IEEE Access
Multidisciplinary : Rapid Review : Open Access Journal

Received 1 July 2022, accepted 19 July 2022, date of publication 21 July 2022, date of current version 28 July 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3193151

RESEARCH ARTICLE

Fault Tolerant and Data Oriented Scientific Workflows Management and Scheduling System in Cloud Computing

ZULFIQAR AHMAD[1], ALI IMRAN JEHANGIRI[1], NADER MOHAMED[2], (Member, IEEE), MOHAMED OTHMAN[3,4], (Senior Member, IEEE), AND ARIF IQBAL UMAR[1]

[1]Department of Computer Science and Information Technology, Hazara University, Mansehra 21300, Pakistan
[2]Department of Computer Science and Information Systems, Pennsylvania Western University, California, PA 15419, USA
[3]Department of Communication Technology and Networks, Universiti Putra Malaysia (UPM), Serdang 43400, Malaysia
[4]Laboratory of Computational Sciences and Mathematical Physics, Institute for Mathematical Research (INSPEM), Universiti Putra Malaysia (UPM), Serdang 43400, Malaysia

Corresponding authors: Ali Imran Jehangiri (ali_imran@hu.edu.pk), Nader Mohamed (mohamed@calu.edu), and Mohamed Othman (mothman@upm.edu.my)

ABSTRACT Cloud computing is a virtualized, scalable, ubiquitous, and distributed computing paradigm that provides resources and services dynamically in a subscription based environment. Cloud computing provides services through Cloud Service Providers (CSPs). Cloud computing is mainly used for delivering solutions to a large number of business and scientific applications. Large-scale scientific applications are evaluated through cloud computing in the form of scientific workflows. Scientific workflows are data-intensive applications, and a single scientific workflow may be comprised of thousands of tasks. Deadline constraints, task failures, budget constraints, improper organization and management of tasks can cause inconvenience in executing scientific workflows. Therefore, we proposed a fault-tolerant and data-oriented scientific workflow management and scheduling system (FD-SWMS) in cloud computing. The proposed strategy applies a multi-criteria-based approach to schedule and manage the tasks of scientific workflows. The proposed strategy considers the special characteristics of tasks in scientific workflows, i.e., the scientific workflow tasks are executed simultaneously in parallel, in pipelined, aggregated to form a single task, and distributed to create multiple tasks. The proposed strategy schedules the tasks based on the data-intensiveness, provides a fault tolerant technique through a cluster-based approach, and makes it energy efficient through a load sharing mechanism. In order to find the effectiveness of the proposed strategy, the simulations are carried out on WorkflowSim for Montage and CyberShake workflows. The proposed FD-SWMS strategy performs better as compared with the existing state-of-the-art strategies. The proposed strategy on average reduced execution time by 25%, 17%, 22%, and 16%, minimized the execution cost by 24%, 17%, 21%, and 16%, and decreased the energy consumption by 21%, 17%, 20%, and 16%, as compared with existing QFWMS, EDS-DC, CFD, and BDCWS strategies, respectively for Montage scientific workflow. Similarly, the proposed strategy on average reduced execution time by 48%, 17%, 25%, and 42%, minimized the execution cost by 45%, 11%, 16%, and 38%, and decreased the energy consumption by 27%, 25%, 32%, and 20%, as compared with existing QFWMS, EDS-DC, CFD, and BDCWS strategies, respectively for CyberShake scientific workflow.

INDEX TERMS Cloud computing, scientific workflows, scheduling, load management, CyberShake, Montage.

I. INTRODUCTION

Cloud computing is a virtualized, scalable, ubiquitous, and distributed computing model that provides resources and services dynamically in a subscription based environment. Cloud computing services and resources include servers, network, storage, bandwidth, virtual machines, and processing units [1]–[3]. These services are provided by Cloud Service Providers (CSPs) and are categorized into three major segments: (a) "Infrastructure as a Service" (IaaS), "Platform

The associate editor coordinating the review of this manuscript and approving it for publication was Agostino Forestiero.

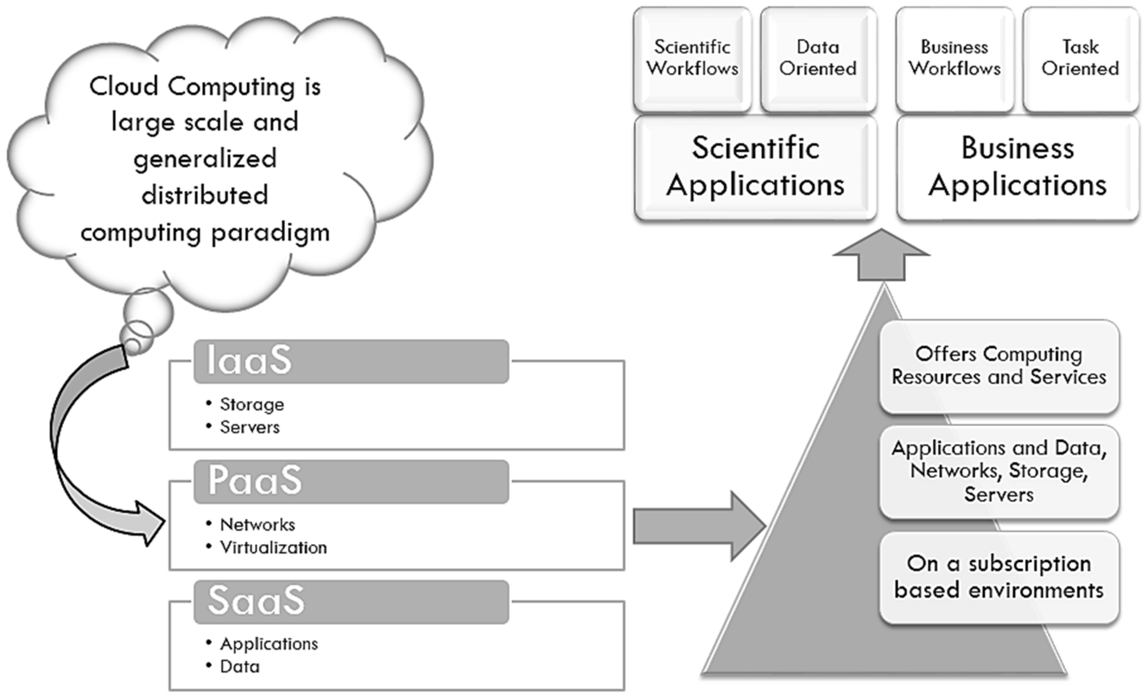

Cloud Computing is large scale and generalized distributed computing paradigm
IaaS
• Storage
• Servers
PaaS
• Networks
• Virtualization
SaaS
• Applications
• Data
Offers Computing Resources and Services
Applications and Data, Networks, Storage, Servers
On a subscription based environments
Scientific Workflows
Data Oriented
Business Workflows
Task Oriented
Scientific Applications
Business Applications

FIGURE 1. Services provided by cloud computing.

as a Service'' (PaaS), and ''Software as a Service'' (SaaS) [4]–[7]. Cloud resources and services are widely used for business and scientific applications [8]. Business applications, such as ''Customer Relationship Management'' (CRM) and ''Enterprise Resource Planning'' (ERP), are task-oriented applications and structured as business workflows. For implementation of business workflows, business models like Amazon EC2 (Elastic Compute Cloud) are used [9]. The cloud computing platforms PaaS and SaaS are significantly used to deploy such type of business applications [10], [11]. On the other hand, large-scale scientific applications, such as Montage [12], [13] and CyberShake [12], [14], are data-oriented scientific applications and organized as scientific workflows. Scientific workflow management systems like Pegasus WMS (Workflow Management System) are used for implementation of scientific workflows [15]. The cloud computing platform IaaS is magnificently used for the deployment of scientific applications [10]. FIGURE 1 represents the services delivered by cloud computing.

Scientific workflows are data-intensive scientific applications that require high computational and storage power for evaluation and computation [12], [16], [17], [18]. In particular, scientific workflow applications are collections of fine-grained computational tasks with various structured activities [4]–[19]. Even a single scientific application, when structured as a scientific workflow, consists of a large number of computational tasks and the dependencies between the tasks also exist. In scientific workflows, each task represents a significant amount of data and requires high computation power. Scientific applications generally include the fields of biology, astronomy, gravitational physics, and earthquake science [12]. For example, a montage 4 degree square scientific workflow consists of 3,027 application tasks, the runtime of which is 85 CPU hours at a cost of \$9 when it is running on 1 processor [20]. Similarly, the CyberShake scientific workflow required 14100 CPU hours to process 755 GB of data [21]. FIGURE 2 shows the characteristics with respect to five basic realistic scientific workflows.

Cloud computing is one of the reliable, effective, and prominent platforms for management, scheduling, and execution of scientific workflows [22]–[25]. However, there are some special characteristics of scientific workflows in respect of data management, scheduling and execution as compared to the traditional tasks and workflows [26], [27]. In scientific workflows, there are a large number of activities and tasks with multiple constraints. For the purpose of management, scheduling and execution of scientific workflows on target resources, various workflow management strategies are used [12], [16], [28]. When scientific workflows are required to be managed and scheduled, there is: (a) collection, classification and management of scientific workflows, (b) management of tasks, (c) resource management, (d) scheduling policies and, (e) fault-tolerant mechanisms, that may be deployed in a system due to which there is a probability of performance degradation [29]. Moreover, some of the workflows are too large and required to be moved from one node to another, resultantly, monetary cost would apply on data movement [30]–[35]. On the other hand, (a) architecture design, (b) integration of cloud infrastructure and resources with scientific workflow systems, (c) computation of workflows, and

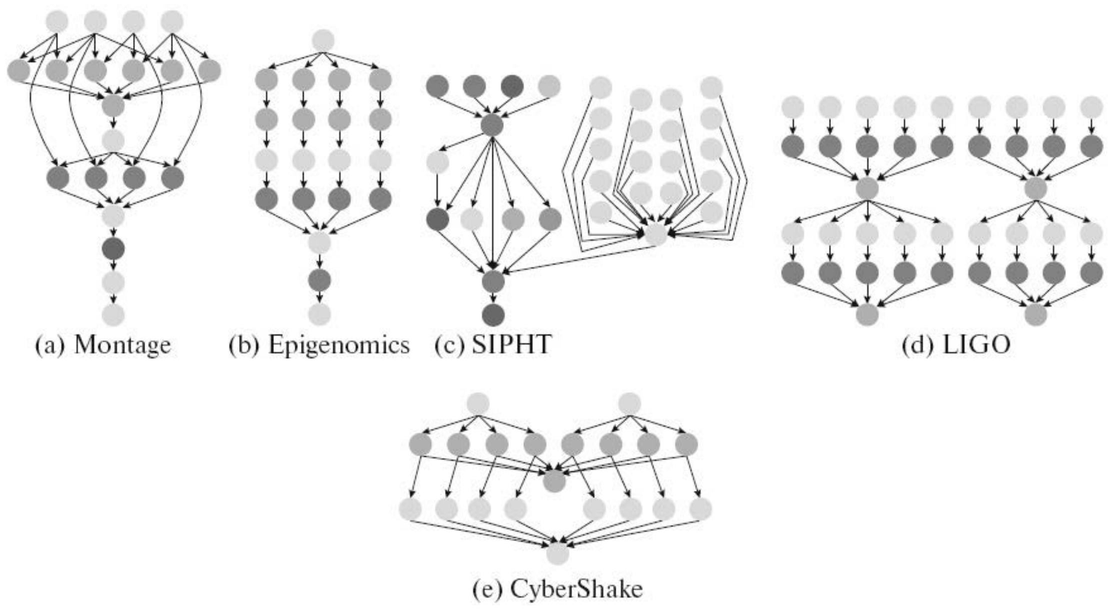


FIGURE 2. **Characteristics of five realistic scientific workflows [12].**

(d) data management, are the principal issues that are need to be resolved effectively and efficiently [15], [9], [36]. All such challenges require to design an effective, well-defined and fault-tolerant scientific workflows management and scheduling system.

More specifically, the research challenges for scientific workflows execution in terms of Workflow Tasks (T_W), Workflow Levels (L_W), and Cloud Resources (R_C) are described as below:

- Suppose, there are "n" number of workflow tasks i.e., TW1, TW2, TW3, ..., TWn in scientific workflows which are required to be executed at multiple workflow levels, $L_{W1}, L_{W2}, L_{W3}, \ldots, L_{Wn}$ on cloud resources, R_{C1}, R_{C2}, R_{C3},..., R_{Cn}. Thus, it requires a workflow management system to organize and manage these workflows.
- Suppose, for each workflow task T_{W1}, T_{W2}, T_{W3}, ..., T_{Wn}, there is predefined data transfer time to each resource and several workflow tasks have diverse requirement of resources from R_{C1}, R_{C2}, R_{C3},..., R_{Cn} at multiple levels T_{W1}, T_{W2}, T_{W3}, ..., T_{Wn}. Thus, require an energy-efficient and data-oriented scheduling policy.
- Several workflow tasks from T_{W1}, T_{W2}, T_{W3}, ..., T_{Wn} are executed at bottleneck node or level, the failure of which makes the whole execution futile. Thus, it requires a fault-tolerant mechanism.
- There are multiple tasks at various levels and sometimes have a similar requirement of services and resources therefore, a cluster-based scheduling and fault-tolerant mechanism are greatly useful for such execution in order to reduce execution time and cost.

In our work, we systematically investigate and solve the above mentioned challenges by presenting: A Data-Oriented and Fault-Tolerant Workflow Management and Scheduling System (DF-WMSS) for Scientific Workflows in Cloud Computing. The main contributions of this research work are given below:

- We proposed a Fault Tolerant and Data Oriented Scientific Workflows Management and Scheduling System (FD-SWMS) in Cloud Computing.
- We provided a data-intensive scheduling method for the execution of scientific workflow tasks in the FD-SWMS strategy.
- We implemented a dynamic re-clustering based fault-tolerant mechanism [37]–[40] in the FD-SWMS strategy.
- We made the proposed FD-SWMS strategy an energy-efficient strategy through a load sharing approach.
- We evaluated the FD-SWMS strategy through simulation in WorkflowSim [41], [42]. We considered the performance evaluation parameters, i.e., execution time, execution cost, budget, deadline, energy consumption, and SLA violation.
- We executed the scientific workflows Montage [43] and CyberShake. We compared the simulation results with the existing four state-of-the-art strategies i.e., QFWMS[44], EDS-DC [6], CFD [45], and BDCWS [46].

The remaining part of the article is organized by providing related work in section II, system design and model in section III, component of the proposed model in section IV, experiments, results and discussion in section V and conclusion in section VI.

II. RELATED WORK

The literature review in respect of the proposed work is thoroughly studied and by considering the proposed system, it is categorized into three parts. In the first part, the basis of scientific workflows and the workflows management system for scientific workflows is reviewed. In the second part, the scientific workflows scheduling mechanisms are studied, while, in the third part of the literature review, the fault-tolerant mechanisms are gone through. At the end of this section, a complete overview of the related work in tabular form is presented.

Starting from the very outset, a detailed study on five realistic scientific workflows for various scientific applications was conducted [12]. These workflows are: (a) cyberShake related earthquake science, (b) SIPHT (sRNA identification protocol using high-throughput technology) associated with biology, (c) montage related to astronomy, (d) epigenomics related to genetics, and (e) LIGO (Laser Interferometer Gravitational Wave Observatory) belongs to gravitational physics. Characteristics of each scientific workflow in respect of structure, data, and computational requirements, are presented in the work. It is also presented that the scientific workflows have some special structural and functional properties in terms of computation. These properties of scientific workflows include pipeline, data parallelism, data distribution and redistribution, data aggregation, and the compositions of scientific workflows. A scalable workflow management system i.e., Pegasus WMS (Workflow Management System) for science automation was described in [15]. It is used to map description of scientific workflow from an abstract level to the distributed computing infrastructures. The system components in Pegasus WMS were described in detail by the authors and then the functioning of the system was built. Pegasus WMS delivers a roadmap for construction of advanced model scientific workflows for their execution and evaluation. Workflow management system in respect of business applications i.e., Heterogeneous Event Management Middleware (HEMM) WMS was presented in [9]. The WMS is responsible for accomplishing inter-enterprise workflow event management. The HEMM WMS can also be suitable for the incorporation of changes in running workflow instances. The WMS is limited only for implementation of business applications and there is no deployment of scientific applications.

For minimizing the financial cost with user-defined deadline constraint, a scheduling approach for scientific workflows i.e., "Dynamic Scheduling of Bag of Tasks based workflows" (DSB) was presented in [4]. The approach initially groups the workflow into Bag of Tasks (BoTs) based on priority constraint and data dependency, and then optimizes the scheduling and allocation of BoTs. For attaining the objectives of the presented method, heterogeneous, elastic and dynamically provisioned cloud resources in the shape of VMs (virtual machines) are used. The presented method considers the cloud platform of pay-as-you-go IaaS (Infrastructure as a Service) with features like elasticity and heterogeneity. Although the approach significantly reduces the cost of workflow computation while meeting the workflow deadline. However, the special features of workflows such as data and compute intensiveness were not considered by the authors. Similarly, the approach also lacks fault tolerance mechanism. Scheduling of scientific workflows with better completion time and utilization of resources efficiently, is an important aspect in order to effectively schedule the tasks of scientific workflows. Thus, a scheduling technique, i.e., Adaptive Data Aware Scheduling (ADAS) was presented in [30] that focuses on utilization of resources and workflow completion time. It is a data and integrated task management technique in the cloud environment for various types of workflows. The presented work is an efficient scheduling technique however, it lacks the fault tolerance begin important aspect in workflow scheduling. In order to reduce the cost of computation and provide a deadline constrained based scheduling mechanism, two algorithms were presented in [47]. These algorithms are: (a) PDC (Proportional Deadline Constrained) and (b) DCCP (Deadline Constrained Critical Path). Both the algorithms address the workflow scheduling problem by providing cloud resources dynamically. The first algorithm i.e., PDC maximizes the parallelism in workflow by separating the tasks into various logical levels and then the deadline is divided proportionally among all the levels. Similarly, the second algorithm i.e., DCCP also works with similar manner as of PDC, however, additionally the DCCP also finds the constrained critical path among the workflow for co-locating the tasks on the same instance that communicate to each other. The algorithms were very useful in respect of deadline-aware workflows scheduling, however, the issues of budget-aware scheduling and provision of fault-tolerant mechanisms are still intact.

In order to optimize workflow execution time under a budget constraint, a scheduling algorithm BAGS (Budget-driven Algorithm for Generating high quality Schedules) was presented in [36]. The algorithm BAGS initially distributes budget among the tasks and then dynamically makes resource provisioning and scheduling decisions in order to consider environmental changes. The approach although works well in case of budget constraints scheduling of scientific workflows, however, it still requires enhancement in respect of deadline constraints and fault tolerance mechanism. A heuristic workflows scheduling algorithm i.e., DBWS (Deadline-Budget Workflow Scheduling was presented in [48]. The scheduling algorithm DBWS considers time and cost as two important constraints in order to schedule the tasks of scientific workflows. The approach simply finds a feasible schedule map that satisfies the values of user-defined budget and deadline constraints. However, provision of fault-tolerance mechanism and data intensiveness of scientific workflows has not been considered by the authors.

Scientific workflows consist of data and compute intensive tasks that when executed maximum energy is being consumed. As such for minimization of energy consumption, a real-time dynamic scheduling system was presented in [49] for execution of task-based applications efficiently. The algorithm was titled as; MHRA (a multi-heuristic resource allocation). In order to attain the goals of minimum energy-consumption and less execution-time, the authors presented a polynomial-time algorithm in which a resource allocation technique is combined with a set of heuristic rules. The algorithm although provides a better solution with regards to minimization of energy-consumption and reducing the execution-time, however, the provision of fault-tolerant mechanism being an important feature is not considered by the authors. In addition to the above stated scheduling policies, various heuristic approaches such as MCT (Minimum Completion Time) [50], Max-min (Maximum-minimum) [51] and Min-min (Minimum-minimum) [51], were also used for scheduling the tasks of workflows. The scheduling policy MCT gives priority to the task with minimum completion time and then assigns it to the required resource for execution. The scheduling policy Max-min, executes the large task firstly on a resource with minimum execution time for that task. Whereas, the scheduling policy Min-min, executes the small task firstly on a resource with minimum execution time for that task. The Max-min scheduling policy leads to the delay of smaller tasks, while the Min-min scheduling policy leads to the delay of larger tasks.

In [52], it was contended that failures occur in a cloud computing system when it is keep away from fault tolerant mechanisms in order to maintain the financial profit or to reduce the overhead. The authors in this paper presented the ''Heterogeneous Earliest Finish Time'' (HEFT) scheme through which the authors used the proactive and reactive methods in order to provide the hybrid fault tolerant technique for cloud computing systems. The fault tolerance in cloud computing is generally categorized into proactive, reactive and resubmission mechanisms [53]. The proactive mechanism needs more information about cloud computing and it works in a probabilistic manner. The proactive strategy in cloud computing reduces the failure time and also increases throughput and capacity. In proactive technique, preemptive-migration and software-rejuvenation strategies are commonly used. While, reactive fault tolerance minimizes the impact of failure on execution of applications. In reactive technique, check-pointing and task-replication strategies are commonly used. For scientific workflow systems, it was stated that the resubmission fault-tolerant mechanism is suitable for execution of scientific workflows. Whenever a failed task is detected, it is resubmitted either to the same resource or to another at runtime [53]. When scientific workflows are executed, there are several tasks that require to be executed at bottleneck nodes and their failure makes the whole execution futile, therefore, provision of fault-tolerant mechanism in scientific workflow scheduling is greatly significant for such execution. As such, a dynamic fault-tolerant scheduling algorithm i.e., FASTER (**f**ault-toler**a**nt **s**cheduling algorithm for real-**t**ime sci**e**ntific wo**r**kflows) was presented in [32]. There are three key features of FASTER scheduling algorithm i.e., (a) task backward shifting method in order to maximum use of idle resources, (b) implementation of vertical and horizontal scaling-up technique for providing quick resources to a burst of workflows, and (c) implementation of vertical scaling-down approach in order to avoid unnecessary and ineffective resource utilization. The FASTER algorithm provides better utilization of resources with backward shifting of tasks, scaling-up and scaling-down approaches. However, time and cost constraints being two major research challenges for execution of scientific workflows were not considered by the authors.

With the intention to optimize the scheduling of scientific workflows, a BaRRS (Balanced and file Reuse-Replication Scheduling) algorithm was presented in [54] for cloud computing environments. The algorithm BaRRS divides the given scientific workflows into multiple sub-workflows for the purpose of balancing the system utilization through parallelization. Moreover, replication and data reuse techniques were also exploited in order to improve the amount of data that requires to be transferred among tasks during run-time. The algorithm BaRRS is an effective approach in respect of efficient utilization of resources and implementation of replication and data reuse techniques. However, depending on the nature and specification of scientific workflows, besides parallelism there are also some other properties of scientific workflows such as pipelining, integration and disintegration, and the same were not considered by the authors. Scientific workflow applications are collections of fine-grained computational tasks that are executed at various levels. At each level, the tasks require homogeneous types of services, therefore, fault-tolerant clustering approaches are useful for such computation. Thus, an ensemble of tasks mechanism, i.e., ''Fault-Tolerant Clustering'' (FTC) for scientific workflow was introduced in [39]. The presented work provides three methods, i.e., DC (Dynamic-Clustering), SR (Selective Re-Clustering), and DR (Dynamic Re-Clustering). The first method DC keeps the clustering factor dynamically and as per the failure rate of the detected tasks. The second method SR retries the tasks which are failed within a job. The last method DR combines both the first two methods and it not only keeps the clustering factor dynamically according to the failure rate of detected tasks but also retries the failed tasks of a job.

An ''Enhanced Data-oriented Scheduling strategy with Dynamic clustering fault-tolerant technique'' (EDS-DC) is presented in [6]. The authors implemented a Data-oriented scheduling in EDS-DC. They used Dynamic-clustering fault-tolerant technique in EDS-DC. They performed simulation in WorkflowSim and compared results with three renowned scheduling policies, i.e., (a) MCT-DC, (b) Max-min-DC, and (c) Min-min-DC. Simulation results show that EDS-DC reduced make-span and cost significantly as compared with existing strategies. A ''Quality of Service aware Fault tolerant Workflow Management System'' (QFWMS) for scientific workflows is given in [44]. There are four core components of QFWMS i.e., (a) workflow admission, (b) workflow mapper, (c) workflow scheduler, and (d) workflow engine. With the combination of two real time scientific workflows, the fault tolerant technique and a scheduling policy, two scenarios were defined. The simulations were performed on WorkflowSim and the result shows that the proposed strategy performed better as compared with the existing strategies. A "cluster-based, fault-tolerant and data-intensive" (CFD) strategy for scientific applications in a cloud environment is provided in [45]. The proposed CFD strategy gives a detailed procedure from scientific data submission to the generation of results. The experiments were performed on Montage workflow and the results of the CFD strategy shows that the CFD strategy performed better as compared with the existing strategies. A novel algorithm i.e., Budget-Deadline Constrained Workflow Scheduling (BDCWS) for multi-Quality of Service constrained workflow scheduling such as cost and time is presented in [46]. The algorithm defines the task optimized available budget with the help of computational cost of the task on the slowest VM and then finds the optimistic spare budget. It then generates the set of affordable VMs according to the task optimized available financial budget to control the range of VM selection, and thus effectively controls the task computation cost. The authors experimentally implemented the algorithm and the results showed that the BDCWS performed better than the existing strategies.

As reflected from the literature review, the scientific workflow applications are data and compute intensive applications with special features of pipelining, parallelism, integration and disintegration. Therefore, the scientific workflow applications require high storage and computational power services and resources with data-oriented and fault-tolerant workflow management and scheduling system.

The literature review reflects that several scientific workflow management and scheduling strategies such as: DSB [4], EDS-DC [6], HEMM [9] Pegasus WMS [15], ADAS [30], FASTER [32], BAGS [36], FTC [39], QFWMS [44], CFD [45], BDCWS [46], PDC & DCCP [47], DBWS [48], MHRA [49], and BaRRS. [54] were presented. These strategies managed and scheduled the tasks of scientific workflow applications in various aspects such as; by providing workflow management systems, integrated tasks management, energy-efficient and fault-tolerant scheduling mechanisms. However, there are following limitations in the existing work. The scientific workflows are highly data and compute intensive applications composed of a large number of tasks, thus, for collection, classification and management of scientific workflows, an energy efficient data-oriented scheduling based workflow management system is required. Several workflow tasks are executed at bottleneck node or level and their failure makes the whole execution futile, thus, a fault-tolerant mechanism is required. Multiple workflow tasks at various levels have a similar requirement of services and resources therefore, a cluster-based scheduling and fault-tolerant mechanism works efficiently for execution of scientific workflows.

All such limitations lead to an energy-efficient, data-oriented and fault-tolerant scientific workflow management and scheduling system.

The summary of related work compared with the proposed strategy is shown in Table 1.

III. SYSTEM DESIGN AND MODEL

This section presents the system design and model of a dynamic re-clustering based data oriented scientific workflows management and scheduling strategy. It is an energy efficient, fault tolerant and data oriented resource scheduling and management strategy for scientific workflows. The proposed strategy is termed as: a Fault Tolerant and Data Oriented Scientific Workflows Management and Scheduling System (FD-SWMS) in cloud computing. In FD-SWMS as reflected from FIGURE 3, the resources are obtained from the cloud environment in terms of IaaS (Infrastructure as a Service). The FD-SWMS strategy schedules the workflows tasks by the process in which it finds the best suitable resource by calculating the minimum data transfer time of a task to the resource. By integrating a dynamic re-clustering method, the FD-SWMS guarantees fault tolerance. Dynamic re-clustering is a key fault tolerant strategy in which related jobs in a workflow are first merged together as a cluster and executed on a single resource. After that, it locates unsuccessful tasks in a cluster, dynamically re-clusters them with a factor of "k" based on task failure rate, and re-executes them. The load sharing method implemented by the FD-SWMS reduces energy usage. It calculates the use of all resources, ranks them in ascending order, and then dynamically transfers the load on the resource with the lowest utilization, to the other nodes. The FD-SWMS strategy works with the following steps:

- A scientific workflows management and scheduling system is designed with multiple components such as: workflow generator, workflow manager, workflow scheduler, workflow engine and workflow load manager.
- Data oriented scheduling is done by assigning the task to a resource with minimum data transfer time.
- Dynamic re-clustering based fault tolerant mechanism is implemented to make the proposed strategy a fault tolerant aware strategy.
- Load sharing based energy efficient optimization is done by obtaining the utilization of resources, making them in ascending order and then sharing the load of nodes with minimum utilization to the other nodes.
- The proposed FD-SWMS strategy is simulated with WorkflowSim and evaluated in terms of performance evaluation parameters such as: (a) time, (b) cost, (c) budget, (d) deadline, (e) energy consumption, and f) SLA violation.
- Two well-known realistic scientific workflows i.e., Montage, and (b) CyberShake are executed. The simulation results are compared with existing four state-of-art

TABLE 1. Summary of related work and comparison with proposed strategy.

References	Scheduling Policy	Fault Tolerance	Workflow Management	Performance Evaluation Parameters			
				Time/ Deadline	**Cost/ Budget**	**Energy Efficiency**	**SLA Violation**
[4]	DSB	×	✓	✓	✓	×	×
[6]	EDS-DC	✓	✓	✓	✓	×	✓
[9]	HEMM	×	✓	×	×	×	×
[12]	×	×	×	×	×	×	×
[15]	Pegasus WMS	×	✓	×	×	×	×
[30]	ADAS	×	✓	✓	×	×	×
[32]	FASTER	✓	✓	×	×	×	×
[36]	BAGS	×	✓	×	✓	×	×
[39]	FTC	✓	✓	×	×	×	×
[44]	QFWMS	✓	✓	✓	✓	×	✓
[45]	CFD	✓	✓	✓	✓	×	✓
[46]	BDCWS	×	×	✓	✓	×	×
[47]	PDC & DCCP	×	✓	✓	✓	×	×
[48]	DBWS	×	✓	✓	✓	×	×
[49]	MHRA	×	✓	✓	×	✓	×
[54]	BaRRS	✓	✓	✓	×	×	×
Proposed Strategy	FD-SWMS	✓	✓	✓	✓	✓	✓

strategies i.e., QFWMS [44], EDS-DC [6], CFD [45], and BDCWS [46].

The proposed system is scalable as it utilizes cloud computing resources, and undoubtedly, cloud computing resources are considered ubiquitous and scalable resources. The proposed system works under the following assumptions:

- Resources are acquired from the cloud in terms of infrastructure as a service.
- Resources are managed through the workflow management system.
- Clustering is a term used for integrating similar tasks into a group for execution.
- The average budget and deadline for each type of scientific workflow are predefined.

The proposed FD-SWMS strategy is reliable and an efficient system in such a way that it manages the scientific workflows. It implements data oriented resource scheduling. It ensures dynamic re-clustering based fault tolerant methods and reduces energy consumption by load sharing mechanism. Resources and services are obtained from cloud computing in terms of IaaS.

The proposed FD-SWMS strategy in terms of pseudo-code is presented as below:

The FD-SWMS technique is a component-based workflow resource scheduling and management strategy. The key elements of FD-SWMS strategy are workflow generator, workflow mapper, workflow scheduler, workflow engine, and workflow load manager. The working of all of these elements are based on sequential procedures. The pseudo-code of the proposed strategy consists of "for-loop" within a "while loop," hence, its time and space complexity is quadratic.

The proposed strategy is a multi-objective optimization system. These objectives are to make the scheduling data-intensive, to provide cluster-based fault-tolerance, and to make the scheduling energy-efficient. All these phases work together simultaneously and optimize the multi-criteria-based performance of the system. The proposed strategy can be implemented in practice through a cloud based system. For the submission of scientific data generated by the scientist in various fields of science, an application interface such as Perl or Hubzero is required. The cloud computational and data resources are required to execute it, and after generating the result, it will be returned to the user.

The comparison was made with the four published strategies, i.e., QFWMS [44], EDS-DC [6], CFD [45], and BDCWS [46]. These strategies were considered for comparison based on the dataset, data-intensive scheduling, and fault tolerance. The first three strategies are the most relevant and latest strategies in respect of fault-tolerant based data-intensive scheduling and the dataset they consider. The fourth strategy is the relevant latest strategy as it works with the budget and deadline constraints, and the proposed strategy also considers the budget and deadline constraints with a similar dataset.

So far as the novelty of the proposed FT-SWMS strategy is concerned, it is a novel contribution towards the scheduling of scientific workflows. There is a large variety

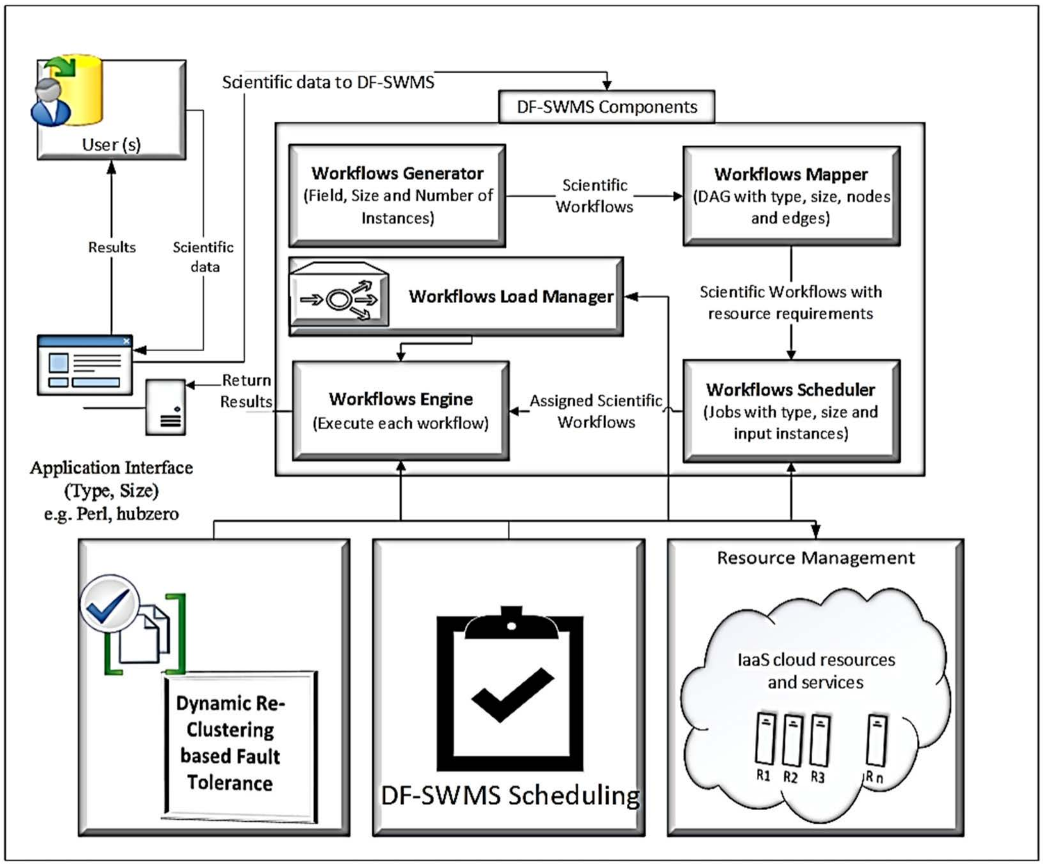

Scientific data to DF-SWMS
DF-SWMS Components
User (s)
Results
Scientific data
Workflows Generator (Field, Size and Number of Instances)
Scientific Workflows
Workflows Mapper (DAG with type, size, nodes and edges)
Workflows Load Manager
Scientific Workflows with resource requirements
Return Results
Workflows Engine (Execute each workflow)
Assigned Scientific Workflows
Workflows Scheduler (Jobs with type, size and input instances)
Application Interface (Type, Size) e.g. Perl, hubzero
Dynamic Re-Clustering based Fault Tolerance
DF-SWMS Scheduling
Resource Management
IaaS cloud resources and services
R1
R2
R3
R n

FIGURE 3. **Component based FT-SWMS system model.**

of scientific workflow tasks. They may be of huge size, normal size, or very small size. The tasks are executed in a pipeline, in parallel, sometimes distributed after execution, and in some situations, merged during execution. Some tasks may be executed at a bottleneck node, the failure of which makes the whole execution fruitless. The data oriented workflow tasks consumed more power during execution. The proposed FT-SWMS considers all these challenges through a multi-criteria approach, i.e., data-intensive scheduling, dynamic re-clustering based fault tolerance, and load balancing mechanism. Each criterion is itself a unique contribution towards the scheduling of scientific workflow tasks. The data-intensiveness, although being considered by the existing work, is not emphasized in the variety, velocity, and volume of the tasks. The dynamic re-clustering fault tolerant mechanism not only provides the fault tolerance but also provides the clustering mechanism of similar tasks in a workflow, which further improves the execution time and cost. The load balancing approach is the most important mechanism to reduce energy consumption for highly data-intensive tasks that are executed on heterogeneous cloud resources. The other recent published strategies have not specifically addressed the issues of scientific workflow management and scheduling by considering the data intensiveness, task variety, and bottleneck failures of tasks in scientific workflows.

IV. COMPONENTS OF FD-SWMS STRATEGY

The components of FD-SWMS strategy are given below:

A. USER

An entity that might be an organization or a person is referred to as a user. If the user is an association, it can collect scientific data from individuals and submit it for execution and review. Users can submit a single type of scientific data or a combination of multiple types of scientific data. After successful execution, the user will get the results in required from.

B. APPLICATION INTERFACE

The Application Interface such as Perl [55] and hubzero [15] are used to connect the user to the FD-SWMS strategy. Through the application interface, the user inputs scientific data for execution and assessment. The Application Interface transmits data to the next part of the model, Workflow Generator, for transforming scientific data to the scientific workflow as DAG (Directed Acyclic Graph) [56] with type, size, nodes and edges. The Application Interface provides an interface to the FD-SWMS strategy for one or more users, as well as one or more scientific workflows can be submitted to FD-SWMS strategy.

Algorithm 1 : FD-SWMS Strategy

```
        Input:    $\alpha$ (Scientific data)

        Output:          $\beta$ (Executed results)

Procedure: FD-SWMS ($\alpha$)
1.     $Q_1 \leftarrow \alpha$ (store scientific data into the queue)
2.     WorkflowGenerator( )
3.     $Q_2 \leftarrow$ Get$_{Workflows}$( ) as $W_1$ to $W_n$ from Q1
4.     for (each workflow $W_1$ to $W_n$) do
5.         WorkflowMappper( )
6.         Generate tasks $T_1$ to $T_n$ for each workflow
       with resource requirements
7.     end for
8.     WorkflowScheduler( )
9.     Get$_{resources}$( ) as $R_1$ to $R_n$
10.     while (WorkflowScheduler( )) do
11.           Generate tasks $T_1$ to $T_n$ for each workflow
12.           for (each task $T_1$ to $T_n$ of workflows $W_1$
        to $W_n$) do
13.                 Find resource Ri with minimum
        data transfer time for task Ti and assign:
14.                 $R_i \leftarrow T_i$
15.           end for
16.     end while
17.     Start Execution. WorkflowEngine( )
18.     while (WorkflowEngine( )) do
19.           if (execution failed)
20.           Dynamic Re-clustering ( )
21.           else
22.           GenerateResults( )
23.           end if else
24.     end while
25.     for (all resources $R_1$ to $R_n$) do
26.           $R_U \leftarrow$ GetUtilization( )
27.           if ($R_{U(i)}$ == Null ) then
28.                 TurnOff($R_{(i)}$)
29.           else if ($R_U$ <= Lower Threshold ) then
30.                 Transfer tasks of $R_{(i)}$ to nearest
          alternative node
31.                 TurnOff($R_{(i)}$)
32.           end if else
33.     end for
34.     $\beta \leftarrow$Executed results
35.     Return $\beta$
```

C. WORKFLOW GENERATOR

Scientific data is received by the Workflow Generator via an application interface created by scientists for the execution and assessment of scientific applications. For scientific data, the Workflow Generator produces a scientific workflow as a Directed Acyclic Graph (DAG) [57]. The workflow generator operates data acquired from scientific processes to create authentically synthetic workflows. At this stage the structure of scientific workflow is represented by equation 1.

$$DAG\,(G) = (T_W, D) \tag{1}$$

where, G is directed acyclic graph with the properties: $T_W = T_{W1}, T_{W2}, \ldots, T_{Wn}$ represents number of tasks and $D = \{(T_{W(i)}, T_{W(j)})|\ T_{W(i)}, T_{W(j)} \in T_W\}$ represents the dataflow dependencies set between the tasks. The dataflow dependency set $(T_{W(i)}, T_{W(j)})$ shows that there are precedence constraints between task $T_{W(i)}$ and task $T_{W(j)}$. The task $T_{W(i)}$ is represented as the immediate predecessor of task $T_{W(j)}$ and $T_{W(j)}$ is represented as the immediate successor of task $T_{W(i)}$. The predecessor and successor tasks are shown in the equations 2 and 3.

$$Pre\left(T_{W(j)}\right) = \{\left(T_{W(i)}|T_{W(i)}, T_{W(j)}\right) \in D\} \tag{2}$$

$$Succ\left(T_{W(i)}\right) = \{\left(T_{W(j)}|T_{W(i)}, T_{W(j)}\right) \in D\} \tag{3}$$

A task with no predecessor task is termed as an entry task and the task with no successor is termed as the exit task. The entry and exit tasks are shown in equations 4 and 5 respectively.

$$Pre\left(T_{W(i)}(entry)\right) = \varphi \tag{4}$$

$$Succ\left(T_{W(j)}(exit)\right) = \varphi \tag{5}$$

A single user can submit a workflow with one or more jobs/inputs that produce a single result/output. The Workflow Generator receives scientific data, which may consist of one or more than one field as requested by the user. It separates the scientific data into their respective scientific workflows, creates their DAG, and submits them to the Workflow Mapper, which is the next part of the model.

D. WORKFLOW MAPPER

Workflow Mapper takes one or more workflows from Workflow Generator and creates executable workflows for each of them with resource requirements. The workflow mapper reconstructs the workflow in order to optimize the performance. It specifically merges the small tasks of a workflow into a job so that the overhead can be reduced. Consequently, after workflow mapper the job is a single unit of execution with multiple tasks [58]. The tasks are organized in terms of dependencies between them. These tasks will be executed sequentially, in parallel, in a distributed manner, and in an integrated manner. At this stage, the parameters of each workflow are the kind of workflow, its size, input instances, and output. Another purpose of the workflow mapper is to create a workflow from data in the form of jobs/inputs that have compute and storage resource requirements, so that these jobs/inputs may be sent to the next component. The Mapper sends all of the jobs in a scientific workflow to the next portion to be scheduled.

E. WORKFLOW SCHEDULER

Workflow Scheduler gets jobs with multiple tasks from the Workflow Mapper and schedules them by allocating resources. The resources are retrieved from the cloud using Infrastructure as a Service (IaaS) and then scheduled/managed separately using the FD-SWMS strategy. Workflow Scheduler also turns jobs into tasks and then assigns resources. The allocation of resources is done in such a way that the tasks consume minimum execution time at the lowest cost. It is achieved by considering the list of tasks and resources. It finds the resource for each task with minimum data transfer time. For each task, the data transfer time of a

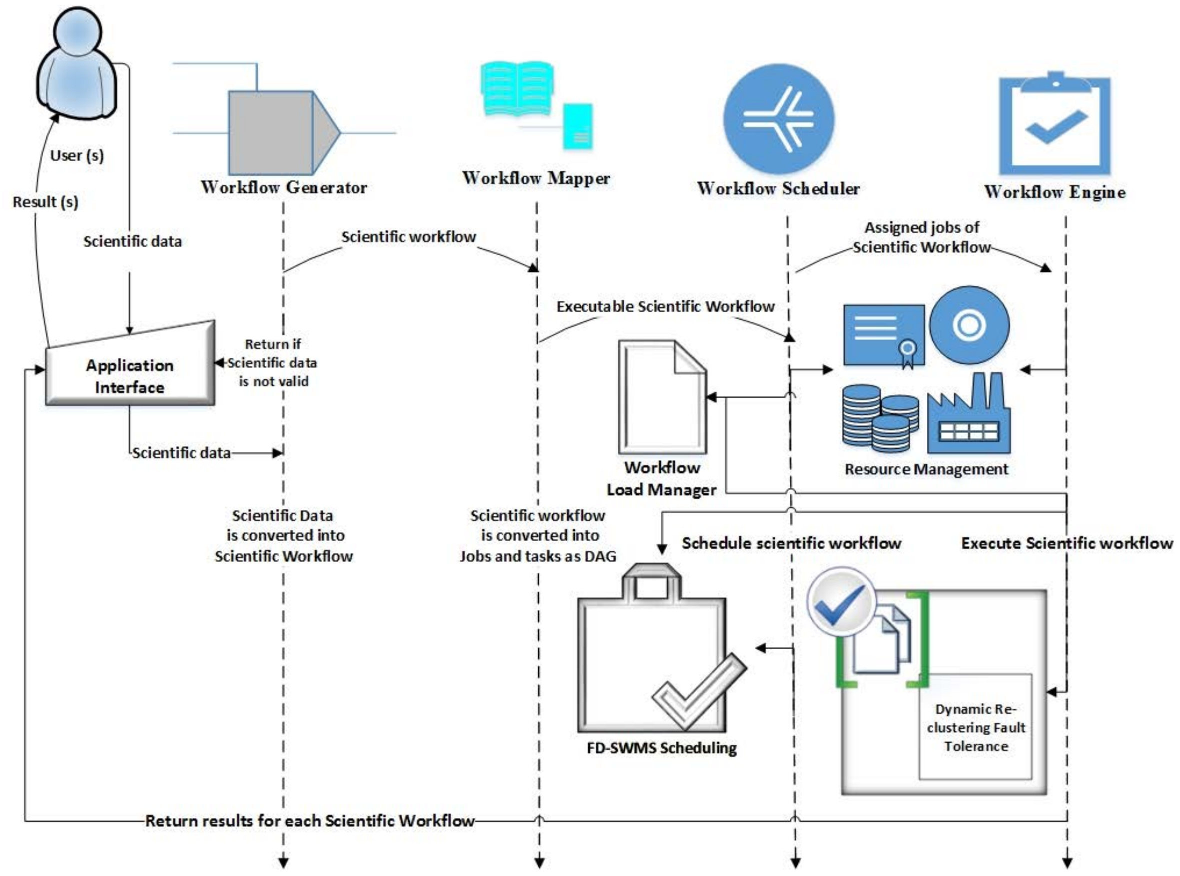

User (s)
Result (s)
Scientific data
Workflow Generator
Workflow Mapper
Workflow Scheduler
Workflow Engine
Scientific workflow
Assigned jobs of Scientific Workflow
Executable Scientific Workflow
Application Interface
Return if Scientific data is not valid
Workflow Load Manager
Resource Management
Scientific data
Scientific Data is converted into Scientific Workflow
Scientific workflow is converted into Jobs and tasks as DAG
Schedule scientific workflow
Execute Scientific workflow
FD-SWMS Scheduling
Dynamic Re-clustering Fault Tolerance
Return results for each Scientific Workflow

FIGURE 4. Dataflow Model of proposed FT-SWMS strategy.

task to all the resources is calculated, and then the task is assigned to the resource with the minimum data transfer time. Equation 6 demonstrates the concept of finding a resource with minimum data transfer time.

$$Resource\left(\mathrm{R}_{\mathrm{C(i)}}\right) = Task\{\mathrm{T}_{\mathrm{W(i)}}|\mathrm{T}_{\mathrm{W(i)}} \longleftarrow Min.D.T.Time\left(\mathrm{R}_{\mathrm{C(i)}}\right))\} \tag{6}$$

where $\mathrm{R}_{\mathrm{C(i)}}$ represents resource 'i', $\mathrm{T}_{\mathrm{W(i)}}$ represents task 'i' and Min.D.T.Time represents minimum data transfer time of task 'i' for resource 'i'. Finally, the mapped list of tasks to resources is obtained by assigning the best available resource to each task with a minimum data transfer time.

F. WORKFLOW ENGINE

Workflow Engine executes the jobs/tasks that Workflow Scheduler has allocated to them on their allotted resources. Workflow Engine also starts the fault-tolerant mechanisms in such a manner that if each job/task is completed successfully, the results are generated and returned to the user via the application interface. Workflow Engine commences the fault-tolerant approach and retries or re-executes the unsuccessful tasks/jobs if the execution of the jobs/tasks fails. Workflow Engine produces results if the execution is successful. Workflow Engine initiates selective re-clustering based on a fault-tolerant method if job/task execution fails. In our scenario, we assume a 5% failure rate based on the total number of jobs, therefore the fault-tolerant approach is used every time the process is submitted.

G. WORKFLOW LOAD MANAGER

An energy aware optimization is done by load sharing mechanism through workflow load manager. It gets resource usage, arranges them in ascending order, and then distributes the load of nodes with the lowest utilization to the remaining nodes. Equation 7 categorized the utilization of resources into three parts i.e., normal, lower threshold and upper threshold.

$$Utilization\left(\mathrm{R}_{\mathrm{C(i)}}\right) = \begin{cases} Normal(Utilizatin = 40\% \ to \ 80\%) \\ L.T.H(Utilizatin = 0\% \ to \ 40\%) \\ U.T.H(Utilizatin = 80\% \ to \ 100\%) \end{cases} \tag{7}$$

where L.T.H represents the resources with lower threshold utilization and U.T.H represents the resources with upper threshold utilization. The resources with normal utilization are considered to be available for transfer of load from the resources with lower threshold utilization. The load on resources with lower threshold utilization is transferred to the resources with normal utilization to make the lower threshold resources as null utilization resources and then the resources with null utilization are turned off.

H. FLOW MODEL OF FD-SWMS STRATEGY

The flow of data in FD-SWMS strategy is shown in FIGURE 4. Through an application interface, one or more users input scientific data to FD-SWMS strategy. The first basic feature of FD-SWMS strategy, Workflow generator, is applied to scientific data. It translates scientific data into a workflow and sends it to the workflow mapper, which is the next key part. When a workflow mapper gets a scientific workflow, it converts it into an executable scientific workflow by identifying jobs/tasks with the appropriate resource requirements. The jobs/tasks of each workflow are then given to the workflow scheduler, the next component. Workflow scheduler sends jobs/tasks of each workflow to the suitable resources based on the scheduling policy. The next component is the workflow engine. It obtains the given jobs/tasks, executes them and returns the results to the user via an application interface. When jobs/tasks fail to execute, a dynamic re-clustering based fault-tolerant mechanism is implemented. The workflow load manager shared the load based energy efficient optimization. It obtains the utilization of resources, making them in ascending order and then shares the load of nodes with minimum utilization to the other nodes.

I. SCHEDULING IN FD-SWMS STRATEGY

Scheduling in FD-SWMS is done by workflow scheduler. It receives jobs/inputs from the Workflow Mapper and schedules them by allocating resources accordingly. Infrastructure as a Service (IaaS) cloud service model is used to get the resources from the cloud, and the FD-SWMS is used to plan and manage them individually. Workflow Scheduler also distributes resources after converting jobs into tasks. The allocation of resources is done in such a way that they take the least amount of time to complete and cost the least amount of money. It is accomplished by taking into account the list of tasks and resources. The mapping list of tasks to resources is then created by allocating the best available resource with the shortest data transfer time to each job.

J. FAULT TOLERANCE IN FD-SWMS STRATEGY

By integrating a dynamic re-clustering method, the FD-SWMS guarantees fault tolerance. Dynamic re-clustering is a key fault tolerant strategy in which related tasks of a job in a workflow are first merged together as a cluster and executed on a single resource. After that, it locates unsuccessful tasks in a cluster, creates dynamic re-clusters based on task failure rate, and re-executes them. Equations 8 and 9 represent the mechanism of dynamic re-clustering.

$$\begin{aligned} &\mathit{Execute}\left(\int_{i=i}^{n} \mathrm{T}_{\mathrm{W(i)}}\right) \\ &\quad \longleftarrow \mathit{Cluster}\left(\int_{i=i}^{n} \mathrm{T}_{\mathrm{W(i)}}\right) \end{aligned} \tag{8}$$

$$\begin{aligned} &\mathit{Execute}\left(\int_{i=j}^{n} \mathrm{T}_{\mathrm{W(j)}}\right) \\ &\quad \longleftarrow \mathit{DynamicRe} \\ &\qquad -\mathit{cluster}\left(\int_{j=i}^{n} \mathrm{T}_{\mathrm{W(j)}}\left(\mathit{Failed}\left(\int_{i=i}^{n} \mathrm{T}_{\mathrm{W(i)}}\right)\right)\right) \end{aligned} \tag{9}$$

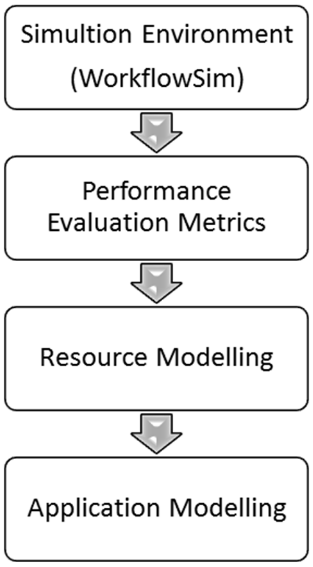
Equation 8 represents the clustering of ‘n’ numbers of similar tasks and executing them. Thereafter, equation 9 represents the finding of failed tasks and dynamically re-cluster them for execution.

K. LOAD MANAGEMENT IN FD-SWMS STRATEGY

The load sharing method implemented by the FD-SWMS strategy reduces energy usage. It calculates the use of all resources, arranges them in ascending order, and then dynamically transfers the load on the resources with the lowest utilization to the other nodes. It categorizes the utilization of resources into three parts i.e., normal, lower threshold and upper threshold. It considers the resources with normal utilization to be available for transfer of load from the resources with lower threshold utilization. The load on resources with lower threshold utilization is transferred to the resources with normal utilization to make the lower threshold resources as null utilization resources and then the resources with null utilization are turned off.

V. EXPERIMENTS, RESULTS AND DISCUSSION

This section provides a brief overview of simulation environment, simulation scenario, results and discussions. The following flowchart elaborate the overall working of entire system.


Simultion Environment (WorkflowSim)
Performance Evaluation Metrics
Resource Modelling
Application Modelling

A. SIMULATION ENVIRONMENT

The simulation of FD-SWMS strategy is performed in WorkflowSim [41]. It is a toolkit extensively used for simulation of scientific workflows which was developed by extended

TABLE 2. The resource experimental specification.

No. VMs	Memory	BW	VM	Arch
1000 VMs	10240 MB	10000 Mbps	Xen	X86
OS	**Cost per VM $/Hr**	**Memory Cost $/s**	**Storage Cost $/S**	**Data Transfer Cost $/s**
Linux	3.0 $/Hr	0.05 $/s	0.1 $/S	0.1 $/s

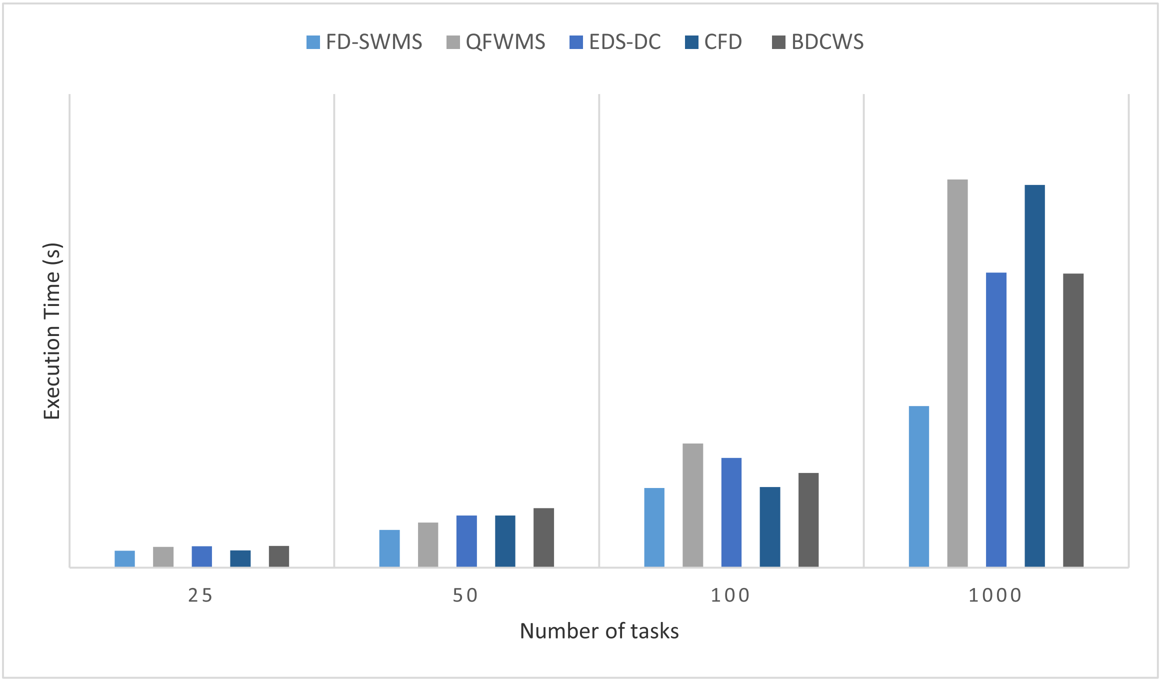


FIGURE 5. Comparison of proposed FD-SWMS strategy for montage scientific workflow in respect of execution time.

the CloudSim [59] simulator. The following performance evaluation metrics have been used.

Execution time: It is the total time required to execute the scientific workflow [60]. It is measured in seconds.

Deadline: It is predefined total execution time to execute the scientific workflow [61]. It is given in seconds.

Execution cost: It is the total budget required to execute the scientific workflow workflow [60]. It is measured in dollars.

Budget: It is the predefined cost required to execute the scientific workflows [61]. It is given in dollars.

Energy consumption: It is the power consumed, when a scientific workflow is executed [61]. It is measured in joules.

SLA Violation: It is a term used when available cost of the system is exceeded from available budget or make-span is exceeded from the deadline [61].

B. RESOURCE MODELING

In order to evaluate, FD-SWMS strategy, the WorkflowSim is the most relevant simulation environment. It is modified to support the working of FD-SWMS strategy with performance parameters. Space-shared resources are used with certain characteristics, such as budget, deadline and energy consumption. Rests of the specifications are shown in Table 2.

C. APPLICATION MODELING

In simulation scenario of FD-SWMS strategy, one user submits two real-time scientific workflows. The workflows are Montage [43] with 25, 50, 100 and 1000 tasks and Cyber-Shake [62] with 30, 50, 100 and 1000 tasks.

D. RESULTS AND DISCUSSION

This section presents the results of a proposed FD-SWMS along with the discussions on results in respect of underlying performance evaluation parameters.

1) SCENARIO 1

In scenario 1, a single user submits the real-time scientific workflows Montage with 25, 50, 100 and 1000 tasks. The objective is to evaluate the FD-SWMS strategy and comparison of results with existing start-of-art strategies.

a: EXECUTION TIME

The results, with regard to the execution time for the FD-SWMS strategy compared with the existing strategies are shown in FIGURE 5. The results reflect that the execution time is minimum for the proposed FD-SWMS strategy. It is because of that the proposed strategy finds the

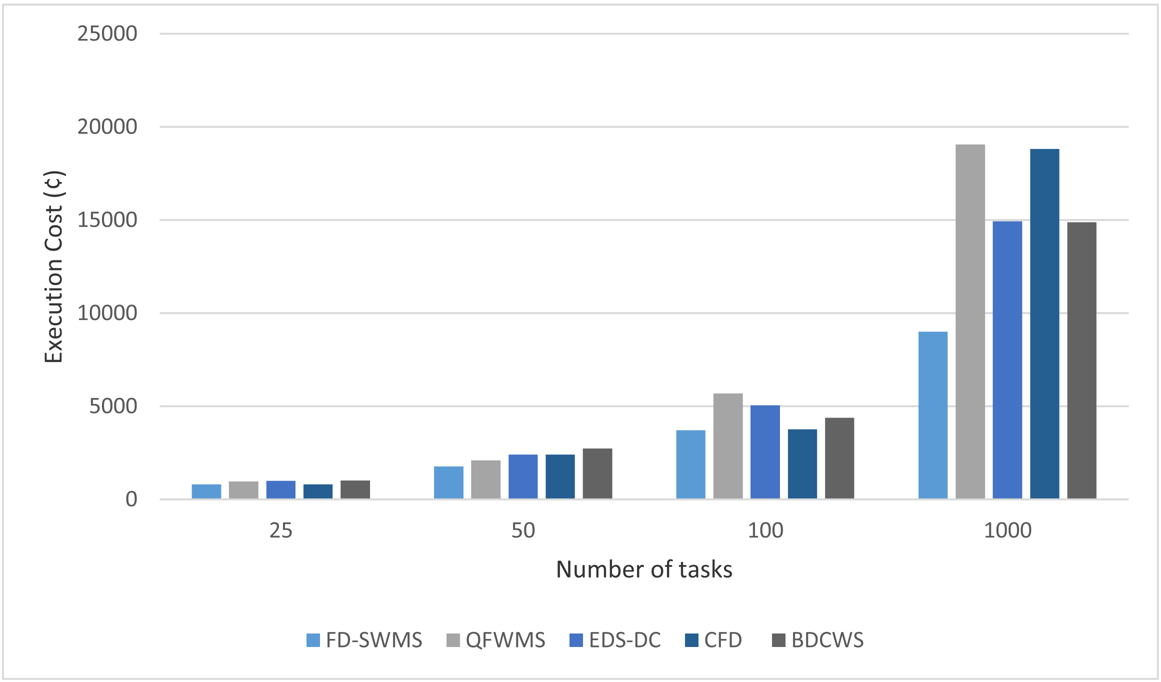


FIGURE 6. Comparison of proposed FD-SWMS strategy for montage scientific workflow in respect of execution cost.

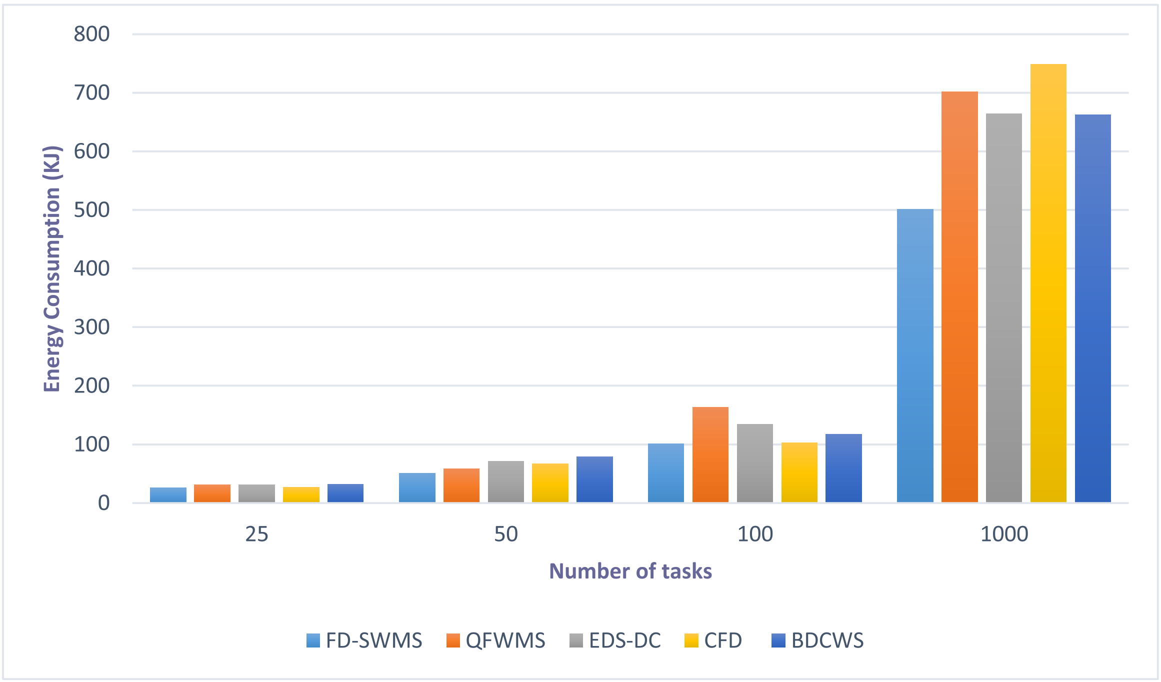


FIGURE 7. Comparison of proposed FD-SWMS strategy for montage scientific workflow in respect of energy consumption.

resource for each task with minimum data transfer time and executes. It also ensures the fault tolerance by implementing the dynamic re-clustering based mechanism.

b: EXECUTION COST

The results, with regard to the execution cost for the FD-SWMS strategy compared with the existing strategies are shown in FIGURE 6. The results reflect that the execution cost is minimum for the proposed FD-SWMS strategy. The reason is that the proposed strategy finds the resource for each task with minimum data transfer time and executes. It also ensures the fault tolerance by implementing the dynamic re-clustering based mechanism.

TABLE 3. Results in respect of deadline, budget and SLA violation for Montage Scientific workflow.

Scientific Workflow	Deadline (s)	Budget (¢)	Scheduling Policy	Execution time (s)	Execution cost (¢)	Energy Consumption (J)	SLA Violation	
							Time	Cost
Montage-25	350	1050	***FD-SWMS***	250.61	804.02	26	×	×
			QFWMS	306.93	973	31	×	×
			EDS-DC	318.57	1007.9	31	×	×
			CFD	253.64	813.11	27	×	×
			BDCWS	323	1021.21	32	×	×
Montage-50	700	2100	***FD-SWMS***	557.2	1771.61	51	×	×
			QFWMS	666.85	2100.55	59	×	✓
			EDS-DC	769.85	2409.54	71	✓	✓
			CFD	771.74	2415.23	67	✓	✓
			BDCWS	881.18	2743.53	79	✓	✓
Montage-100	1400	4200	***FD-SWMS***	1177.74	3720.41	101	×	×
			QFWMS	1834.54	5690.83	163	✓	✓
			EDS-DC	1623.05	5056.36	134	✓	✓
			CFD	1192.3	3764.09	103	×	×
			BDCWS	1397.93	4380.99	117	×	✓
Montage-1000	14000	42000	***FD-SWMS***	12389.46	39013.3	1001	×	×
			QFWMS	15736.85	49055.44	1202	✓	✓
			EDS-DC	14361.62	44929.78	1164	✓	✓
			CFD	15657.37	48817.02	1249	✓	✓
			BDCWS	14345.45	44881.25	1162	✓	✓

c: ENERGY CONSUMPTION
The results, with regard to the energy consumption for the FD-SWMS strategy compared with the existing strategies are shown in FIGURE 7. The results reflect that the energy consumption is minimum for the proposed FD-SWMS strategy. It is because of that the proposed strategy initially finds the resource for each task with minimum data transfer time and executes. It also ensures the fault tolerance by implementing the dynamic re-clustering based mechanism. Finally, it implements the load sharing mechanism by finding the resources with minimum utilization and transfer the data from such nodes to other nodes in order to reduce energy consumption.

d: SLA VIOLATION
Table 3 shows the results of the FD-SWMS strategy along with the existing strategies. In case of the proposed FD-SWMS strategy, the SLA is not violated by time or cost constraints. However, for other strategies, it is violated number of times.

2) SCENARIO 2
In scenario 2, a single user submits the real-time scientific workflows CyberShake with 30, 50, 100 and 1000 tasks. The objective is to evaluate the FD-SWMS strategy and comparison of results with existing start-of-art strategies. The execution cost, budget, make-span, deadline, energy consumption and the SLA violation are the performance evaluation parameters.

a: EXECUTION TIME
The results, with regard to the execution time for the FD-SWMS strategy compared with the existing strategies are shown in FIGURE 8. The results reflect that the execution time is minimum for the proposed FD-SWMS strategy. It is because of that the proposed strategy finds the resource for each task with minimum data transfer time and executes. It also ensures the fault tolerance by implementing the dynamic re-clustering based mechanism.

b: EXECUTION COST
The results, with regard to the execution cost for the FD-SWMS strategy compared with the existing strategies are shown in FIGURE 9. The results reflect that the execution cost is minimum for the proposed FD-SWMS strategy. The reason is that the proposed strategy finds the resource for each task with minimum data transfer time and executes. It also ensures the fault tolerance by implementing the dynamic re-clustering based mechanism.

c: ENERGY CONSUMPTION
The results, with regard to the energy consumption for the FD-SWMS strategy compared with the existing strategies are shown in FIGURE 10. The results reflect that the energy consumption is minimum for the proposed FD-SWMS strategy. It is because of that the proposed strategy initially finds the resource for each task with minimum data transfer time and executes. It also ensures the fault tolerance by implementing

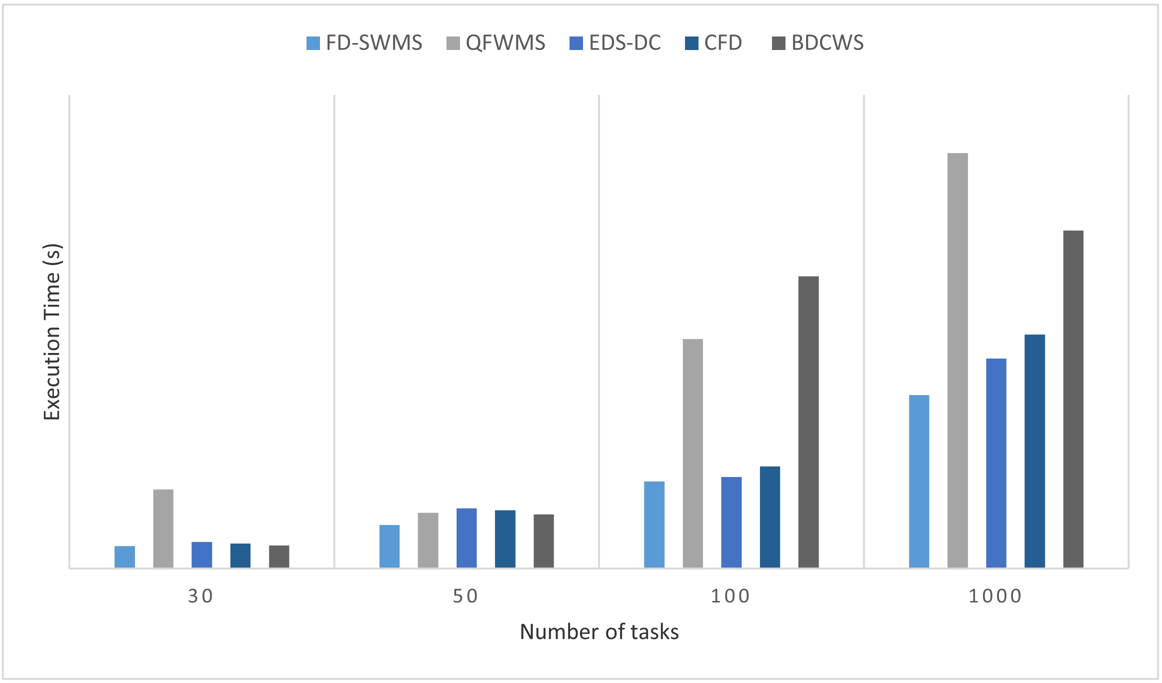


FIGURE 8. Comparison of proposed FD-SWMS strategy for CyberShake scientific workflow in respect of execution time.

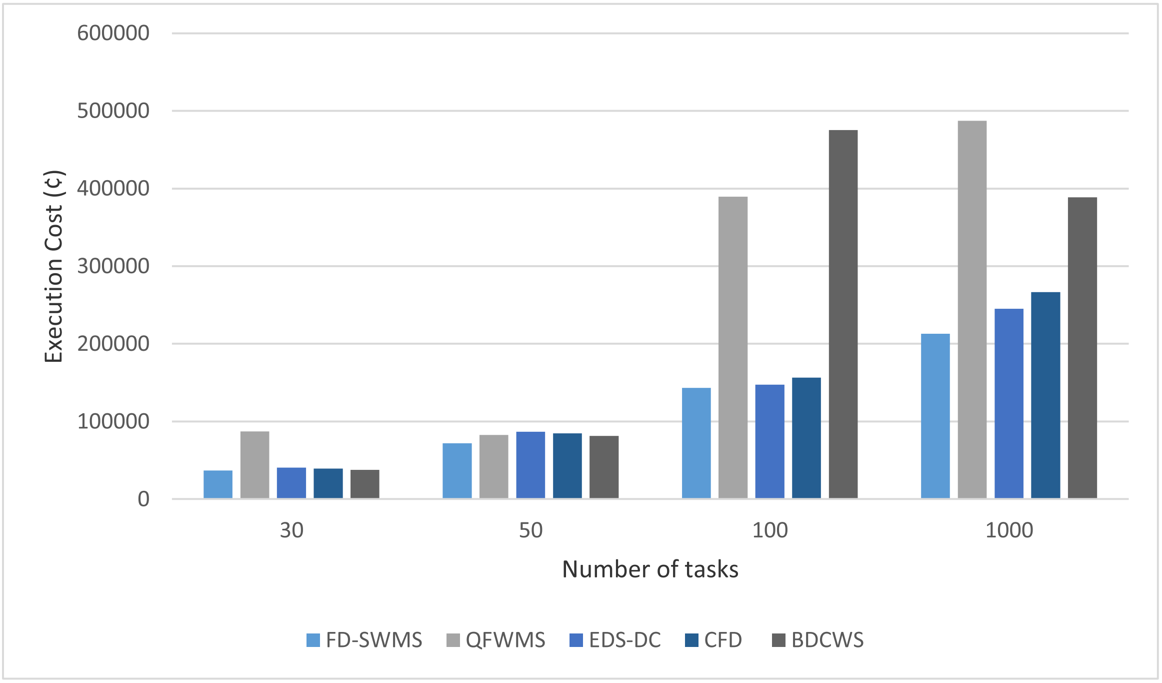


FIGURE 9. Comparison of proposed FD-SWMS strategy for CyberShake scientific workflow in respect of execution cost.

the dynamic re-clustering based mechanism. Finally, it implements the load sharing mechanism by finding the resources with minimum utilization and transfer the data from such nodes to other nodes in order to reduce energy consumption.

d: SLA VIOLATION

Table 4 shows the results of the FD-SWMS strategy along with the existing strategies. In case of the proposed FD-SWAMS strategy, the SLA is not violated by time constraints or cost constraints. While for the other strategies, it is violated several times.

The FD-SWMS strategy is an efficient approach as it includes a thorough procedure from the submission of scientific data to the generation of results. The simulations are carried out on WorkflowSim for Montage and CyberShake workflows. The proposed FD-SWMS strategy

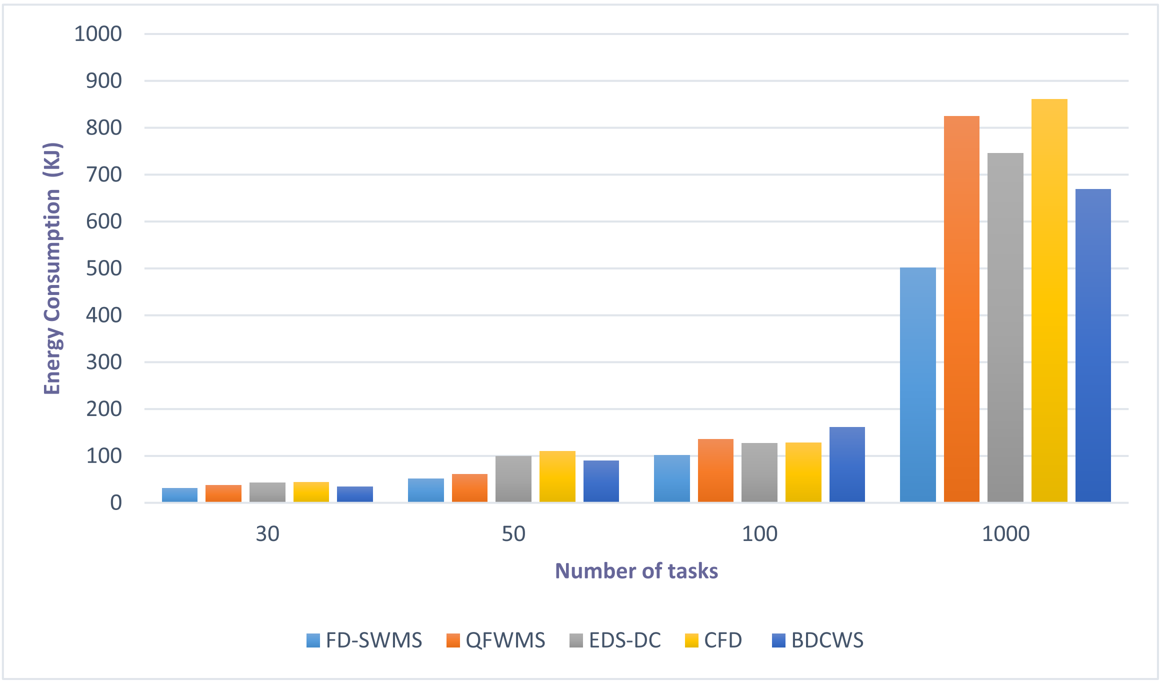


FIGURE 10. Comparison of proposed FD-SWMS strategy for CyberShake scientific workflow in respect of energy consumption.

TABLE 4. Results in respect of deadline, budget and SLA violation for CyberShake Scientific workflow.

Scientific Workflow	Deadline (s)	Budget (¢)	Scheduling Policy	Execution time (s)	Execution cost (¢)	Energy Consumption (KJ)	SLA Violation	
							Time	Cost
CyberShake -30	10000	40000	***FD-SWMS***	6612.95	36758.24	31	×	×
			QFWMS	23409.03	87146.5	38	✓	✓
			EDS-DC	7877.81	40552.82	43	×	✓
			CFD	7445.64	39256.31	44	×	×
			BDCWS	6871.12	37532.75	34	×	×
CyberShake -50	20000	80000	***FD-SWMS***	12889.19	71911.27	51	×	×
			QFWMS	16532.72	82841.87	61	×	✓
			EDS-DC	17792.88	86622.35	99	×	✓
			CFD	17208.52	84869.26	110	×	✓
			BDCWS	16048.05	81387.86	90	×	✓
CyberShake -100	40000	160000	***FD-SWMS***	25784.59	143414.98	101	×	×
			QFWMS	40866.75	209661.44	136	✓	✓
			EDS-DC	27073.72	147282.37	127	×	×
			CFD	30179.97	156601.12	128	×	×
			BDCWS	50361.69	255146.28	161	✓	✓
CyberShake -1000	100000	400000	***FD-SWMS***	51273.06	212889.69	1001	×	×
			QFWMS	77746.44	357309.83	1325	×	✓
			EDS-DC	62024.53	245144.09	1246	×	×
			CFD	69185.45	266626.86	1361	×	×
			BDCWS	74863.07	308659.7	1169	×	×

performs better as compared with the existing state-of-art strategies. The proposed strategy on average reduced execution time 25%, 17%, 22% and 16%, minimized the execution cost 24%, 17%, 21% and 16%, and decreased the energy consumption 21%, 17%, 20% and 16%, as compared with existing QFWMS, EDS-DC, CFD and BDCWS strategies respectively for Montage scientific workflow. Similarly, the proposed strategy on average reduced execution time 48%, 17%, 25% and 42%, minimized the execution cost 45%, 11%, 16% and 38%, and decreased the energy consumption 27%, 25%, 32% and 20%, as compared with existing QFWMS, EDS-DC, CFD and BDCWS strategies respectively for CyberShake scientific workflow. It is because of that the proposed strategy initially finds the resource for each task with minimum data transfer time and executes. It also ensures the fault tolerance by implementing the dynamic re-clustering based mechanism. Finally, it implements the load sharing mechanism by finding the resources with minimum utilization and transferring the data from such nodes to other nodes in order to reduce energy consumption.

The novelty of the work is that the proposed strategy applies a multi-criteria-based approach to schedule and manage the tasks of scientific workflows. The proposed strategy considers the special characteristics of tasks in scientific workflows, i.e., the scientific workflow tasks are executed simultaneously in parallel, in pipelined, aggregated to form a single task, and distributed to create multiple tasks. The proposed strategy schedules the tasks based on the data-intensiveness, provides a fault tolerant technique through a cluster-based approach, and makes it energy efficient through a load sharing mechanism. The data-intensive scheduling is achieved in such a way that the proposed strategy finds and assigns the resource to each task with the minimum data transfer time. The fault tolerance is achieved through dynamic re-clustering of scientific workflow tasks. The proposed strategy makes a cluster of tasks with similar resource requirements, executes the cluster, then finds the failed tasks from each cluster dynamically, re-clusters them, and sends them for execution. The proposed strategy achieves load management through a load sharing approach. The proposed strategy obtains the utilization of resources by making them in ascending order and then sharing the load of nodes with minimum utilization of the other nodes. The literature review shows that the existing works have not provided a multi-criteria based strategy that considers the special features of scientific workflows.

VI. CONCLUSION

This research work proposed the FD-SWMS strategy, a re-cluster based, fault-tolerant and data-intensive workflow management and scheduling strategy for scientific workflows in a cloud environment. The FD-SWMS strategy schedules the workflow tasks by the process in which it finds the best suitable resource based on data transfer time. The FD-SWMS ensures fault tolerance by implementing a dynamic re-clustering mechanism. The FD-SWMS minimizes the energy consumption by implementing a load sharing mechanism. The simulation was carried out on WorkflowSim for Montage and CyberShake workflows. The proposed FD-SWMS strategy performs better as compared with the existing state-of-the-art strategies. The proposed strategy on average reduced execution time by 25%, 17%, 22%, and 16%, minimized the execution cost by 24%, 17%, 21%, and 16%, and decreased the energy consumption by 21%, 17%, 20%, and 16%, as compared with existing QFWMS, EDS-DC, CFD, and BDCWS strategies, respectively for Montage scientific workflow. Similarly, the proposed strategy on average reduced execution time by 48%, 17%, 25%, and 42%, minimized the execution cost by 45%, 11%, 16%, and 38%, and decreased the energy consumption by 27%, 25%, 32%, and 20%, as compared with existing QFWMS, EDS-DC, CFD, and BDCWS strategies, respectively for CyberShake scientific workflow. It is because of that the proposed strategy initially finds the resource for each task with minimum data transfer time. It ensures fault tolerance by implementing a dynamic re-clustering based mechanism. It implements the load sharing mechanism by finding the resources with the minimum utilization and transferring the data from such nodes to other nodes in order to reduce energy consumption.

In the future, this work will be expanded to consider features such as resource intensive scheduling, application-aware scheduling, and domain-oriented scheduling. This research work will be extended to implement application aware resource scheduling for big data applications. The study will also be extended to explore and implement fault tolerant techniques with task oriented scheduling for business workflows.

REFERENCES

[1] M. Sookhak, F. R. Yu, M. K. Khan, Y. Xiang, and R. Buyya, "Attribute-based data access control in mobile cloud computing: Taxonomy and open issues," *Future Generat. Comput. Syst.*, vol. 72, pp. 273–287, Jul. 2017, doi: 10.1016/j.future.2016.08.018.

[2] V. Mauch, M. Kunze, and M. Hillenbrand, "High performance cloud computing," *Future Gener. Comput. Syst.*, vol. 29, no. 6, pp. 1408–1416, 2013, doi: 10.1016/j.future.2012.03.011.

[3] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, and I. Brandic, "Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility," *Future Gener. Comput. Syst.*, vol. 25, no. 6, pp. 599–616, 2009, doi: 10.1016/j.future.2008.12.001.

[4] N. Anwar and H. Deng, "Elastic scheduling of scientific workflows under deadline constraints in cloud computing environments," *Future Internet*, vol. 10, no. 1, p. 5, Jan. 2018, doi: 10.3390/fi10010005.

[5] D. Sun, G. Chang, C. Miao, and X. Wang, "Analyzing, modeling and evaluating dynamic adaptive fault tolerance strategies in cloud computing environments," *J. Supercomput.*, vol. 66, no. 1, pp. 193–228, 2013, doi: 10.1007/s11227-013-0898-7.

[6] 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," *Program. Comput. Softw.*, vol. 45, no. 8, pp. 506–516, Dec. 2019, doi: 10.1134/S0361768819080097.

[7] M. Farid, R. Latip, M. Hussin, and N. A. W. A. Hamid, "A survey on QoS requirements based on particle swarm optimization scheduling techniques for workflow scheduling in cloud computing," *Symmetry*, vol. 12, no. 4, p. 551, Apr. 2020, doi: 10.3390/SYM12040551.

[8] J. Liu, L. Pineda, E. Pacitti, A. Costan, P. Valduriez, G. Antoniu, and M. Mattoso, "Efficient scheduling of scientific workflows using hot metadata in a multisite cloud," *IEEE Trans. Knowl. Data Eng.*, vol. 31, no. 10, pp. 1940–1953, 2018, doi: 10.1109/TKDE.2018.2867857.

[9] D. Chakraborty, V. V. Mankar, and A. A. Nanavati, "Enabling runtime adaptation of workflows to external events in enterprise environments," in *Proc. IEEE Int. Conf. Web Services (ICWS)*, Jul. 2007, pp. 1112–1119, doi: 10.1109/ICWS.2007.85.

[10] M. A. Rodriguez and R. Buyya, "Budget-driven scheduling of scientific workflows in IaaS clouds with fine-grained billing periods," *ACM Trans. Auton. Adapt. Syst.*, vol. 12, no. 2, pp. 1–22, 2017, doi: 10.1145/3041036.

[11] G. L. Stavrinides and H. D. Karatza, "An energy-efficient, QoS-aware and cost-effective scheduling approach for real-time workflow applications in cloud computing systems utilizing DVFS and approximate computations," *Future Gener. Comput. Syst.*, vol. 96, pp. 216–226, Jul. 2019, doi: 10.1016/j.future.2019.02.019.

[12] S. Bharathi, E. Deelman, G. Mehta, K. Vahi, A. Chervenak, and M. Su, "Characterization of scientific workflows," in *Proc. Workshop Workflows Support Large-Scale Sci.*, 2008, pp. 1–10.

[13] M. H. Shirvani and R. N. Talouki, "Bi-objective scheduling algorithm for scientific workflows on cloud computing platform with makespan and monetary cost minimization approach," *Complex Intell. Syst.*, vol. 8, no. 2, pp. 1085–1114, Apr. 2022, doi: 10.1007/s40747-021-00528-1.

[14] J. Zhou, T. Wang, P. Cong, P. Lu, T. Wei, and M. Chen, "Cost and makespan-aware workflow scheduling in hybrid clouds," *J. Syst. Archit.*, vol. 100, Nov. 2019, Art. no. 101631, doi: 10.1016/j.sysarc.2019.08.004.

[15] E. Deelman, K. Vahi, G. Juve, M. Rynge, S. Callaghan, P. J. Maechling, R. Mayani, W. Chen, R. F. Da Silva, M. Livny, and K. Wenger, "Pegasus, a workflow management system for science automation," *Future Gener. Comput. Syst.*, vol. 46, pp. 17–35, May 2015, doi: 10.1016/j.future.2014.10.008.

[16] M. Atkinson, S. Gesing, J. Montagnat, and I. Taylor, "Scientific workflows: Past, present and future," *Future Gener. Comput. Syst.*, vol. 75, pp. 216–227, Oct. 2017, doi: 10.1016/j.future.2017.05.041.

[17] F. Pop, A. Iosup, and R. Prodan, "HPS-HDS: High performance scheduling for heterogeneous distributed systems," *Future Gener. Comput. Syst.*, vol. 78, pp. 242–244, Jan. 2018, doi: 10.1016/j.future.2017.09.012.

[18] K. Kanagaraj and S. Swamynathan, "Structure aware resource estimation for effective scheduling and execution of data intensive workflows in cloud," *Future Gener. Comput. Syst.*, vol. 79, pp. 878–891, Feb. 2018, doi: 10.1016/j.future.2017.09.001.

[19] Y. Wen, Z. Wang, Y. Zhang, J. Liu, B. Cao, and J. Chen, "Energy and cost aware scheduling with batch processing for instance-intensive IoT workflows in clouds," *Future Gener. Comput. Syst.*, vol. 101, pp. 39–50, Dec. 2019, doi: 10.1016/j.future.2019.05.046.

[20] R. Sakellariou, H. Zhao, and E. Deelman, "Mapping workflows on grid resources: Experiments with the montage workflow," in *Grids, P2P and Services Computing*. Boston, MA, USA: Springer, 2010, pp. 119–132, doi: 10.1007/978-1-4419-6794-7_10.

[21] S. Callaghan, P. Maechling, P. Small, K. Milner, G. Juve, T. H. Jordan, E. Deelman, G. Mehta, K. Vahi, D. Gunter, K. Beattie, and C. Brooks, "Metrics for heterogeneous scientific workflows: A case study of an earthquake science application," *Int. J. High Perform. Comput. Appl.*, vol. 25, no. 3, pp. 274–285, Aug. 2011, doi: 10.1177/1094342011414743.

[22] K. Vinay, S. M. D. Kumar, S. Raghavendra, and K. R. Venugopal, "Cost and fault-tolerant aware resource management for scientific workflows using hybrid instances on clouds," *Multimedia Tools Appl.*, vol. 77, no. 8, pp. 10171–10193, Apr. 2018, doi: 10.1007/s11042-017-5304-7.

[23] M. Ghose, P. Verma, S. Karmakar, and A. Sahu, "Energy efficient scheduling of scientific workflows in cloud environment," in *Proc. IEEE 19th Int. Conf. High Perform. Comput. Commun., IEEE 15th Int. Conf. Smart City, IEEE 3rd Int. Conf. Data Sci. Syst. (HPCC/SmartCity/DSS)*, Dec. 2017, pp. 170–177, doi: 10.1109/HPCC-SmartCity-DSS.2017.22.

[24] S. Elsherbiny, E. Eldaydamony, M. Alrahmawy, and A. E. Reyad, "An extended intelligent water drops algorithm for workflow scheduling in cloud computing environment," *Egyptian Informat. J.*, vol. 19, no. 1, pp. 33–55, Mar. 2018, doi: 10.1016/j.eij.2017.07.001.

[25] V. Priya, C. S. Kumar, and R. Kannan, "Resource scheduling algorithm with load balancing for cloud service provisioning," *Appl. Soft Comput.*, vol. 76, pp. 416–424, Mar. 2019, doi: 10.1016/j.asoc.2018.12.021.

[26] S. Basu, M. Karuppiah, K. Selvakumar, K.-C. Li, S. K. H. Islam, M. M. Hassan, and M. Z. A. Bhuiyan, "An intelligent/cognitive model of task scheduling for IoT applications in cloud computing environment," *Future Gener. Comput. Syst.*, vol. 88, pp. 254–261, Nov. 2018, doi: 10.1016/j.future.2018.05.056.

[27] V. Singh, I. Gupta, and P. K. Jana, "A novel cost-efficient approach for deadline-constrained workflow scheduling by dynamic provisioning of resources," *Future Gener. Comput. Syst.*, vol. 79, pp. 95–110, Feb. 2018, doi: 10.1016/J.FUTURE.2017.09.054.

[28] I. Casas, J. Taheri, R. Ranjan, L. Wang, and A. Y. Zomaya, "GA-ETI: An enhanced genetic algorithm for the scheduling of scientific workflows in cloud environments," *J. Comput. Sci.*, vol. 26, pp. 318–331, May 2018, doi: 10.1016/J.JOCS.2016.08.007.

[29] M. A. Rodriguez and R. Buyya, "A taxonomy and survey on scheduling algorithms for scientific workflows in IaaS cloud computing environments," *Concurrency Comput., Pract. Exper.*, vol. 29, no. 8, pp. 1–23, Dec. 2017, doi: 10.1002/cpe.4041.

[30] L. Zeng, B. Veeravalli, and A. Y. Zomaya, "An integrated task computation and data management scheduling strategy for workflow applications in cloud environments," *J. Netw. Comput. Appl.*, vol. 50, pp. 39–48, Apr. 2015, doi: 10.1016/j.jnca.2015.01.001.

[31] M. Safari and R. Khorsand, "Energy-aware scheduling algorithm for time-constrained workflow tasks in DVFS-enabled cloud environment," *Simul. Model. Pract. Theory*, vol. 87, pp. 311–326, Sep. 2018, doi: 10.1016/j.simpat.2018.07.006.

[32] X. Zhu, J. Wang, H. Guo, D. Zhu, L. T. Yang, and L. Liu, "Fault-tolerant scheduling for real-time scientific workflows with elastic resource provisioning in virtualized clouds," *IEEE Trans. Parallel Distrib. Syst.*, vol. 27, no. 12, pp. 3501–3517, Dec. 2016, doi: 10.1109/TPDS.2016.2543731.

[33] Y. Wen, J. Liu, W. Dou, X. Xu, B. Cao, and J. Chen, "Scheduling workflows with privacy protection constraints for big data applications on cloud," *Future Gener. Comput. Syst.*, vol. 108, pp. 1084–1091, Jul. 2020, doi: 10.1016/j.future.2018.03.028.

[34] H. R. Faragardi, M. R. Saleh Sedghpour, S. Fazliahmadi, T. Fahringer, and N. Rasouli, "GRP-HEFT: A budget-constrained resource provisioning scheme for workflow scheduling in IaaS clouds," *IEEE Trans. Parallel Distrib. Syst.*, vol. 31, no. 6, pp. 1239–1254, Jun. 2020, doi: 10.1109/TPDS.2019.2961098.

[35] G. Khojasteh Toussi and M. Naghibzadeh, "A divide and conquer approach to deadline constrained cost-optimization workflow scheduling for the cloud," *Cluster Comput.*, vol. 24, no. 3, pp. 1711–1733, Sep. 2021, doi: 10.1007/s10586-020-03223-x.

[36] M. A. Rodriguez and R. Buyya, "Budget-driven scheduling of scientific workflows in IaaS clouds with fine-grained billing periods," *ACM Trans. Auton. Adapt. Syst.*, vol. 12, no. 2, pp. 1–22, Jun. 2017, doi: 10.1145/3041036.

[37] K. Qureshi, F. G. Khan, P. Manuel, and B. Nazir, "A hybrid fault tolerance technique in grid computing system," *J. Supercomput.*, vol. 56, no. 1, pp. 106–128, Apr. 2011, doi: 10.1007/s11227-009-0345-y.

[38] A. Bala and I. Chana, "Fault tolerance-challenges, techniques and implementation in cloud computing," *Int. J. Comput. Sci.*, vol. 9, no. 1, pp. 288–293, 2012.

[39] W. Chen and E. Deelman, "Fault tolerant clustering in scientific workflows," in *Proc. IEEE 8th World Congr. Services*, Jun. 2012, pp. 9–16, doi: 10.1109/SERVICES.2012.5.

[40] W. Chen, R. F. da Silva, E. Deelman, and T. Fahringer, "Dynamic and fault-tolerant clustering for scientific workflows," *IEEE Trans. Cloud Comput.*, vol. 4, no. 1, pp. 49–62, Jan. 2016, doi: 10.1109/TCC.2015.2427200.

[41] W. Chen and E. Deelman, "WorkflowSim: A toolkit for simulating scientific workflows in distributed environments," in *Proc. IEEE 8th Int. Conf. E-Sci.*, Oct. 2012, pp. 1–8, doi: 10.1109/eScience.2012.6404430.

[42] W. Tang, J. Jenkins, F. Meyer, R. Ross, R. Kettimuthu, L. Winkler, X. Yang, T. Lehman, and N. Desai, "Data-aware resource scheduling for multicloud workflows: A fine-grained simulation approach," in *Proc. IEEE 6th Int. Conf. Cloud Comput. Technol. Sci.*, Dec. 2014, pp. 887–892, doi: 10.1109/CloudCom.2014.19.

[43] E. Deelman, G. Singh, M. Livny, B. Berriman, and J. Good, "The cost of doing science on the cloud: The montage example," in *Proc. Int. Conf. High Perform. Comput., Netw., Storage Anal.*, Nov. 2008, pp. 1–12, doi: 10.1109/SC.2008.5217932.

[44] 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," *Int. J. Commun. Syst.*, vol. 34, no. 1, pp. 1–23, 2021, doi: 10.1002/dac.4649.

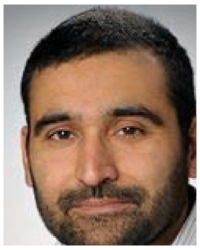
[45] Z. Ahmad, A. I. Jehangiri, M. A. Ala'anzy, M. Othman, and A. I. Umar, "Fault-tolerant and data-intensive resource scheduling and management for scientific applications in cloud computing," *Sensors*, vol. 21, no. 21, p. 7238, Oct. 2021, doi: 10.3390/s21217238.

[46] N. Zhou, W. Lin, W. Feng, F. Shi, and X. Pang, "Budget-deadline constrained approach for scientific workflows scheduling in a cloud environment," *Cluster Comput.*, vol. 1, pp. 1–15, Sep. 2020, doi: 10.1007/s10586-020-03176-1.

[47] V. Arabnejad, K. Bubendorfer, and B. Ng, "Scheduling deadline constrained scientific workflows on dynamically provisioned cloud resources," *Future Gener. Comput. Syst.*, vol. 75, pp. 348–364, Oct. 2017, doi: 10.1016/j.future.2017.01.002.

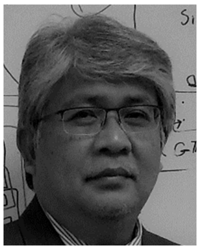
[48] M. Ghasemzadeh, H. Arabnejad, and J. G. Barbosa, "Deadline-budget constrained scheduling algorithm for scientific workflows in a cloud environment," in *Proc. Leibniz Int. Informat.*, 2017, vol. 70, no. 19, pp. 19.1–19.16, doi: 10.4230/LIPIcs.OPODIS.2016.19.
[49] F. Juarez, J. Ejarque, and R. M. Badia, "Dynamic energy-aware scheduling for parallel task-based application in cloud computing," *Future Gener. Comput. Syst.*, vol. 78, pp. 257–271, Jan. 2018, doi: 10.1016/j.future.2016.06.029.
[50] A. M. Madureira and A. B. Definitions, "Ordered minimum completion time heuristic for unrelated parallel-machines problems," in *Proc. 9th Iberian Conf. Inf. Syst. Technol.*, 2014, pp. 1–6.
[51] R. JeminaPriyadarsini and L. Arockiam, "Performance evaluation of min-min and max-min algorithms for job scheduling in federated cloud," *Int. J. Comput. Appl.*, vol. 99, no. 18, pp. 47–54, Aug. 2014.
[52] M. Amoon, "A framework for providing a hybrid fault tolerance in cloud computing," in *Proc. Sci. Inf. Conf. (SAI)*, Jul. 2015, pp. 844–849, doi: 10.1109/SAI.2015.7237242.
[53] K. Ganga and S. Karthik, "A fault tolerent approach in scientific workflow systems based on cloud computing," in *Proc. Int. Conf. Pattern Recognit., Informat. Mobile Eng.*, Feb. 2013, pp. 387–390, doi: 10.1109/ICPRIME.2013.6496507.
[54] 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 Gener. Comput. Syst.*, vol. 74, pp. 168–178, Sep. 2017, doi: 10.1016/j.future.2015.12.005.
[55] J. Liu, E. Pacitti, P. Valduriez, and M. Mattoso, "A survey of data-intensive scientific workflow management," *J. Grid Comput.*, vol. 13, no. 4, pp. 457–493, 2015, doi: 10.1007/s10723-015-9329-8.
[56] C. Acevedo, P. Hernández, A. Espinosa, and V. Mendez, "A data-aware MultiWorkflow scheduler for clusters on WorkflowSim," in *Proc. 2nd Int. Conf. Complex., Future Inf. Syst. Risk*, 2017, pp. 79–86, doi: 10.5220/0006303500790086.
[57] X. Zhou, G. Zhang, J. Sun, J. Zhou, T. Wei, and S. Hu, "Minimizing cost and makespan for workflow scheduling in cloud using fuzzy dominance sort based HEFT," *Future Gener. Comput. Syst.*, vol. 93, pp. 278–289, Apr. 2019, doi: 10.1016/j.future.2018.10.046.
[58] W. Chen, R. Ferreira, E. Deelman, and R. Sakellariou, "Balanced task clustering in scientific workflows," in *Proc. 9th Int. Conf. e-Sci.*, 2013, pp. 1–8.
[59] R. Calheiros, R. Ranjan, A. Beloglazov, C. A. F. De Rose, and R. Buyya, "CloudSim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," *Softw., Pract. Exper.*, vol. 41, no. 1, pp. 23–50, 2011, doi: 10.1002/spe.995.
[60] D. Poola, K. Ramamohanarao, and R. Buyya, "Fault-tolerant workflow scheduling using spot instances on clouds," *Proc. Comput. Sci.*, vol. 29, pp. 523–533, Jan. 2014, doi: 10.1016/j.procs.2014.05.047.
[61] T. Mathew, K. C. Sekaran, and J. Jose, "Study and analysis of various task scheduling algorithms in the cloud computing environment," in *Proc. Int. Conf. Adv. Comput., Commun. Informat. (ICACCI)*, Sep. 2014, pp. 658–664, doi: 10.1109/ICACCI.2014.6968517.
[62] S. Callaghan, E. Deelman, D. Gunter, G. Juve, P. Maechling, C. Brooks, K. Vahi, K. Milner, R. Graves, E. Field, D. Okaya, and T. Jordan, "Scaling up workflow-based applications," *J. Comput. Syst. Sci.*, vol. 76, no. 6, pp. 428–446, Sep. 2010, doi: 10.1016/j.jcss.2009.11.005.

ZULFIQAR AHMAD received the M.Sc. degree (Hons.) in computer science from Hazara University, Mansehra, Khyber Pakhtunkhwa, Pakistan, in 2012, and the M.S. degree in CS from COMSATS University Islamabad, Abbottabad, Khyber Pakhtunkhwa, in 2016. He is currently pursuing the Ph.D. degree in computer science with the Department of Computer Science and Information Technology, Hazara University. His research interests include fog computing, cloud computing, high performance computing, and scientific workflows execution and management.



ALI IMRAN JEHANGIRI graduated from Bergische Universität Wuppertal, Germany, in 2010. He received the Ph.D. degree in computer science from Georg-August-Universität Göttingen, Germany, in 2015. He is currently working as an Assistant Professor with the Department of Computer Science and Information Technology, Hazara Universtiy, Mansehra, Pakistan. He gained industrial experience with service computing working as a Research Assistant at GWDG. He is the author of several publications in international journals and conferences. His research interests include parallel, grid computing, cloud computing, and big data.

NADER MOHAMED (Member, IEEE) received the Ph.D. degree in computer science from the University of Nebraska–Lincoln, Lincoln, NE, USA. He was a Faculty Member with the Stevens Institute of Technology, Hoboken, NJ, USA; and United Arab Emirates University, Al Ain, United Arab Emirates. He is currently a Professor of computer science and information systems with Pennsylvania Western University, California, PA, USA. He also has several years of industrial experience in information technology. His current research interests include cybersecurity, middleware, industry 4.0, cloud and fog computing, networking, healthcare systems, and cyber-physical systems.



MOHAMED OTHMAN (Senior Member, IEEE) received the Ph.D. degree (Hons.) from the National University of Malaysia. He is currently a Professor in computer science with the Department of Communication Technology and Networks, Universiti Putra Malaysia (UPM). Prior to that, he was the Deputy Director of the Information Development and Communication Centre, where he was in charge of the UMPNet Network Campus, uSport Wireless Communication Project, and the UPM Data Center. He was also a Visiting Professor with South Kazakhstan State University, Shymkent, Kazakhstan; and the L. N. Gumilyov Eurasian National University, Astana, Kazakhstan. He is also an Associate Researcher and a Coordinator of high-speed machines with the Laboratory of Computational Science and Informatics, Institute of Mathematical Science, UPM. He has published more than 300 international journals and 330 proceeding papers. He has also filed six Malaysian, one Japanese, one South Korean, and three U.S. patents. His main research interests include computer networks, parallel and distributed computing, high speed interconnection networks, network design and management (network security, wireless, and traffic monitoring), consensus in the IoT, and mathematical models in scientific computing. He is a Life Member of the Malaysian National Computer Confederation and the Malaysian Mathematical Society. He was awarded the Best Ph.D. Thesis, in 2000, by Sime Darby Malaysia and the Malaysian Mathematical Science Society. In 2017, he has received an Honorary Professorship from SILKWAY International University (formerly known as South Kazakhstan Pedagogical University), Shymkent.

ARIF IQBAL UMAR received the M.Sc. degree in computer science from the University of Peshawar, Pakistan, and the Ph.D. degree in computer science from Beihang University (BUAA), Beijing, China. He is currently working as an Associate Professor of computer science with the Department of Computer Science and Information Technology, Hazara University, Mansehra. He is leading the Department as the Chairperson. He has supervised seven Ph.D. candidates and 34 M.S. candidates. He is the author of more than 70 research publications in the leading research journals and conferences. He has at his credit 27 years' experience of teaching, research, planning, and academic management. His research interests include data mining, machine learning, information retrieval, digital image processing, computer networks security, and sensor networks.