In light of this, a cloud-based scientific workflow management and scheduling system that is fault-tolerant and data-oriented method are proposed. This research designs a novel hybrid cost-aware fault tolerant (HCFT) mechanism for minimizing the cost. Moreover, HCFT integrates optimal clustering and efficient resource utilization through parallel and distributed execution. Novelty of HCFT lies in novel clustering of the similar task for improvisation, CyberShake, laser interferometer gravitational wave observatory (LIGO), Montage, and sRNA identification protocol using high throughput technology (SIPHT) processes are used in the simulations to evaluate the performance of the proposed approach.

This is an open access article under the [CC BY-SA](https://creativecommons.org/licenses/by-sa/4.0/) license.



## Corresponding Author:

Chaya T. Doddaiiah

Department of Computer Science and Engineering, University BDT College of Engineering

Davangere-577 004, Karnataka, India

Email: t.chaya@rediffmail.com

## 1. INTRODUCTION

The objective of "computer as a utility" has been realized via cloud computing, a new platform for distributed computing [1]. The cloud was created by combining two significant computing technologies, cluster computing, and grid computing [2]. Cloud computing provides specifically on-demand access to trustworthy resources and pay-per-use customization of computer settings [3]. Cloud computing offers a range of dynamically scaled, virtualized, abstracted, and adaptable computer resources and services [4]. Networks, storage, servers, and applications are all made available as cloud resources in a subscription-based paradigm. The three main architectural elements of cloud services are infrastructure as a service (IaaS), platform as a service (PaaS), and software as a service (SaaS). Due to the demand-based availability of high-speed internet connectivity, these services are accessible to clients outside the company SaaS.

Research and commercial applications make extensive use of cloud computing's capabilities and services [5]. Business apps are organized according to best practices and are task-oriented. To manage such operations, organizations adopt business models like Amazon EC2. In contrast, scientific applications are created according to scientific principles and are data-driven. In addition to the Pegasus workflow management system (WMS), other systems are utilized to coordinate scientific activities.

According to Faragardi *et al.* [6], scientific applications are data-intensive and need a substantial amount of computing power and storage space for assessment and computation. Applications for scientific workflows are precisely groups of computing activities structured in various ways. When organized as a scientific process, even a single scientific application comprises several computer network tasks. Each phase of scientific study requires a large amount of data and computational resources. Numerous additional academic disciplines, including astronomy, biology, gravitational physics, and earthquake science, have universal applicability. Multiple cloud models are designed to be optimized by workflow schedule; nevertheless, customers must give priority to quality of service (QoS) satisfaction when submitting workflow applications, including cost execution and deadline. In addition, there are issues with energy consumption, time constraints, makespan optimization, and cost reduction due to the increasing demand for computers and services in scientific workflow applications. Nodes are used to represent jobs in the direct acyclic graph (DAG) modeling approach for workflows, while edges are used to show interactions between tasks [7]. DAG is therefore used to express workflows.

The fault tolerance model utilizes a single backup in the event of a failure; as a novel concept, it was initially enticing, but the complexity of scientific processes renders it incapable of withstanding several failures. As a result, it is required to develop and execute a fault tolerance model that can withstand repeated failures and increase the requirement for dependability; cost optimization is also crucial, as the ideal cost symbolizes the model's effectiveness [8]. There are two sorts of fault occurrence methodologies for managing the various types of failure: active and reactive fault occurrence processes. In addition, a reactive fault-tolerance method is intended to mitigate the issue and enhance the efficacy of the fault-tolerance technique. In addition, while incurring execution costs, this fault-tolerant technique offers system dependability [9]. The goal was to decrease response time and energy usage. In addition, by using replication, this strategy decreases the labor required to reject data due to system failure occurrences; yet, cost optimization remains its largest challenge. Moreover, Shahidinejad and Barshandeh [10] created fault-tolerant dynamic scheduling (FTDS), a fault-tolerance technique intended to overcome the static scheduling strategy. FTDS recognizes the processor fault and then attempts to efficiently reschedule the halted operations.

There are several actions and tasks with various limitations in scientific workflows. For management's benefit, various workflow management techniques are used to plan and carry out scientific processes on the intended resources [11], [12]. When managing and scheduling scientific workflows, several factors may be used in a system that could result in performance degradation, involving the creation, organization, and control of scientific workflows as well as task management, resource management, scheduling guidelines, and fault-tolerant procedures [13]. Additionally, some of the processes are too big to transfer across nodes without incurring additional costs. Five practical scientific methods for various scientific applications were thoroughly analyzed from the start [14]. In the following fields, these techniques can be applied: i) seismic research with CyberShake; ii) biological research with sRNA identification protocol using high throughput technology (SIPHT); iii) astronomy using Montage; iv) genetic research with epigenomics; and v) gravitational physics with laser interferometer gravitational wave observatory (LIGO).

The research details each scientific approach's organizational, data, and technological needs. Additionally, words from the computer industry are used to illustrate the many structural and functional aspects of scientific procedures. A few examples of these characteristics are the pipeline, data parallelism, data dispersion and redistribution, data aggregation, and compositions of scientific procedures. A scalable workflow management system used for automating research is called a WMS. It was characterized by Fan *et al.* [15]. Distributed computer infrastructures are changed to provide room for scientific process conceptual models. In our research, we analyze and comprehensively address the aforementioned issues by presenting: a data-driven, fault-tolerant model is developed.

The following are the most significant contributions of this study: i) this research work designs and develops a hybrid cost-aware fault tolerant (HCFT) mechanism in workflow scheduling considering a cloud-computing environment; ii) HCFT minimizes cost through novel clustering and optimal balancing mechanism, which reduces the entire cost; moreover fault-tolerance is carried out through the clustering approach; also parallel processing is carried out for further optimization of the process; and iii) HCFT is evaluated considering scientific workflow to prove the model efficiency based on the execution cost for various variants of workflow like CyberShake, Montage, SIPHT, and Inspiral.

This particular research is organized as follows: the first section starts with the background of cloud computing and the integration of workflow scheduling along with it. The further section proceeds with the challenges and needs for the fault-tolerant model in workflow scheduling. The section ends with a research contribution. The second section focuses on discussing the existing fault-tolerant technique concerning cost reduction; the third section designs the mathematical modeling of the HCFT model and is evaluated in the fourth section.

## 2. RELATED WORK

Five real scientific methods with various scientific applications were thoroughly examined, starting with Fan *et al.* [15]. Epigenomics is used to investigate genetics, Montage is used to study astronomy, CyberShake is used to analyze seismic activity, SIPHT is used to study biology, and LIGO is used to study gravitational physics. The organizational, data, and computer needs of each scientific technique are covered in the article. Additionally, certain structural and functional features of scientific methods are examined using computers. This includes information on data aggregation, data parallelism, data dispersion and redistribution, and data aggregation. It also includes works that employ the scientific approach. According to research by Ahmad *et al.* [16], Pegasus is a scalable WMS for the automation of research. Distributed computer systems are constructed using abstract models of scientific procedures.

A scheduling technique for scientific workflows called dynamic scheduling of a bag of tasks-based workflows (DSB) tries to cut costs while still following user-specified time limits. The technique breaks down into bag of tasks (BoTs), optimizes their distribution, and then schedules them in line with priority restrictions and data dependencies. The method satisfies the process's deadline and significantly lowers the cost of workflow computing. Utilizing methods that quickly and efficiently utilize resources is a requirement for planning scientific work. According to previous research [17]–[19], adaptive data-aware scheduling (ADAS) is a scheduling technique that gives priority to resource use and workflow completion time. It is a technique for combining task management and data for a range of jobs in the cloud. Although the suggested scheduling approach is efficient, it does not consider error tolerance, which is an essential component of workflow scheduling. The budget driven algorithm for generating high-quality schedules (BAGS) scheduling method was developed in [20] to maximize process execution time while respecting budgetary limitations. The BAGS algorithm distributes money to activities before making choices about dynamic resource provisioning and scheduling in response to environmental changes. The approach still needs to be improved in terms of time constraints and error-tolerance measures even if it is effective for scheduling scientific activities while adhering to financial constraints. When organizing scientific procedures, the designed dynamic benefit weighted scheduling (DBWS) method takes time and financial constraints into consideration.

Activities that require a lot of data and calculation are thought to utilize the most energy. To use less energy, Juarez *et al.* [21] offered a real-time dynamic scheduling system for the effective execution of task-based applications. The authors developed a polynomial-time solution that satisfies the requirements of low energy consumption and quick execution speed by combining a resource allocation method with a set of heuristic criteria. Authors fail to address the significant problem of accessibility of fault-tolerant systems [22]. Proclaims that when fault-tolerant practices are foregone to maintain profitability or save costs, cloud-computing systems collapse. The fault-tolerant technique for scientific workflow systems received approval for use in scientific operations in its second submission. For scientific processes to run properly, fault-tolerant techniques are required since the failure of bottleneck nodes renders the entire operation meaningless [23]. If a job is not successfully finished during execution, it is promptly resubmitted to the same resource or another. As a result, faster, a dynamic fault-tolerant scheduling approach was created (a method for fault-tolerant scheduling of real-time scientific workflows).

Applications for scientific methods include detailed, multi-level computation. Since all activity levels demand the same services, this sort of computing is perfect for fault-tolerant clustering (FTC) systems. The FTC method, developed in [24]–[26], is useful for many scientific projects. The aforementioned study offers three techniques: dynamic clustering (DC), selective re-clustering (SR), and dynamic re-clustering (DR). DC's initial approach is to maintain a clustering factor that is dynamically adjusted to the rate of job failure. The second technique, known as SR, is repeatedly carrying out failed tasks inside a single position. The third technique, known as DR, combines the first two strategies by providing failing tasks within a job a second chance in addition to dynamically maintaining the clustering factor based on the failure rate of recognized activities. Improvements to data-oriented scheduling methods with dynamic clustering defeat-resistant mechanism the title of the piece is [27]. Data-oriented scheduling was integrated by the enhanced data-oriented scheduling strategy with dynamic clustering fault-tolerant technique (EDS-DC) developers. A dynamic clustering method with fault tolerance was offered by EDS-DC. The results of processes that were modeled using workflows were compared to the three well-known scheduling rules minimum completion time-dynamic clustering (MCT-DC), max-min-DC, and min-min-DC. Simulations show that EDS-DC significantly decreased cost and manufacturing time when compared to traditional methods. For scientific activities, Chakraborty *et al.* [28] proposes a QoS aware fault tolerant workflow management system (QFWMS). A cluster-based, fault-tolerant, data-intensive (CFD) approach is given in [29] for cloud-based scientific applications. The suggested CFD approach provides a precise strategy for obtaining outcomes from the presentation of scientific data. The figures show that, when using the Montage workflow, the CFD approach outperformed the alternatives [30]. Applications of scientific processes have other distinctive features aside from pipelining, parallelism, integration, and disintegration. These programs use a lot of computation and

information. Scientific workflow applications require workflow management and fault-tolerant scheduling systems that are data-centric and have high data storage and processing capacities.

Scientific workflows must be gathered, categorized, and managed using a data-oriented scheduling-based energy-efficient WMS since they are composed of several data- and computation-intensive applications at the bottleneck node or level, a large number of workflow activities are performed, and if even one of them fails, the execution as a whole is meaningless. A fault-tolerant system is therefore required. Given that different workflow activity at various levels, have comparable service and resource needs, scientific operations can be carried out using a cluster-based scheduling and fault-tolerant approach. A fault-tolerant, data-centric, and energy-efficient system for coordinating and scheduling scientific activities is produced because of these restrictions.

### 3. PROPOSED METHOD

The scientific community to perform the task using IaaS cloud as a platform has extensively utilized workflow schedules; moreover, being fault tolerant is one of the major entities in the cloud platform, which tends to minimize the cost per task and further workflow. This research work aims at designing a hybrid fault-tolerant mechanism, which reduces the cost as given in Figure 1. The data here is fetched from each user, which is further reviewed and executed. A user here refers to the individual from whom the data is collected. Here the user submits a collection of various types of data. Once it is executed, the output is generated respectively. The application interface (AI) here is responsible to connect the user with the proposed method. The user here feeds the data to the AI for execution. The AI transfers data to the next component of the model, the  $\beta$  model that transmits the data to the workflow model for size, nodes, and edges. The AI serves as an interface to one or more users or for workflows submitted to the proposed model.

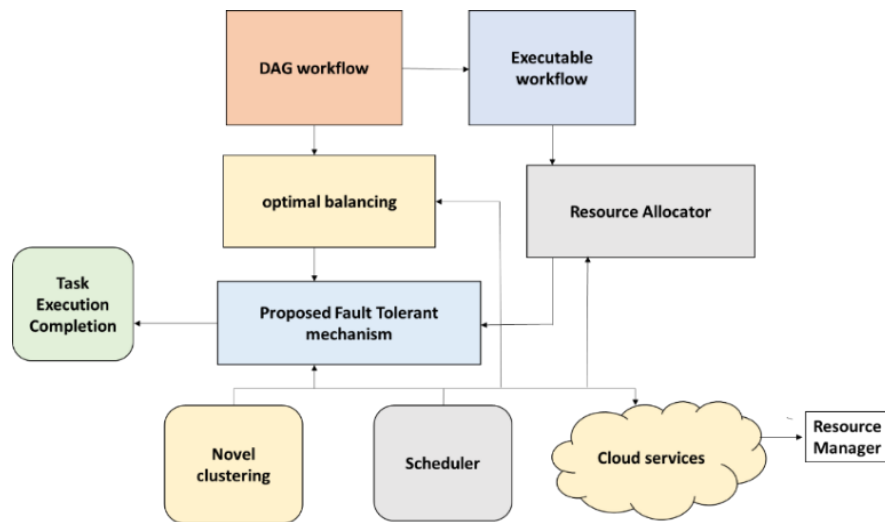


Figure 1. Hybrid fault tolerance workflow

#### 3.1. Designing optimal DAG<sub>Workflow</sub>

DAG<sub>Workflow</sub> here is depicted as  $\beta$ , the data received by the interface for the functioning and evaluation of various applications. For the data generated the  $\beta$  generates a DAG graph. The  $\beta$  functions as the data collected by various workflows produce a reliably synthetic workflow. The DAG graph structure is represented as shown in (1).

$$G_{DAG} = (S_{wf}, DF) \quad (1)$$

$G_{DAG}$  denotes a DAG with features such as  $S_{wf} = S_{wf1}, S_{wf2}, S_{wf3}, \dots, S_{wfn}$  denotes the number of iterations involved and  $DF = \{(S_{wf(x)}, S_{wf(y)} | S_{wf(x)}, S_{wf(y)} \in S_{wf}\}$  denotes the DF dependencies in between the iterations. The dependency set  $S_{wf(x)}, S_{wf(y)}$  consisting of the parameters in between the iterations  $S_{wf(x)}$  and  $S_{wf(y)}$ . The step  $S_{wf(x)}$  is termed the successor of the iteration  $S_{wf(y)}$ . Any step toward the initiation is termed the initiation step and the end step is termed the termination step shown in (2). A person can submit input and

generate an output; the  $\beta$  receives data that contains multiple inputs used by the user. This depicts the data in their subsequent workflows that generate a DAG which is further passed on to  $\Upsilon$ .

$$S_{wf(x)\_initiate} = P \quad (2)$$

$$S_{wf(x)\_terminate} = P \quad (3)$$

### 3.2. Optimal-workflow execution

The executable workflow is denoted as  $\Upsilon$  fed with multiple workflows through the  $\Upsilon$  for each resource. The  $\Upsilon$  rebuild the workflow to enhance the performance. This integrates each iteration of the workflow into a job into a single iteration henceforth the overhead is minimized. However, after that  $\Upsilon$  the job includes the execution of several iterations. The iterations are arranged following their dependencies. The iterations here are executed in a parallel and integrated sequence. At this point, the parameters for each workflow are provided by the data for the output. Additionally, the  $\Upsilon$  is also responsible to produce the data from various jobs and then estimate the storage resources, from various inputs transferred to the next model. The  $\Upsilon$  transmits all the jobs to the next component.

### 3.3. Resource allocator

The resource allocator is denoted as  $\vartheta$  allocated multiple jobs through the  $\Upsilon$  and then allocates resources to it. The allocated resources are extracted from the cloud and handled separately through the proposed technique. The  $\Upsilon$  is responsible to allocate the resources; this is done in a manner that the cost is optimized. This is accomplished by a set of iterations allocated to the resources. The resource allocated for each iteration to optimize the period to transfer data. Each iteration allocated all the resources, henceforth the task is allocated to a resource within the optimized time limit. Here  $res_{cost(x)}$  depicts the cost of the resource  $x$  and  $DT_{time}$  denotes the data transfer time for step  $x$  for resource  $x$ . By assigning the resource to each phase with the least amount of data transmission time possible, the list is traversed entirely.

$$Resource(res_{cost(x)}) = Step\{S_{wf(x)} | S_{wf(x)} \leftarrow DT_{time}(res_{cost(x)})\} \quad (4)$$

### 3.4. Fault tolerant system ()

Fault tolerant system () is denoted as  $\tau$  executes the iterations allocated by the  $\Upsilon$  which is executed completely, the interface is used to return the results to the user. The  $\Upsilon$  initiates the fault-tolerant approach and executes the failure of subsequent iterations. This model produces results upon successful execution, the  $\tau$  adapts Automate\_clustering () based on  $\tau$  once the iteration fails. In this state, the assumption is considered based on a 5% failure rate based on the jobs selected, henceforth the  $\tau()$  is initiated in each period. The optimal workload balancer here is responsible for cost optimization requiring resource utilization in ascending a payload of nodes that are dispersed to use the available nodes. The resource utilization is done based on three levels i.e Top\_peak, bottom\_peak, and Mid\_peak.

$$Resource\ usage\ (Resource(res_{cost(x)})) = \begin{cases} Mid\_peak\ (Resource\ usage = 0.4\ to\ 0.8) \\ bottom\_peak\ (Resource\ usage = 0\ to\ 0.4) \\ Top\_peak\ (Resource\ usage = 0.8\ to\ 1) \end{cases} \quad (5)$$

Here bottom\_peak, denotes the lowest peak value usage and Top\_peak value indicates the highest peak value. The resources here are made available to transfer the workload from resources for the bottom\_peak value a null value when the resources with null value are terminated. The data is fed via an AI that is responsible for providing data from two to three workflows to the proposed model. The proposed method is applied to the  $\Upsilon$  which is applied to the data given as input, the  $\Upsilon$  is the next key component. The  $\Upsilon$  converts the workflow model into appropriate resource initialization based on the iterations for the work allocated. This is then transferred to the  $\vartheta$  that sends the jobs and tasks to the workflow and then transmitted to the  $\vartheta$ , this system is responsible to schedule the resources based on the scheduling policy. The  $\tau$  uses an AI, receives the corresponding iterations, and returns the outcome to the entity. The cost-aware optimal balancer for initiating effective cost-efficient optimization is launched when the iterations fail to execute. The energy is used based on the ascending order of the resources, which further distributes the workload throughout the nodes for minimal consumption of other nodes. The  $\vartheta$  receives the jobs through the  $\Upsilon$  and schedules them by allocation of resources to the cloud,  $\Upsilon$  allots the jobs upon conversion to essential tasks. The resources are fetched from the cloud, along with the proposed system to plan and manage the tasks specifically. The  $\Upsilon$  allocates the resources in a way to spend a minimal amount of money for the completion of the task. The task is completed

by considering the list of tasks and resources, the mapping list is then generated upon the allocation of efficient resources in the shortest period for each iteration. The integration of automated clustering of the proposed method ensures fault tolerance mechanisms that are related to the iterations involved in a job for workflow are combined as a cluster and execution on a single resource. The uncompleted tasks remaining in the cluster, create clusters automatically depending on the completion of the given job and execute it repeatedly. The (6) and (7) depict the mechanism of automatic clustering.

$$\text{Ex}\left(\int_{x=1}^n S_{\text{wf}(x)}\right) \quad (6)$$

$$\text{Clus}\left(\int_{x=1}^n S_{\text{wf}(x)}\right) \quad (7)$$

$$\text{Ex}\left(\int_{x=y}^n S_{\text{wf}(y)}\right) \quad (8)$$

$$\text{AutoClus}\left(\int_{y=x}^n S_{\text{wf}(y)}\left(\text{Fail}\left(\int_{x=x}^n S_{\text{wf}(x)}\right)\right)\right) \quad (9)$$

The (8) denotes the clusters for n similar tasks and executes them, henceforth the (9) denotes the job that has failed and automatically clusters it for execution purposes. To reduce the workload, the proposed mechanism results in evaluating the resources, arranging them in ascending order, and then automatically clustering and transferring the workload for resource utilization of other resources and arranging them in ascending order that transfers the workload for the resources with the least utilization of other nodes. This categorizes the usage into three segments i.e Top\_peak, bottom\_peak, and Mid\_peak. The workload is transferred from the bottom\_peak to the Mid\_peak to make bottom\_peak resources null and then the resources with null usage are switched off. Algorithm 1 depicts the algorithm designed for minimizing the cost.

#### Algorithm 1. Hybrid-fault tolerant cost algorithm

```

Input  μ (data)
Step 1  Z1←μ (data fed to the queue), DAGWorkflow → β, Executable_workflow → Y,
Resource_Allocator → ϑ, Fault_tolerant system → τ
Step 2  β()
Step 3  Z2←fetchWorkflow() in the range wf1.....wfn up to Z1
Step 4  for(All the Workflow ( wf1.....wfn) do
        Y()
        Steps S1 to Sn for the Workflow
        end for
        ϑ()
        fetchresource() from r1.....rn)
        While (ϑ()) do
            Steps S1 to Sn for the Workflow
            for(each Steps S1 to Sn for the Workflow ( wf1.....wfn) do
                fetchresource resx and allocate it to Step S1
                resx←S1
            end for
        end while
Step 5  Initiate_execution.τ()
        While(τ ()) do
            If (System Faulty)
                Automate_clustering()
            else
                fetchresults
            end if else
        end while
Step 6  for(All resx to resi)do
        resused←fetch(resource_used)()
        If(resused(x) ==Null) then
            terminate(res((x)))
        else if (resused≤lowerthreshold) then
            shift resused alternatively
            terminate(res((x)))
        end if else
        end for
Step 7  δ←results
        return δ
output δ(results)

```

## 4. RESULT SECTION

In this section, the proposed model is evaluated upon further minimizing the cost optimization. The run-time is evaluated based on allocating jobs at values 30, 50, 100, and 1,000. This is depicted in the form of graphs for the expected outcomes shown by taking into account cost optimization. The proposed model is tested with the CyberShake dataset by using the simulator of cloudsim. The model is deployed against 64-bit Windows 11 operating system for 16 GB RAM that has Intel ®Core™i7 processor. It contains a 3.20 GHz CPU, the proposed model is deployed on EclipseWS Neon.3 editor, and the code is written in JAVA.

### 4.1. Dataset description

The performance of the proposed model is evaluated via the different scientific areas, here CyberShake is used to facilitate the earthquake hazards that are produced by synthetic seismograms and categorized according to a data-centric workflow model through by wide memory-range and basic CPU necessities. The custom Montages of the sky are based on the Montage application for the input images considered. The tasks are differentiated by using the I/O intensive applications by not considering the CPU processing activity. Gravitational waves are detected by using the astrophysics LIGO method. It is composed mostly of CPU-intensive tasks with high memory requirements. Several genome-sequencing processes are automated using the CPU-intensive epigenomics approach, which is utilized by the bioinformatics industry. In bioinformatics, SIPHT is used to automate the search for sRNA-encoding genes. The majority of SIPHT workloads are CPU-heavy but I/O-light. You can review examples of the design of these processes, as well as their full description and characterization. Using the Pegasus workflow generator, five scientific processes from the actual world were generated during experiments (CyberShake, Montage, LIGO, and SIPHT), there are 50, 100, 200, 500, and 1,000 jobs of varying sizes; the suggested model is compared to the existing model, and all prices are estimated in dollars.

### 4.2. Cost evaluation

#### 4.2.1. CyberShake approach

The CyberShake method, which is utilized to generate the seismic hazard probability curve for numerous areas in Southern California, takes an average of 8 hours and 51 minutes to execute. Figure 2 is a depiction of the same topic. In addition, it creates a significant quantity of data and has a high number of jobs; processing such a vast quantity of data demands a tremendous deal of energy, rendering the model inefficient. The suggested model efficiently minimizes energy usage in consideration of ecosystem adoption of practices over time (EAPT). To further evaluate the new model, it is contrasted with the current mechanism.

By considering the four CyberShake method, Table 1 compares the total expenses of the existing system with the proposed system. In addition, the existing model for CyberShake\_30 costs \$22,232.6165 whereas the proposed model costs \$17,497.7745; similarly, the existing model for CyberShake\_50 costs \$24,669.0295 while the proposed model costs \$19,036.5655. The existing model for CyberShake\_100 requires \$30,138.901, but the proposed model only requires \$22,674.6285 milliseconds. The existing model costs \$204,102 to perform the CyberShake\_1000 job, whereas the proposed method costs \$145,983.8115. Here PS stands for proposed system and ES stands for existing system.

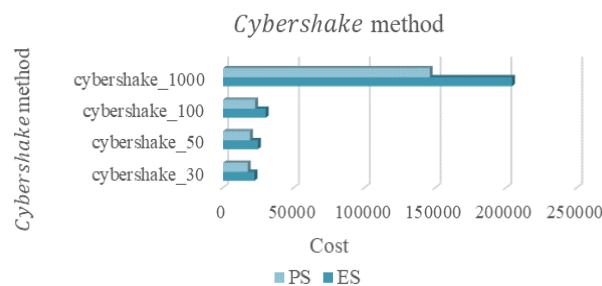


Figure 2. Cost comparison for CyberShake method

Table 1. Cost comparison for CyberShake method

VM	ES	PS	Cost difference (%)
CyberShake_30	22,232.6165	17,497.7745	23.835
CyberShake_50	24,669.0295	19,036.5655	25.775
CyberShake_100	30,138.901	22,674.6285	28.267
CyberShake_1000	204,102	145,983.8115	33.202

**4.2.2. Laser interferometer gravitational wave observatory approach**

The LIGO method is used to generate and analyze gravitational waves from data collected by merging compact binary models. The LIGO workflow is represented in Figure 3. In addition, this process has four separate DAX files; the present model for Inspiral\_30 costs \$22,951.4585, but the proposed model costs \$12,673.1235. Inspiral\_50 and in Inspiral\_100 incur additional costs of \$22,519.9115 and \$40,281.976, respectively, compared to the older models' costs of \$40,788.6025 and \$72,942.9245. The existing model costs 789,741.133 and the proposed model costs 436,056.0895 for Inspiral\_1000. Table 2 compares the values, and Figure 3 illustrates a graph comparing the existing and proposed values.

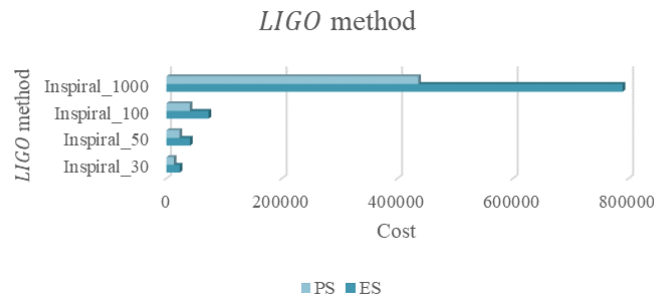


Figure 3. Graphical comparison of LIGO method

Table 2. Cost comparison for LIGO method

VM	ES	PS	Cost difference (%)
Inspiral_30	22,951.4585	12,673.1235	57.704
Inspiral_50	40,788.6025	22,519.9115	57.713
Inspiral_100	72,942.9245	40,281.976	57.692
Inspiral_1000	789,741.133	436,056.0895	57.707

**4.2.3. Montage approach**

With the NASA-developed Montage approach, several images are fed as input and are used to create one-of-a-kind mosaics. The Montage technique requires four different dax files for a cost comparison. The existing model forecasts costs of \$846.8255 and \$1,872.66 for Montage\_25 and Montage\_50, whereas the proposed model, predicts costs of \$482.8945 and \$1,062.9835 for Montage\_25 and Montage\_50, respectively. Figure 4 displays the cost comparison on Montage method. Table 3 displays the cost comparison on Montage technique. The existing model for Montage\_100 and Montage\_1000 costs \$3,944.9945 and \$41,426.565, but the proposed model costs \$2,231.8825 and \$23,398.8515, respectively.

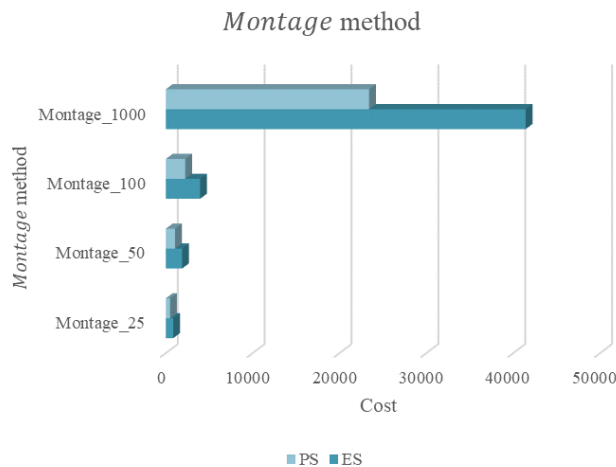


Figure 4. Cost comparison on Montage method



Table 3. Cost comparison on Montage technique

VM	ES	PS	Cost difference (%)
Montage_25	846.8255	482.8945	54.738
Montage_50	1,872.66	1,062.9835	55.162
Montage_100	3,944.9945	2,231.8825	55.469
Montage_1000	41,426.565	23,398.8515	55.619

#### 4.2.4. SIPHT approach

SIPHT is a Harvard project that searches the NCRI database of bacterial replicons for untranslated RNAs. Four SIPHT DAX files, namely SIPHT\_30, SIPHT\_60, SIPHT\_100, and SIPHT\_1000, have been examined. Existing models for SIPHT\_30 and SIPHT\_60 cost \$19,169.0855 and \$40,314.883, while the proposed models cost \$10,565.216 and \$22,216.6525, respectively. Existing models of the SIPHT\_100, and SIPHT\_1000 cost \$60,047.434 and \$599,918.9015, whereas the proposed models cost \$33,091.578 and \$330,580.468 respectively. Figure 5 and Table 4 give comparative cases.

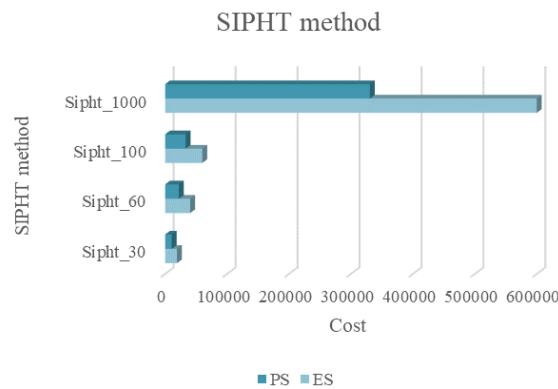


Figure 5. Cost comparisons of SIPHT method

Table 4. Cost comparisons of SIPHT method

VM	ES	PS	Cost difference (%)
SiphT_30	19,169.0855	10,565.216	57.872
SiphT_60	40,314.883	22,216.6525	57.885
SiphT_100	60,047.434	33,091.578	57.883
SiphT_1000	599,918.9015	330,580.468	57.891

#### 4.3. Comparative analysis

In this section, several methods are compared. The cost for PS is 23.835%, 25.775%, 28.267% and 33.202% less expensive than the prior model for CyberShake\_30, CyberShake\_50, CyberShake\_100, and CyberShake\_1000, respectively. Similarly, the LIGO procedure's PS criteria are 57.704%, 57.713%, 57.692%, and 57.707% for Inspiral\_30, Inspiral\_50, and Inspiral\_1000, respectively. For Montage\_25, Montage\_50, Montage\_100, and Montage\_1000, PS is 54.738%, 55.162%, 55.469%, and 55.619% less expensive than the preceding model. The cost difference between ES and PS for SIPHT\_30, SIPHT\_60, SIPHT\_100, and SIPHT\_1000 is \$57.872, \$57.885, \$57.883, and \$57.891, respectively. In addition, the comparative analysis states that the proposed model performs better in comparison with the existing model.

## 5. CONCLUSION

This research work develops HCFT mechanism for cost reduction; HCFT comprises novel clustering and optimal resource utilization mechanism for fault tolerance, which tends to reduce workflow cost. HCFT mechanism is evaluated considering various scientific workflow with its various variant. Furthermore, considering cost as an evaluation parameter, HCFT observes marginal improvisation over the existing model. In case of CyberShake\_30, CyberShake\_50, CyberShake\_100, and CyberShake\_1000, HCFT-mechanism observes 23.83%, 25.77%, and 28.26% respectively. Furthermore, considering the Inspiral workflow, HCFT-mechanism observes improvisation of 57.704%, 57.713%, 57.692%, and 57.707% for Inspiral\_30, Inspiral\_50,

and Inspiral\_1000, respectively. At last, HCFT-mechanism observes improvisation of 57.872%, 57.885%, 57.883%, and 57.891% for SIPHT\_30, SIPHT\_60, SIPHT\_100, and SIPHT\_1000. HCFT observes marginal improvisation; however, considering the complexity of scientific workflow, future work should be focused on the implementation of a machine learning approach to predict the resources efficiently.




## REFERENCES

- [1] X. Li *et al.*, "A novel workflow-level data placement strategy for data-sharing scientific cloud workflows," *IEEE Transactions on Services Computing*, vol. 12, no. 3, pp. 370–383, May 2019, doi: 10.1109/TSC.2016.2625247.
- [2] A. Song, W.-N. Chen, X. Luo, Z.-H. Zhan, and J. Zhang, "Scheduling workflows with composite tasks: a nested particle swarm optimization approach," *IEEE Transactions on Services Computing*, vol. 15, no. 2, pp. 1074–1088, Mar. 2022, doi: 10.1109/TSC.2020.2975774.
- [3] S. Qin, D. Pi, Z. Shao, and Y. Xu, "A knowledge-based adaptive discrete water wave optimization for solving cloud workflow scheduling," *IEEE Transactions on Cloud Computing*, vol. 11, no. 1, pp. 200–216, Jan. 2023, doi: 10.1109/TCC.2021.3087642.
- [4] X. Tang, "Reliability-aware cost-efficient scientific workflows scheduling strategy on multi-cloud systems," *IEEE Transactions on Cloud Computing*, vol. 10, no. 4, pp. 2909–2919, Oct. 2022, doi: 10.1109/TCC.2021.3057422.
- [5] W. Chen, R. F. da Silva, E. Deelman, and T. Fahringer, "Dynamic and fault-tolerant clustering for scientific workflows," *IEEE Transactions on Cloud Computing*, vol. 4, no. 1, pp. 49–62, Jan. 2016, doi: 10.1109/TCC.2015.2427200.
- [6] H. R. Faragardi, M. R. S. Sedghpour, S. Fazliahmadi, T. Fahringer, and N. Rasouli, "GRP-HEFT: a budget-constrained resource provisioning scheme for workflow scheduling in IaaS clouds," *IEEE Transactions on Parallel and Distributed Systems*, vol. 31, no. 6, pp. 1239–1254, Jun. 2020, doi: 10.1109/TPDS.2019.2961098.
- [7] M. Niu, B. Cheng, Y. Feng, and J. Chen, "GMTA: a geo-aware multi-agent task allocation approach for scientific workflows in container-based cloud," *IEEE Transactions on Network and Service Management*, vol. 17, no. 3, pp. 1568–1581, Sep. 2020, doi: 10.1109/TNSM.2020.2996304.
- [8] H. Ghavamipoor, S. A. K. Mousavi, H. R. Faragardi, and N. Rasouli, "A reliability aware algorithm for workflow scheduling on cloud spot instances using artificial neural network," in *2020 10th International Symposium on Telecommunications (IST)*, Dec. 2020, pp. 67–71, doi: 10.1109/IST50524.2020.9345896.
- [9] F. Yao, C. Pu, and Z. Zhang, "Task duplication-based scheduling algorithm for budget-constrained workflows in cloud computing," *IEEE Access*, vol. 9, pp. 37262–37272, 2021, doi: 10.1109/ACCESS.2021.3063456.
- [10] A. Shahidinejad and S. Barshandeh, "Sink selection and clustering using fuzzy-based controller for wireless sensor networks," *International Journal of Communication Systems*, vol. 33, no. 15, Oct. 2020, doi: 10.1002/dac.4557.
- [11] Y. Ding, G. Yao, and K. Hao, "Fault-tolerant elastic scheduling algorithm for workflow in Cloud systems," *Information Sciences*, vol. 393, pp. 47–65, Jul. 2017, doi: 10.1016/j.ins.2017.01.035.
- [12] A. Marahatta, Y. Wang, F. Zhang, A. K. Sangaiah, S. K. S. Tyagi, and Z. Liu, "Energy-aware fault-tolerant dynamic task scheduling scheme for virtualized cloud data centers," *Mobile Networks and Applications*, vol. 24, no. 3, pp. 1063–1077, Jun. 2019, doi: 10.1007/s11036-018-1062-7.
- [13] S. Bharathi, A. Chervenak, E. Deelman, G. Mehta, M.-H. Su, and K. Vahi, "Characterization of scientific workflows," in *2008 Third Workshop on Workflows in Support of Large-Scale Science*, Nov. 2008, pp. 1–10, doi: 10.1109/WORKS.2008.4723958.
- [14] A. Marahatta, Q. Xin, C. Chi, F. Zhang, and Z. Liu, "PEFS: AI-driven prediction based energy-aware fault-tolerant scheduling scheme for cloud data center," *IEEE Transactions on Sustainable Computing*, vol. 6, no. 4, pp. 655–666, Oct. 2021, doi: 10.1109/TSUSC.2020.3015559.
- [15] G. Fan, L. Chen, H. Yu, and D. Liu, "Modeling and analyzing dynamic fault-tolerant strategy for deadline constrained task scheduling in cloud computing," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 50, no. 4, pp. 1260–1274, Apr. 2020, doi: 10.1109/TSMC.2017.2747146.
- [16] Z. Ahmad, A. I. Jehangiri, N. Mohamed, M. Othman, and A. I. Umar, "Fault tolerant and data oriented scientific workflows management and scheduling system in cloud computing," *IEEE Access*, vol. 10, pp. 77614–77632, 2022, doi: 10.1109/ACCESS.2022.3193151.
- [17] L. Zeng, B. Veeravalli, and A. Y. Zomaya, "An integrated task computation and data management scheduling strategy for workflow applications in cloud environments," *Journal of Network and Computer Applications*, vol. 50, pp. 39–48, Apr. 2015, doi: 10.1016/j.jnca.2015.01.001.
- [18] V. Arabnejad, K. Bubendorfer, and B. Ng, "Scheduling deadline constrained scientific workflows on dynamically provisioned cloud resources," *Future Generation Computer Systems*, vol. 75, pp. 348–364, Oct. 2017, doi: 10.1016/j.future.2017.01.002.
- [19] M. A. Rodriguez and R. Buyya, "Budget-driven scheduling of scientific workflows in IaaS clouds with fine-grained billing periods," *ACM Transactions on Autonomous and Adaptive Systems*, vol. 12, no. 2, pp. 1–22, Jun. 2017, doi: 10.1145/3041036.
- [20] M. Ghasemzadeh, H. Arabnejad, and J. G. Barbosa, "Deadline-budget constrained scheduling algorithm for scientific workflows in a cloud environment," in *Leibniz International Proceedings in Informatics, LIPIcs*, 2017, vol. 70, pp. 19.1-19.16, doi: 10.4230/LIPIcs.OPODIS.2016.19.
- [21] F. Juarez, J. Ejarque, and R. M. Badia, "Dynamic energy-aware scheduling for parallel task-based application in cloud computing," *Future Generation Computer Systems*, vol. 78, pp. 257–271, Jan. 2018, doi: 10.1016/j.future.2016.06.029.
- [22] K. Ganga and S. Karthik, "A fault tolerant approach in scientific workflow systems based on cloud computing," in *2013 International Conference on Pattern Recognition, Informatics and Mobile Engineering*, Feb. 2013, pp. 387–390, doi: 10.1109/ICPRIME.2013.6496507.
- [23] I. Casas, J. Taheri, R. Ranjan, L. Wang, and A. Y. Zomaya, "A balanced scheduler with data reuse and replication for scientific workflows in cloud computing systems," *Future Generation Computer Systems*, vol. 74, pp. 168–178, Sep. 2017, doi: 10.1016/j.future.2015.12.005.
- [24] W. Chen and E. Deelman, "Fault tolerant clustering in scientific workflows," in *2012 IEEE Eighth World Congress on Services*, Jun. 2012, pp. 9–16, doi: 10.1109/SERVICES.2012.5.
- [25] Z. Ahmad, A. I. Jehangiri, M. Iftikhar, A. I. Umer, and I. Afzal, "Data-oriented scheduling with dynamic-clustering fault-tolerant technique for scientific workflows in clouds," *Programming and Computer Software*, vol. 45, no. 8, pp. 506–516, Dec. 2019, doi: 10.1134/S0361768819080097.
- [26] Z. Ahmad, B. Nazir, and A. Umer, "A fault-tolerant workflow management system with quality-of-service-aware scheduling for scientific workflows in cloud computing," *International Journal of Communication Systems*, vol. 34, no. 1, Jan. 2021, doi: 10.1002/dac.4649.




- [27] Z. Ahmad, A. I. Jehangiri, M. A. Ala'anzy, M. Othman, and A. I. Umar, "Fault-tolerant and data-intensive resource scheduling and management for scientific applications in cloud computing," *Sensors*, vol. 21, no. 21, pp. 1-19, Oct. 2021, doi: 10.3390/s21217238.
- [28] D. Chakraborty, V. V. Mankar, and A. A. Nanavati, "Enabling runtime adaptation of workflows to external events in enterprise environments," in *IEEE International Conference on Web Services (ICWS 2007)*, Jul. 2007, pp. 1112-1119, doi: 10.1109/ICWS.2007.85.
- [29] E. Deelman *et al.*, "Pegasus, a workflow management system for science automation," *Future Generation Computer Systems*, vol. 46, pp. 17-35, May 2015, doi: 10.1016/j.future.2014.10.008.
- [30] S. H. Sreedhara, V. Kumar, and S. Salma, "Efficient big data clustering using adhoc fuzzy c means and auto-encoder CNN," in *Inventive Computation and Information Technologies*, Singapore: Springer, 2023, pp. 353-368, doi: 10.1007/978-981-19-7402-1\_25.

## BIOGRAPHIES OF AUTHORS



**Chaya T. Doddaiiah**    earned her bachelors of engineering (B.E.) degree in computer science and engineering from VTU, Belagavi in 2007. She obtained her master's degree in M.Tech. (computer science and engineering) from Siddhartha University in 2011 and currently she is a research scholar at University BDT College of Engineering, Davangere, doing her Ph.D. in computer science and engineering. She has more than 10+ years of experience in teaching, worked as an assistant professor and attended many workshops and induction programs conducted by various universities. Her areas of interest are big data analytics and cloud computing. She can be contacted at email: [chaya.td@gmail.com](mailto:chaya.td@gmail.com), [t.chaya@rediffmail.com](mailto:t.chaya@rediffmail.com).



**Dr. Mohamed Rafi**    from Koratagere, Tumkur, obtains B.E., M.E., and Ph.D. in computer science and engineering. He is having more than 25 years of experience in teaching and research. Currently he is working as professor in Department of Studies in Computer Science and Engineering, University BDT College of Engineering, Davangere, a constituent College of Visvesvaraya Technological University, Belagavi, India. Also serving as member of board of studies, Bangalore University, Subject Expert of Doctoral Committee Bangalore University and BOE of Computer Science and Information Science Board, VTU. Worked at Universities of Ethiopia and Saudi Arabia. Published more than 170 papers in conferences, national, and international journals. His area of interest includes digital image processing and managing big data. He can be contacted at email: [mdrafi2km@yahoo.com](mailto:mdrafi2km@yahoo.com).