*applied sciences*

*Article*

# Replication-Based Dynamic Energy-Aware Resource Provisioning for Scientific Workflows

**Mohammed Alaa Ala'anzy [1,*], Mohamed Othman [1,2], Emad Mohammed Ibbini [3], Odai Enaizan [4], Mazen Farid [1], Yousef A. Alsaaidah [1], Zulfiqar Ahmad [5,*] and Rania M. Ghoniem [6]**

1 Department of Communication Technology and Networks, Universiti Putra Malaysia (UPM), Serdang 43400, Malaysia
2 Laboratory of Computational Science and Mathematical Physics, Institute for Mathematical Research (INSPEM), Universiti Putra Malaysia (UPM), Serdang 43400, Malaysia
3 Department of Computer Science, Al Balqa Applied University, Al-Salt 19117, Jordan
4 Department of Management Information System, College of Haql, University of Tabuk, Tabuk 71491, Saudi Arabia
5 Department of Computer Science and Information Technology, Hazara University, Mansehra 21300, Pakistan
6 Department of Information Technology, College of Computer and Information Sciences, Princess Nourah bint Abdulrahman University, Riyadh 11671, Saudi Arabia
* Correspondence: m.alanzy.cs@gmail.com (M.A.A.); zulfiqarahmad@hu.edu.pk (Z.A.)

**Abstract:** Distributed computing services in cloud environments are easily accessible to end users. These services are delivered to end users via a subscription-based model. The "infrastructure as a service" (IaaS) cloud model is one of the best cloud environment models for running data- and computing-intensive applications. Real-world scientific applications are the best examples of data and computing intensiveness. For their implementation, scientific workflow applications need high-performance computational resources and a large volume of storage. The workflow tasks are linked based on computational and data interdependence. Considering the high volume and variety of scientific workflows (SWs), the resources of the IaaS cloud model require managing energy efficiently and without failure or loss. Therefore, in order to address the issues of power consumption and task failure for real-world SWs, this research work proposes a replication-based dynamic energy-aware resource provisioning (R-DEAR) strategy for SWs in an IaaS cloud environment. The proposed strategy, R-DEAR, is a resource- and service-provisioning strategy that implements a replication-based fault-tolerant and load-balancing mechanism. The proposed R-DEAR strategy schedules the tasks of a scientific workflow with a replication-based fault-tolerant mechanism. The proposed R-DEAR strategy also manages the power consumption of IaaS cloud resources dynamically through a load-sharing process. Simulation results show that the proposed R-DEAR strategy reduces energy consumption, execution cost, and execution time by 9%, 15%, and 18%, respectively, as compared with the existing state-of-the-art strategy.

**Keywords:** cloud computing; scientific workflow; Montage; CyberShake; replication; resource provisioning

**Citation:** Ala'anzy, M.A.; Othman, M.; Ibbini, E.M.; Enaizan, O.; Farid, M.; Alsaaidah, Y.A.; Ahmad, Z.; Ghoniem, R.M. Replication-Based Dynamic Energy-Aware Resource Provisioning for Scientific Workflows. *Appl. Sci.* **2023**, *13*, 2644. https://doi.org/10.3390/app13042644

Academic Editors: Ce Li and Bei Guan

Received: 31 January 2023
Revised: 10 February 2023
Accepted: 15 February 2023
Published: 18 February 2023

## 1. Introduction

The cloud environment provides end users with distributed computing services [1]. The computing services are provided to the end users in a subscription-based atmosphere [2]. The cloud environment provides computing services in the shape of applications, platforms, and computing resources, termed "software as a service" (SaaS) [3], "platform as a service" (PaaS) [4] and "infrastructure as a service" (IaaS) [5], respectively. IaaS is a flexible cloud environment model in which distributed computing resources such as storage, memory, bandwidth, and processing power are provided to clients on a pay-per-use basis [6,7].

The three main subcategories of cloud computing services are SaaS, PaaS, and IaaS. SaaS is a method of distributing computer programs in which the vendor hosts the program and makes it accessible to users online. Customers who use SaaS do not have to worry about maintaining software or managing the underlying infrastructure. Instead, users merely use a web browser or a thin client to access the software. Systems for managing client relationships (CRM), corporate resource planning (ERP), and email services are examples of SaaS [3]. PaaS offers a platform for creating, deploying, and managing services and applications. Customers who use PaaS do not have to worry about constructing and maintaining the platform or managing the underlying infrastructure. They can instead concentrate on creating and implementing their applications. Google App Engine, Microsoft Azure, and Heroku are some examples of PaaS [4]. Over the internet, IaaS offers virtualized computer resources. Clients can rent computing resources including virtual machines, storage, and networking through IaaS. They are in charge of managing the software and applications that are used by the infrastructure. IaaS examples include "Google Cloud Platform", "Microsoft Azure", and "Amazon Web Services" (AWS) [5].

The IaaS cloud service model is currently the most suitable cloud environment model for the execution of highly data- and computing-intensive applications [8]. The best examples of this in the context of data- and computing-intensiveness are real-world scientific workflows (SWs). Real-world SWs are those such as Montage in astronomy, CyberShake in earthquake science, and Epigenomics in biology; they are scientific applications developed by or for scientists for analyses in their scientific research [9–11]. For their implementation, SW applications need high-performance computing resources and large volumes of storage, which can be obtained from the IaaS cloud service model. Scientists in most scientific fields, such as astronomy, bioinformatics and high-energy physics, conduct research that requires and creates terabytes of data obtained from physical equipment. Their workflow activities are connected based on computational and data interdependence [12,13]. SWs are highly complex applications and are represented in terms of directed acyclic graphs (DAGs) [14]. Tasks are represented by nodes in a DAG, and computational or data dependencies among nodes are represented by the edges [15–17].

The resource-provisioning process for the execution of SWs with minimum energy consumption and without failure of tasks is a challenging process due to the variety of workflow tasks and complex dependencies among them [10,18,19]. Since workflow tasks need massive amounts of computation and storage with diverse resource requirements, the best solution is to deploy the workflow tasks on an IaaS cloud service [20–22]. However, keeping in mind the high volume and variety of SWs, the resources of an IaaS cloud environment require managing energy efficiently and without failure [23–25]. Therefore, in order to address the issues of energy consumption and task failure for real-world SWs, this research work proposed a replication-based dynamic energy-aware resource-provisioning (R-DEAR) strategy for SWs in an IaaS cloud environment.

The main contributions of the proposed work are as follows:

1. This research work presents a dynamic energy-aware resource-provisioning (DEAR) strategy for SWs in an IaaS cloud service model.
2. This work also implements a replication-based fault-tolerant mechanism in the proposed DEAR strategy to make it R-DEAR, a failure-free resource-provisioning technique for the IaaS cloud service model.
3. In the proposed R-DEAR strategy, one or more users will submit SWs to the workflow manager.
4. The workflow/resource information provider (WRIP) will obtain the details of resources and scientific workflow requirements. It is presumed that the energy consumption for each resource is predefined for the submitted workflow on the basis of tasks contained within it.
5. The workflow scheduler obtains information regarding the workflow tasks from the workflow manager. The tasks are sorted per their energy consumption in ascending order T1 < T2 < T3 . . . Tn.

6. The WRIP provides information about resources and workflows to the workflow scheduler. The accessible resources are arranged in descending order by energy consumption: R1 > R2 > R3 . . . Rn.
7. The workflow scheduler sends the information about tasks and resources to the workflow engine.
8. The workflow engine assigns tasks to resources in each process based on sorted lists, and then starts the workflow. The workflow engine distributes the workload across available resources and reports task and resource status to the workflow replica manager.
9. During task execution, the workflow replica manager keeps a copy of each task and checks the resource status. If a task fails, a duplicate of the task will be sent to the resource to finish the execution.
10. The workflow engine will compile the result after successful execution and return it to the end user.

The remaining parts of the paper are presented as follows: work related to the proposed research is given in Section 2; the system design and model are provided in Section 3; the details of the evaluation methods are highlighted in Section 4; the experimental setup, results, and discussion are presented in Section 5; and the conclusion, limitations and future work are given in Section 6.

## 2. Related Work

The related work is explored and analyzed regarding resource management in cloud computing, SW scheduling, and fault-tolerant techniques for SWs in cloud computing.

As part of their cyber-infrastructure, many workflow applications and workflow management systems (WMSs) are developed to allow scientists to run their applications in a variety of distributed environments. Despite the fact that scientists have used both practical and theoretical techniques, detection, failure prediction, and recovery remain research issues. In [26], a strategy is presented for detecting and mitigating issues before they arise, based on control theory and developed as part of autonomic computing. The suggested technique is based on the "proportional-integral-derivative controller" (PID controller). It is widely employed in industrial control systems, and it entails the controller altering its output in response to difficulties. To show the practicality of the proposed technique, the authors tackled two common execution challenges of large-scale data-intensive processes. The authors constructed a simulator that implements and analyses basic standalone PID-inspired controllers in a data-oriented bioinformatics process to manage data and memory use autonomously. To execute all processes simultaneously, the simulator generated over 4.4 TB of data and required over 24 TB of RAM. According to simulation data, the controller-inspired method looks to greatly improve workflow performance, particularly in online and unpredictable contexts.

A multi-objective scheduling technique for cloud computing scientific processes is presented in [27]. The method is based on a genetic algorithm that tries to strike a balance between make-span, financial cost, and load. The recommended method firstly finds the best answer for each parameter. Based on these answers, the algorithm chooses the best solution for all parameters. The proposed method was evaluated on benchmark datasets. The results demonstrate that the recommended strategy enhances the make-span and decreases the cost when used with a well-balanced system.

Scientific operations are complicated, and they necessitate the efficient use of cloud resources. SW scheduling is considered an NP-complete problem. Some criteria, such as quality of service (QoS), interdependence between jobs, and user deadlines, comprise challenges. There is a substantial body of knowledge about scheduling SWs in cloud systems. Standard schedulers, evolutionary optimization approaches and other options are available. A hybrid approach for scheduling scientific activities in cloud systems is presented in [28]. The algorithm creates task lists for the particle swarm optimization (PSO) algorithm in the first phase. To reduce execution time, bottleneck tasks are given top priority. The PSO algorithm is used to schedule tasks in the next phase, which reduces both execution time and cost.

In [17], an approach for scheduling SWs called the dynamic scheduling of bag-of-tasks-based workflows (DSB) is presented; itaims to reduce the financial cost of hired virtual machines (VMs) while meeting user-defined time restrictions. The suggested technique divides the workflow into bag-of-tasks (BoT) applications based on data dependencies and priority restrictions, and then enhances the assignment and scheduling of BoTs. Using benchmark scientific procedures that resemble real-world applications, a trace-based simulation indicated a considerable decrease in workflow computing costs while still meeting deadlines. When compared to modified state-of-the-art methodologies, the findings show that the proposed model produces higher success rates in terms of meeting deadlines and cost efficiencies.

As discussed in [29], the scheduling of SWs on a hybrid cloud architecture is modelled as a bi-objective optimization problem with a make-span and monetary cost minimization goal. This research suggested a "hybrid bi-objective optimization based on simulated annealing and task duplication algorithm" (BOSA-TDA) would improve conventional SA by utilizing two fundamental heuristics: duplication and "heterogeneous earliest finish time" (HEFT) techniques. Simulation was performed and the results reflect that the proposed BOSA-TDA outperforms the existing approaches.

Security is a significant issue in the cloud; therefore, the authors in [30] included security limitations in an optimization model. Their main goal is to establish an optimum plan in the shortest time and with the lowest cost, while also satisfying the necessary security limitations. During the negotiation process, consumers can submit their security demands to the cloud provider. The workflow is first scheduled using list-based heuristics, and then particle swarm optimization is used to optimize. As a result, the authors devised a secured scheduling algorithm based on "smart particle swarm optimization" (SPSO) to determine the optimal schedule with the shortest time and lowest cost. The suggested method is capable of allocating tasks in SWs to the most appropriate virtual cloud machine. Thus, the suggested strategy addresses resource allocation. In addition, a variation of the PSO algorithm known as variable neighborhood PSO is being tested to solve the local optima problem.

By providing on-demand and low-cost computer resources to customers, cloud computing is now a promising paradigm. As a result, scientific workflows (SWs), which are big-data applications, are becoming increasingly reliant on cloud computing resources. However, in such a huge distributed computing system, internal failure (or host fault) is unavoidable. It has also been established that cloud data centers would be subjected to hostile attacks on a regular basis. As a result, when conducting SWs in the cloud, external failure (failure due to a malicious assault) should be taken into account. In [31], a "fault-tolerant scheduling" (FTS) solution for SW in a cloud computing environment is presented to minimize workflow cost while achieving deadline constraints, even when internal and external defects are present. The FTS method, which is based on the jobs replication strategy, is one of the most widely used fault-tolerant algorithms. The outcomes of the experiments have proven to be beneficial in real-world SW applications.

The authors in [32] describe a "dynamic fault-tolerant workflow scheduling" (DFTWS) approach that includes hybrid spatial and temporal re-execution strategies. Initially, DFTWS finds task time attributes and detects the workflow critical path. Then, the resource allocation phase is started, in which a suitable virtual machine (VM) is assigned to each job based on the task deadline and budget. Finally, during execution, DFTWS runs online scheduling, where real-time fault-tolerant decisions are taken depending on failure type and job criticality. Real-world workflows are used to test the suggested method. In addition, the elements that influence DFTWS performance are examined. DFTWS acquires a trade-off between high dependability and minimum cost objectives in cloud computing.

In [9], the authors present a quality-of-service aware "fault-tolerant workflow management system" (QFWMS) for scientific processes in cloud computing. For the purpose of evaluating the proposed QFWMS, the authors used two real-time SWs: Montage and CyberShake. The proposed QFWMS scheduling was tested using the WorkflowSim [33] simulation environment. A proactive intelligent fault-tolerant model is required to complete a workflow without interruption. The research in [34] offers a cognitive fault-tolerant (CFT) model with three key phases for proactively tolerating task and VM failure. Combined ensemble prediction methods are used to anticipate task failures in the prediction phase, and label-tuning techniques are employed to generate intermediate labels and reinforce the prediction. The work is isolated during the segregation step based on its priority. Recovery is the final phase in the CFT paradigm. Fitness checking is used to determine whether the expected failure is caused by the task or by the virtual machine. When a job fails, the post-prediction check-pointing (PPC) mechanism is utilized to recover. The post- or pre-replication overlapped migration approach can be used to recover VM failure. Experiments show that the suggested CFT model improves process execution reliability in a cloud context.

The authors in [35] presented a replication-based partial critical path (R-PCP) workflow scheduling strategy for upcoming data transmission and completing process activities while staying within deadlines and financial restrictions. The first data placement step in the proposed method clusters and distributes datasets depending on their relationships. R-PCP schedules activities based on data proximity and dynamically maintains a dependency matrix for locating created datasets. To reduce runtime–dataset migration, the authors use inter-datacenter task replication and dataset replication to assure dataset availability. R-PCP beats previous approaches by successfully reducing data transmission and performing all chosen operations within the user-specified budget and timescale, as evidenced by simulation results with four workflow applications [35]. Various energy saving protocols, such as a budget-constrained energy consumption optimization approach and particle swarm optimization, have been highlighted in [36–38]. The proposed R-DEAR strategy saves energy through optimization of resource utilization as compared with the existing energy-saving protocols.

An overview of the related work has been given in Table 1.

**Table 1.** An overview of related work.

| References | Strategy | Contribution | Evaluation Platform | Limitations |
|---|---|---|---|---|
| [9] | QFWMS | Provides scheduling solution based on data awareness | WorkflowSim | Not an energy efficient approach |
| [17] | DSB | Bag-of-tasks-based solution to reduce financial cost | WorkflowSim | Not an energy efficient and fault tolerant approach |
| [26] | PID controller | Prevents and mitigates data storage overload and memory overflow | Activity-based simulator | Not an energy efficient approach |
| [27] | GA-PSO | Provides a solution based on genetic algorithm to minimize time and cost | CloudSim | No consideration of fault tolerance |
| [35] | R-PCP | Provides a replication-based scheduling solution in multi-cloud environment | WorkflowSim | Not an energy efficient approach |

## 3. System Design and Model

This research work proposes an R-DEAR strategy for SWs in IaaS cloud environments. Figure 1 shows the workings of the proposed R-DEAR strategy. One or more users submit SWs to the workflow manager. The workflow/resource information provider (WRIP) registers the details of resources and SW requirements. It is assumed that for each computing resource, the energy consumption is predefined on the basis of tasks contained in the workflow. The workflow scheduler acquires information about the tasks of each workflow from the workflow manager. The tasks are arranged in ascending order as T1 < t2 < t3< span = “”> . . . Tn, according to energy consumption. The WRIP provides information about resources and workflows to the workflow scheduler. The available resources are listed in descending order by energy usage R1 > R2 > R3 . . . Rn. The tasks and resource information are sent to the workflow engine by the workflow scheduler. The workflow engine starts the execution of each workflow by assigning tasks to resources based on sorted lists. The workflow engine distributes workload among available resources and communicates task and resource status to the workflow replica manager. During task execution, the workflow replica manager keeps a copy of each task and checks the resource status. If a task fails, a duplicate of the task will be sent to the resource to finish the execution. The workflow engine will compile the result after successful execution and return it to the end user.

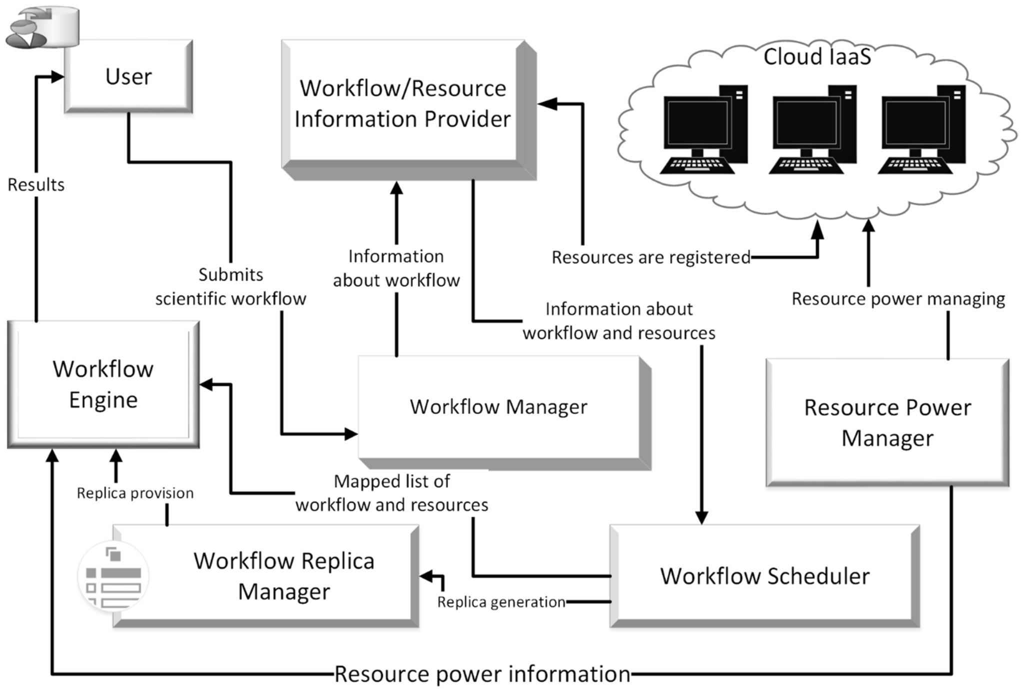


**Figure 1.** R-DEAR Strategy.

### 3.1. Parts of R-DEAR Strategy

The R-DEAR strategy has the following parts:

1. User
2. Workflow manager
3. Workflow/resource information provider
4. Workflow scheduler
5. Workflow engine
6. Workflow replica manager

#### 3.1.1. User

The user is an entity that submits an SW for execution and obtains the required results. The users in the proposed strategy are assumed to be scientists in various fields, including medical science, gravitational physics and earth science. The scientists perform experiments by using the computational and storage resources of cloud infrastructures.

#### 3.1.2. Workflow Manager

The workflow manager collects SWs from users. It identifies the validity and pre-requisites of the SWs and transforms them into tasks as directed acyclic graphs. It then submits the details of tasks to the workflow/resource information provider. The workflow manager also preserves the energy consumption and computational cost information about tasks.

#### 3.1.3. Workflow/Resource Information Provider (WRIP)

The WRIP lists the computational and storage resources from the IaaS cloud platform. It also obtains workflow information from the workflow manager with resource requirements. The information also comprises the costs of computation and power consumption of the accessible resources.

#### 3.1.4. Workflow Scheduler

The workflow scheduler obtains information regarding the workflow tasks and resource information from the workflow/resource information provider. Then, the tasks of a workflow are arranged in ascending order according to energy consumption as T1 < T2 < T3 . . . Tn, by the resource scheduler. The workflow task with minimum energy consumption is placed as T1, the one with more energy consumption is placed as T2, and so on. Similarly, it also arranges the available resources per their power consumption in the descending form R1 > R2 > R3 . . . Rn for each task. The workflow scheduler maps the list of available resources to each workflow task, and then finds the resource with minimum energy consumption from a sorted list and assigns it to a workflow task. Then, the workflow scheduler sends the workflow tasks and information about the resources to the workflow engine.

#### 3.1.5. Workflow Engine

The workflow engine obtains a mapped list of workflow tasks and resources from the workflow scheduler and runs them. It also sends the current status of tasks and computing resources to the resource power manager. When execution is successfully completed, the workflow engine generates and sends the results to the user.

#### 3.1.6. Resource Power Manager

The resource power manager component is responsible for examining the current status of resources during the execution of tasks. After inspecting the status, it checks if any resource is being underutilized and transfers the load of resources with minimum utilization to other available resources and powers off the idle resource. If any of the tasks fail during execution, it obtains a replica of the task from the workflow replica manager and re-executes the task.

#### 3.1.7. Workflow Replica Manager

The workflow replica manager maintains a replica of each task of a workflow and provides it to the workflow engine whenever the execution of any task has failed.

### *3.2. Algorithm for the R-DEAR Strategy*

The proposed R-DEAR strategy, in terms of pseudo code, is elaborated in Algorithm 1. The algorithm takes input as SWs. Each workflow is stored in a queue referred to as $W_Q$. The workflow then transforms them into tasks for execution and stores them in a queue referred to as $T_Q$. Resource information is obtained from the WRIP and stored in a queue referred to as $R_Q$.

The power utilization of each task is obtained, and then all the tasks of a workflow are sorted in ascending order per their power utilization. At this stage, the replica of each node is also generated. Resources with minimum power consumption are found for each task and assigned to the tasks. The tasks are executed through the workflow engine. If a task's execution fails during execution, a replica of the task is provided, and it is re-executed. However, if the node on which the task is executed has failed, the replicas of tasks executed on that node are provided to an alternative node for execution. The resources are monitored during execution, and if power utilization of any resource is null, it is powered off. Finally, the results are obtained and returned to the user.

**Algorithm 1** R-DEAR strategy

```
Input: δ (Scientific workflow)
Output: γ (Generated results)
Procedure: R-DEAR (δ)
1.   $W_Q$ ← $Get_{Workflows}$ ( )
2.   $T_Q$ ← $Generate_{Tasks}$ ( ) for each Workflow
3.   $R_Q$ ← $Register_{Resources}$ ( ) from WRIP
4.   for (each task $T_1$ to $T_n$) do
5.       $Get_{PowerUtilization}$ ( ) ← $T_i$
6.   end for
7.   Sort tasks task $T_1$ to $T_n$ in ascending order as per PowerUtilization ( )
8.   Update $T_1 < T_2 < T_3 \ldots, < T_n$
9.   for (each task $T_1$ to $T_n$ ) do
10.      $Get_{Resource}$ ( ) ← $R_i$ with minimum $Power_{Consumption}$ ( )
11.      $Generate_{Replica}$ ( ) ← $T_{Replica\ (i)}$
12.      Assign $T_i$ ← $R_i$
13.  end for
14.  Start Execution: $Workflow_{Engine}$ ( )
15.  while ($Workflow_{Engine}$ ( )) do
16.  if (task execution failed)
17.  Provide the task replica and re-execute
18.  else if (node failed)
19.  Provide the replica of task to alternative node and re-execute
20.  end if else
21.  end while
22.  for (all available resources) do
23.      Rp ← $Get_{Power}$ ( )
24.      if ($R_P$ == Null ) then
25.          $Resource_{PowerOff}$( )
26.      end if
27.  end for
28.  $Generate_{Result}$( )
```

The proposed R-DEAR strategy is a resource- and service-provisioning strategy that incorporates a replication-based fault-tolerant and load-balancing mechanism. The proposed R-DEAR strategy is a way to schedule the tasks of a scientific workflow using a replication-based fault-tolerant mechanism. The proposed R-DEAR strategy manages the power consumption of IaaS cloud resources dynamically through a load-sharing process.

## 4. Evaluation Methods

Comprehensive details on the simulation environment and application modeling are provided in this section.

### *4.1. Simulation Tool*

In order to establish the cloud computing environment, WorkflowSim [33], "a toolkit for simulating SWs," was amended to support the replication-based fault-tolerant mechanism and dynamic energy-aware resource scheduling. Shared-space resources were used. The rest of the specifications are shown in Table 2.

**Table 2.** Specification of resources.

| **No. VMs** | **Memory** | **BW** | **VM** | **Arch** |
|---|---|---|---|---|
| **1000 VMs** | 10,240 MB | 10,000 Mbps | Xen | X86 |
| **OS** | **Cost per VM $/Hr** | **Memory cost $/s** | **Storage cost $/S** | **Data transfer cost $/s** |
| **Linux** | 3.00 **$/Hr** | 0.05 **$/s** | 0.10 **$/S** | 0.10 **$/s** |

### *4.2. Application Modelling*

The proposed strategy was simulated for real time SWs with three dataset sizes (i.e., small, medium and large) using two well-known SWs: Montage [39] and CyberShake [40]. The sizes of small, medium, and large workflows have been distinguished based on the number of jobs contained within them. Small workflows have 50 jobs, medium contain 100 jobs and large contain 1000 jobs. The results were compared with the state-of-the-art R-PCP strategy [35].

### *4.3. Performance Evaluation Parameters*

The proposed strategy is a replication-based, dynamic energy-aware resource-provisioning strategy. In the context of the proposed strategy, the energy consumption, execution time and execution cost are significant parameters and were evaluated through the simulation environment.

## **5. Experiment Setup**

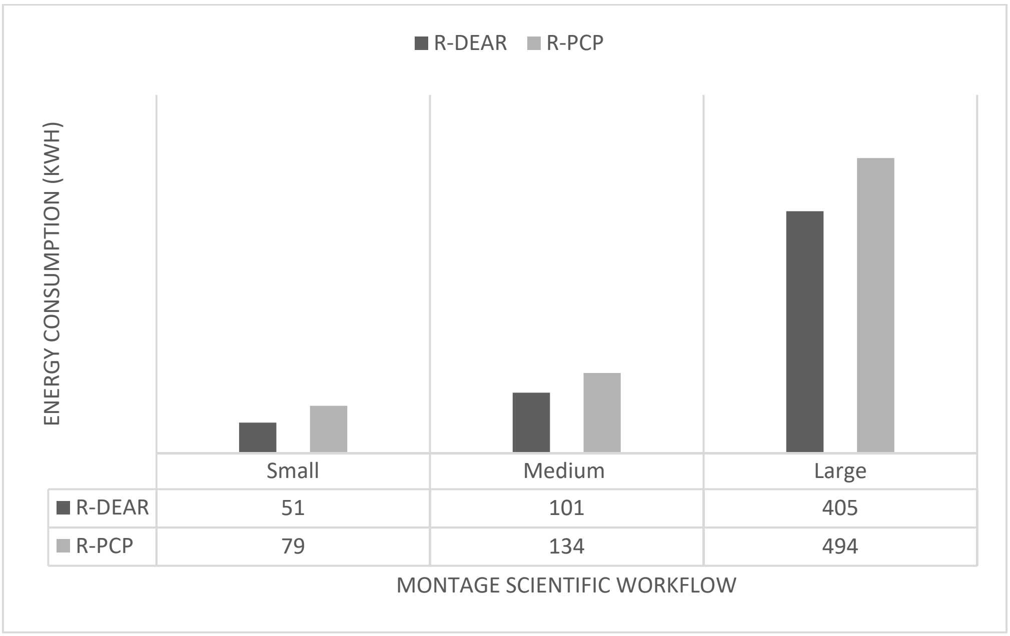
The WorkflowSim [33] workflow simulator was modified and used and to support the replication-based fault-tolerant mechanism and dynamic energy-aware resource scheduling.

### *5.1. Results and Discussion*

The proposed R-DEAR strategy was simulated with three datasets of two well-known SWs. The average values of results were considered and compared with the state-of-the-art R-PCP strategy.

#### 5.1.1. Energy Consumption

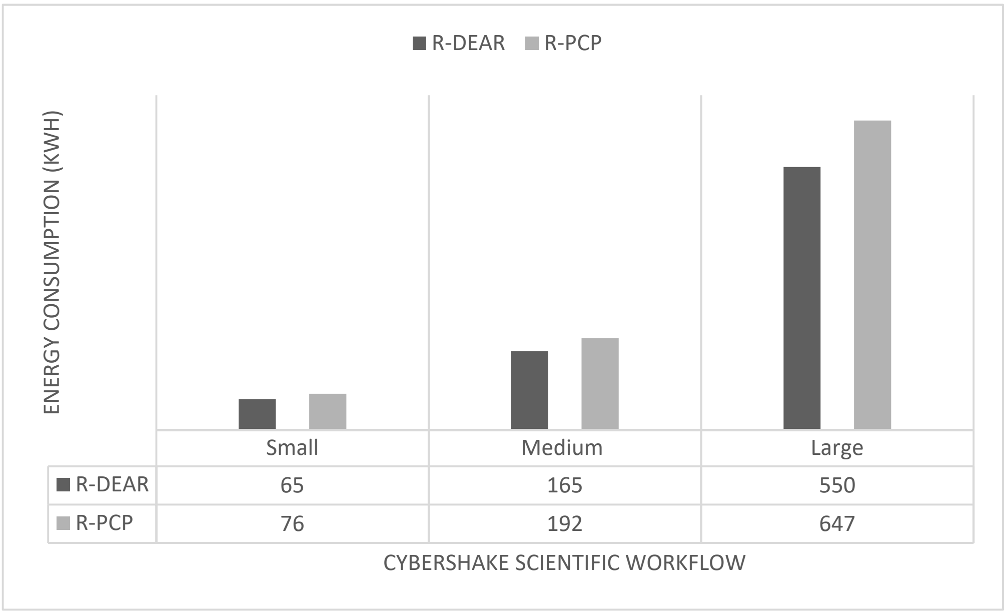
The energy consumed by the proposed R-DEAR strategy for the Montage SW is shown in Figure 2. The R-DEAR strategy's energy consumption was minimal as compared with the existing R-PCP strategy. The reason for this was the provision of proper power management in the proposed R-DEAR strategy through its load-sharing approach.



| | Small | Medium | Large |
|---|---|---|---|
| ■ R-DEAR | 51 | 101 | 405 |
| ■ R-PCP | 79 | 134 | 494 |

**Figure 2.** Energy consumption for Montage scientific workflow.

The energy consumed by the proposed R-DEAR strategy for the CyberShake SW is shown in Figure 3. The R-DEAR strategy consumed a minimum amount of energy as compared with the existing R-PCP strategy. The reason for this was the provision of proper power management in the proposed R-DEAR strategy through its load-sharing approach.



| | Small | Medium | Large |
|---|---|---|---|
| ■ R-DEAR | 65 | 165 | 550 |
| ■ R-PCP | 76 | 192 | 647 |

**Figure 3.** Energy consumption for CyberShake scientific workflow.

### 5.1.2. Execution Cost

The cost of execution for the proposed R-DEAR strategy for the Montage SW is shown in Figure 4. The proposed R-DEAR strategy reduced the execution cost as compared with the existing R-PCP strategy. The reason for this was the provision of the replica manager for dealing with the failure of tasks and power management in the R-DEAR strategy through its load-sharing approach.

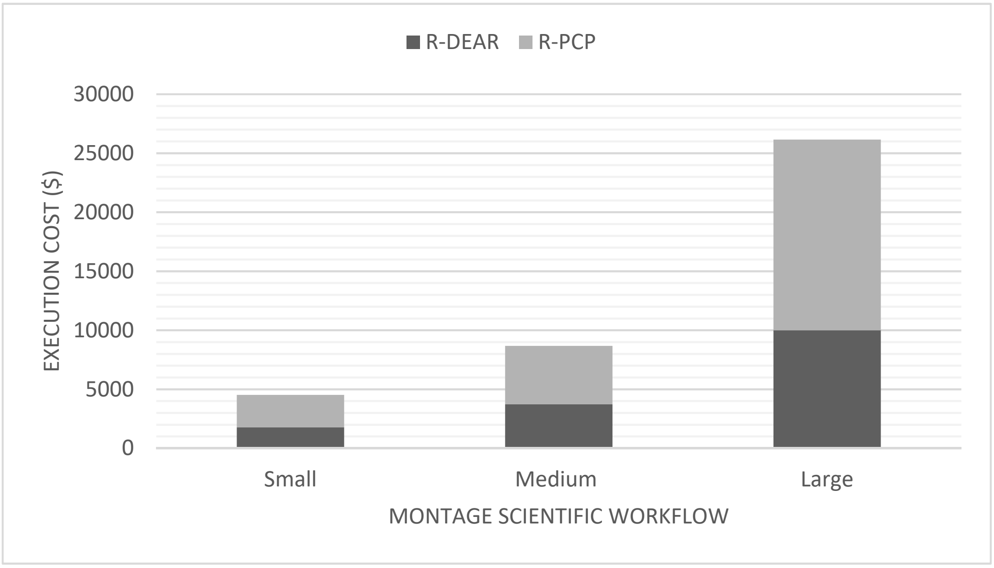


**Figure 4.** Execution cost for Montage scientific workflow.

The cost of execution for the proposed R-DEAR strategy for the CyberShake SW is shown in Figure 5. The proposed R-DEAR strategy reduced the cost as compared with the existing R-PCP strategy. The reason for this was the provision of the replica manager for dealing with the failure of tasks and power management in the proposed R-DEAR strategy through its load-sharing approach.

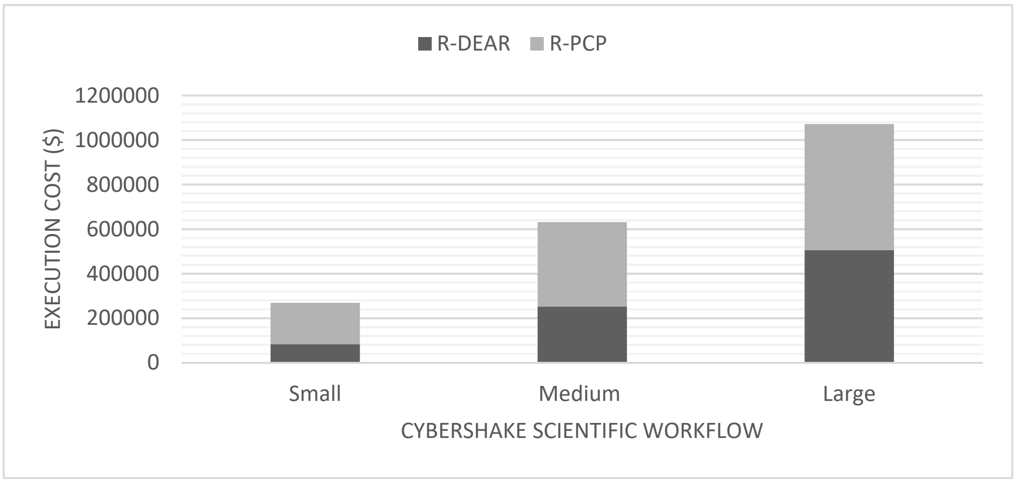


**Figure 5.** Execution cost for CyberShake scientific workflow.

The differences between the execution costs of R-DEAR and R-PCP are normal for small and medium workflows due to a lower failure rate in terms of the number of tasks. However, the differences between the execution costs of R-DEAR and R-PCP get larger for large workflows. Large workflows contain a comparatively large number of tasks, each with a higher probability of failure. The proposed R-DEAR strategy in such a situation addresses it through a replication-based load-sharing approach.

### 5.1.3. Execution Time

The execution time for the proposed R-DEAR strategy for the Montage SW is shown in Figure 6. The proposed R-DEAR strategy decreased the execution time as compared with the existing R-PCP strategy. The reason for this was the provision of the replica manager for dealing with the failure of tasks and power management in the proposed R-DEAR strategy through its load-sharing approach.

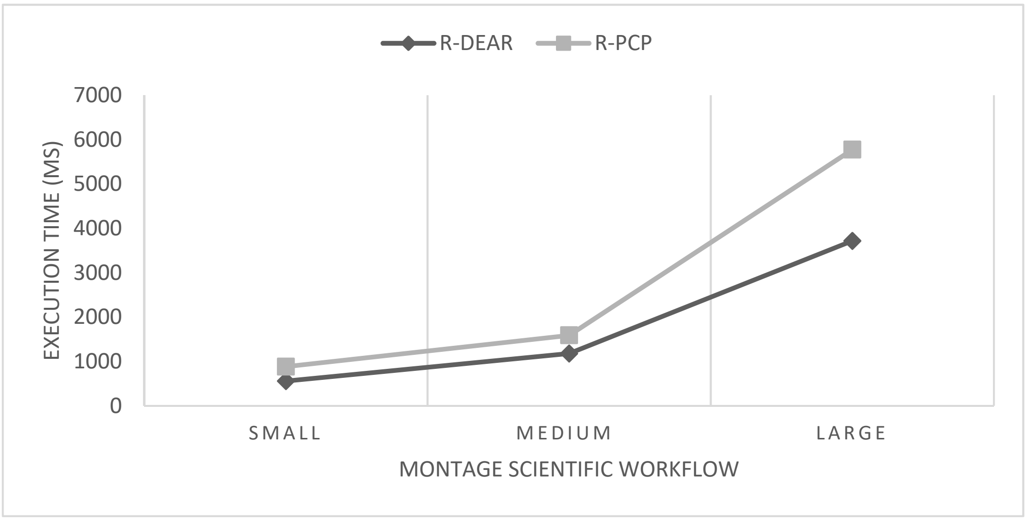


**Figure 6.** Execution time for Montage scientific workflow.

The execution time for the proposed R-DEAR strategy for the CyberShake SW is shown in Figure 7. The proposed R-DEAR strategy decreased the execution time as compared with the existing R-PCP strategy. The reason for this was the provision of the replica manager for dealing with the failure of tasks and power management in the proposed R-DEAR strategy through its load-sharing approach.

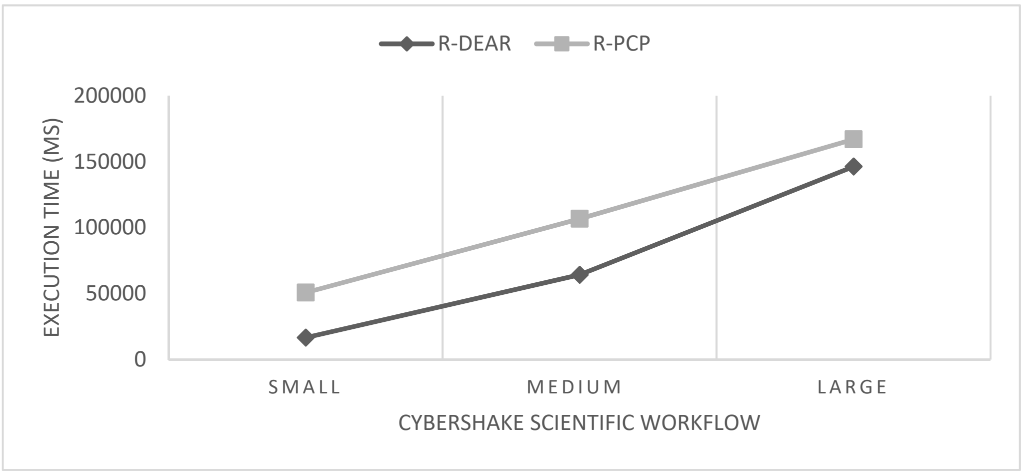


**Figure 7.** Execution time for CyberShake scientific workflow.

Scientific workflows have some special characteristics, such as tasks that are integrated, disintegrated, and executed sometimes in parallel and sometimes in sequence. Workflows such as Montage and CyberShake differ from each other based on these special characteristics. Per the structure of CyberShake, it has a higher number of parallel task executions with minimum inclusion of other workflow characteristics, which explains why there is a strong correlation between R-DEAR and R-PCP through small, medium, and large workflows.

The results conclusively show that on average, the proposed R-DEAR strategy reduces energy consumption, execution cost and execution time by 9%, 15% and 18%, respectively, as compared with the existing state-of-the-art strategy. The reason is that the proposed R-DEAR strategy assigns resources by finding the best resource for each task. The R-DEAR strategy manages the power by finding and powering off idle resources. The R-DEAR strategy also ensures the successful execution of tasks by providing a replication-based fault-tolerant mechanism.

## 6. Conclusions

This research work presents an R-DEAR strategy for SWs in an IaaS cloud environment. The motive for presenting the R-DEAR strategy is to address the issues of energy consumption and task failure for real SWs. Real-world SWs, including Montage and CyberShake, are examples of data- and computing-intensiveness. For their implementation, SW applications need high-performance computational resources and a huge amount of storage. Scientists in most scientific fields, such as astronomy, high-energy physics, and bioinformatics, conduct research that requires and produces terabytes of data obtained from physical devices. The workflow tasks are linked based on computational and data interdependence. Considering the high volume and variety of SWs, the resources of the IaaS cloud model require efficient energy management without failure or loss.

The proposed strategy, R-DEAR, is a resource-provisioning strategy that works with replication-based fault-tolerant mechanisms. The proposed R-DEAR strategy also manages the power consumption of IaaS cloud resources dynamically through a load-sharing process. Simulation was performed on a WorkflowSim simulator. The results reveal that the proposed R-DEAR strategy minimizes energy consumption, execution cost, and execution time by 9%, 15%, and 18%, respectively, against the existing state-of-the-art strategy.

The proposed study is limited to scientific workflows. Therefore, for business workflows, we need to develop another strategy based on the nature of business workflows. The proposed study implements a cloud platform for evaluation; however, a fog environment can be used in some aspects. In the future, this work will be strengthened to implement workflow scheduling and fault tolerance in fog environments. Similarly, business workflow-based implementation of a similar study will also be achieved.

**Author Contributions:** M.A.A., M.O., E.M.I., O.E. and Z.A. conceived and designed the experiments; M.F., Y.A.A., Z.A. and R.M.G. wrote the paper; M.A.A., Z.A. and R.M.G. performed the simulations and analyzed the results; R.M.G. and M.O. technically reviewed the paper. All authors have read and agreed to the published version of the manuscript.

**Funding:** Princess Nourah bint Abdulrahman University Researchers Supporting Project number (PNURSP2023R138), Princess Nourah bint Abdulrahman University, Riyadh, Saudi Arabia.

**Data Availability Statement:** No data were used to support this study. However, any query about the research conducted in this paper is highly appreciated and can be answered by the corresponding authors upon request.

**Acknowledgments:** The authors appreciate the support from Princess Nourah bint Abdulrahman University Researchers Supporting Project number (PNURSP2023R138), Princess Nourah bint Abdulrahman University, Riyadh, Saudi Arabia.

**Conflicts of Interest:** The authors declare no conflict of interest.

## References

1. Sookhak, M.; Yu, F.R.; Khan, M.K.; Xiang, Y.; Buyya, R. Attribute-based data access control in mobile cloud computing: Taxonomy and open issues. *Futur. Gener. Comput. Syst.* **2017**, *72*, 273–287. [CrossRef]
2. Ullah, A.; Li, J.; Shen, Y.; Hussain, A. A control theoretical view of cloud elasticity: Taxonomy, survey and challenges. *Clust. Comput.* **2018**, *21*, 1735–1764. [CrossRef]
3. Mustafa, S.; Nazir, B.; Hayat, A.; Khan, A.U.R.; Madani, S.A. Resource management in cloud computing: Taxonomy, prospects, and challenges. *Comput. Electr. Eng.* **2015**, *47*, 186–203. [CrossRef]
4. Masdari, M.; ValiKardan, S.; Shahi, Z.; Azar, S.I. Towards workflow scheduling in cloud computing: A comprehensive analysis. *J. Netw. Comput. Appl.* **2016**, *66*, 64–82. [CrossRef]
5. Serrano, D.; Bouchenak, S.; Kouki, Y.; de Oliveira, F.A., Jr.; Ledoux, T.; Lejeune, J.; Sopena, J.; Arantes, L.; Sens, P. SLA guarantees for cloud services. *Futur. Gener. Comput. Syst.* **2016**, *54*, 233–246. [CrossRef]
6. Dimitri, N. Pricing cloud IaaS computing services. *J. Cloud Comput.* **2020**, *9*, 14. [CrossRef]
7. Alanzy, M.; Latip, R.; Muhammed, A. Range wise busy checking 2-way imbalanced algorithm for cloudlet allocation in cloud environment. *J. Physics: Conf. Ser.* **2018**, *1018*, 012018. [CrossRef]
8. Rodriguez, M.A.; Buyya, R. Budget-Driven Scheduling of Scientific Workflows in IaaS Clouds with Fine-Grained Billing Periods. *ACM Trans. Auton. Adapt. Syst.* **2017**, *12*, 1–22. [CrossRef]
9. Ahmad, Z.; Nazir, B.; Umer, A. A fault-tolerant workflow management system with Quality-of-Service-aware scheduling for scientific workflows in cloud computing. *Int. J. Commun. Syst.* **2021**, *34*, e4649. [CrossRef]
10. Casas, I.; Taheri, J.; Ranjan, R.; Wang, L.; Zomaya, A.Y. A balanced scheduler with data reuse and replication for scientific workflows in cloud computing systems. *Futur. Gener. Comput. Syst.* **2016**, *74*, 168–178. [CrossRef]
11. da Silva, R.F.; Casanova, H.; Orgerie, A.-C.; Tanaka, R.; Deelman, E.; Suter, F. Characterizing, Modeling, and Accurately Simulating Power and Energy Consumption of I/O-intensive Scientific Workflows. *J. Comput. Sci.* **2020**, *44*, 101157. [CrossRef]
12. Choi, J.; Adufu, T.; Kim, Y. Data-Locality Aware Scientific Workflow Scheduling Methods in HPC Cloud Environments. *Int. J. Parallel Program.* **2016**, *45*, 1128–1141. [CrossRef]
13. Ala'Anzy, M.A.; Othman, M.; Hasan, S.; Ghaleb, S.M.; Latip, R. Optimising Cloud Servers Utilisation Based on Locust-Inspired Algorithm. In Proceedings of the 7th International Conference on Soft Computing & Machine Intelligence, Stockholm, Sweden, 14–15 November 2020; pp. 23–27. [CrossRef]
14. Ahmad, Z.; Jehangiri, A.I.; Ala'Anzy, M.A.; Othman, M.; Umar, A.I. Fault-Tolerant and Data-Intensive Resource Scheduling and Management for Scientific Applications in Cloud Computing. *Sensors* **2021**, *21*, 7238. [CrossRef] [PubMed]
15. Acevedo, C.; Hernández, P.; Espinosa, A.; Mendez, V. A Data-aware MultiWorkflow Scheduler for Clusters on WorkflowSim. In Proceedings of the COMPLEXIS 2017: 2nd International Conference on Complexity, Future Information Systems and Risk, Online Streaming, 23–24 April 2022; pp. 79–86. [CrossRef]
16. Gottin, V.M.; Pacheco, E.; Dias, J.; Ciarlini, A.E.M.; Costa, B.; Vieira, W.; Souto, Y.M.; Pires, P.; Porto, F.; Rittmeyer, J.G. Automatic Caching Decision for Scientific Dataflow Execution in Apache Spark. In Proceedings of the 5th ACM SIGMOD Workshop on Algorithms and Systems for MapReduce and Beyond, Houston, TX, USA, 15 June 2018. [CrossRef]
17. Anwar, N.; Deng, H. Elastic Scheduling of Scientific Workflows under Deadline Constraints in Cloud Computing Environments. *Futur. Internet* **2018**, *10*, 5. [CrossRef]
18. Rehman, A.U.; Ahmad, Z.; Jehangiri, A.I.; Ala'Anzy, M.A.; Othman, M.; Umar, A.I.; Ahmad, J. Dynamic Energy Efficient Resource Allocation Strategy for Load Balancing in Fog Environment. *IEEE Access* **2020**, *8*, 199829–199839. [CrossRef]
19. Stavrinides, G.L.; Karatza, H.D. An energy-efficient, QoS-aware and cost-effective scheduling approach for real-time workflow applications in cloud computing systems utilizing DVFS and approximate computations. *Futur. Gener. Comput. Syst.* **2019**, *96*, 216–226. [CrossRef]

20. Rodriguez, M.A.; Buyya, R. A taxonomy and survey on scheduling algorithms for scientific workflows in IaaS cloud computing environments. *Concurr. Comput. Pract. Exp.* **2016**, *29*, e4041. [CrossRef]
21. Marozzo, F.; Talia, D.; Trunfio, P. A Workflow Management System for Scalable Data Mining on Clouds. *IEEE Trans. Serv. Comput.* **2016**, *11*, 480–492. [CrossRef]
22. Ahmad, Z.; Jehangiri, A.I.; Iftikhar, M.; Umer, A.I.; Afzal, I. Data-Oriented Scheduling with Dynamic-Clustering Fault-Tolerant Technique for Scientific Workflows in Clouds. *Program. Comput. Softw.* **2019**, *45*, 506–516. [CrossRef]
23. Verma, R.K.; Pattanaik, K.; Bharti, S.; Saxena, D. In-network context inference in IoT sensory environment for efficient network resource utilization. *J. Netw. Comput. Appl.* **2019**, *130*, 89–103. [CrossRef]
24. Chen, W.; da Silva, R.F.; Deelman, E.; Fahringer, T. Dynamic and Fault-Tolerant Clustering for Scientific Workflows. *IEEE Trans. Cloud Comput.* **2015**, *4*, 49–62. [CrossRef]
25. Zhu, X.; Wang, J.; Guo, H.; Zhu, D.; Yang, L.T.; Liu, L. Fault-Tolerant Scheduling for Real-Time Scientific Workflows with Elastic Resource Provisioning in Virtualized Clouds. *IEEE Trans. Parallel Distrib. Syst.* **2016**, *27*, 3501–3517. [CrossRef]
26. da Silva, R.F.; Filgueira, R.; Deelman, E.; Pairo-Castineira, E.; Overton, I.M.; Atkinson, M.P. Using simple PID-inspired controllers for online resilient resource management of distributed scientific workflows. *Futur. Gener. Comput. Syst.* **2019**, *95*, 615–628. [CrossRef]
27. Sardaraz, M.; Tahir, M. A parallel multi-objective genetic algorithm for scheduling scientific workflows in cloud computing. *Int. J. Distrib. Sens. Networks* **2020**, *16*, 1550147720949142. [CrossRef]
28. Sardaraz, M.; Tahir, M. A Hybrid Algorithm for Scheduling Scientific Workflows in Cloud Computing. *IEEE Access* **2019**, *7*, 186137–186146. [CrossRef]
29. Shirvani, M.H.; Talouki, R.N. Bi-objective scheduling algorithm for scientific workflows on cloud computing platform with makespan and monetary cost minimization approach. *Complex Intell. Syst.* **2021**, *8*, 1085–1114. [CrossRef]
30. Sujana, J.A.J.; Revathi, T.; Priya, T.S.S.; Muneeswaran, K. Smart PSO-based secured scheduling approaches for scientific workflows in cloud computing. *Soft Comput.* **2017**, *23*, 1745–1765. [CrossRef]
31. Li, Z.; Yu, J.; Hu, H.; Chen, J.; Hu, H.; Ge, J.; Chang, V. Fault-Tolerant Scheduling for Scientific Workflow with Task Replication Method in Cloud. In Proceedings of the 3rd International Conference on Internet of Things, Big Data and Security, IoTBDS 2018, Funchal, Portugal, 19–21 March 2018; pp. 95–104. [CrossRef]
32. Wu, N.; Zuo, D.; Zhang, Z. Dynamic Fault-Tolerant Workflow Scheduling with Hybrid Spatial-Temporal Re-Execution in Clouds. *Information* **2019**, *10*, 169. [CrossRef]
33. Chen, W.; Deelman, E. WorkflowSim: A toolkit for simulating scientific workflows in distributed environments. In Proceedings of the 2012 IEEE 8th International Conference on E-Science, Chicago, IL, USA, 8–12 October 2012. [CrossRef]
34. Padmakumari, P.; Umamakeswari, A. Development of cognitive fault tolerant model for scientific workflows by integrating overlapped migration and check-pointing approach. *J. Ambient. Intell. Humaniz. Comput.* **2019**, 1–11. [CrossRef]
35. Ulabedin, Z.; Nazir, B. Replication and data management-based workflow scheduling algorithm for multi-cloud data centre platform. *J. Supercomput.* **2021**, *77*, 10743–10772. [CrossRef]
36. Zhang, L.; Wang, L.; Xiao, M.; Wen, Z.; Peng, C. EM_WOA: A budget-constrained energy consumption optimization approach for workflow scheduling in clouds. *Peer-to-Peer Netw. Appl.* **2022**, *15*, 973–987. [CrossRef]
37. Choudhary, R.; Perinpanayagam, S. Applications of Virtual Machine Using Multi-Objective Optimization Scheduling Algorithm for Improving CPU Utilization and Energy Efficiency in Cloud Computing. *Energies* **2022**, *15*, 9164. [CrossRef]
38. Bharany, S.; Sharma, S.; Khalaf, O.I.; Abdulsahib, G.M.; Al Humaimeedy, A.S.; Aldhyani, T.H.H.; Maashi, M.; Alkahtani, H. A Systematic Survey on Energy-Efficient Techniques in Sustainable Cloud Computing. *Sustainability* **2022**, *14*, 6256. [CrossRef]
39. Deelman, E.; Singh, G.; Livny, M.; Berriman, B.; Good, J. The cost of doing science on the cloud: The Montage example. In Proceedings of the SC '08: Proceedings of the 2008 ACM/IEEE Conference on Supercomputing, Austin, TX, USA, 15–21 November 2008. [CrossRef]
40. Callaghan, S.; Maechling, P.J.; Small, P.; Milner, K.; Juve, G.; Jordan, T.H.; Deelman, E.; Mehta, G.; Vahi, K.; Gunter, D.; et al. Metrics for heterogeneous scientific workflows: A case study of an earthquake science application. *Int. J. High Perform. Comput. Appl.* **2011**, *25*, 274–285. [CrossRef]