electronics

MDPI

Article

Multi-Dependency and Time Based Resource Scheduling Algorithm for Scientific Applications in Cloud Computing

Vijay Prakash [1,*], Seema Bawa [1] and Lalit Garg [2,3]

[1] Department of Computer Science & Engineering, Thapar Institute of Engineering & Technology, Patiala 147001, India; seema@thapar.edu
[2] Computer Information Systems, Faculty of Information & Communication Technology, University of Malta, MSD 2080 Msida, Malta; lalit.garg@um.edu.mt or lalit.garg@online.liverpool.ac.uk
[3] Computer Science Department, University of Liverpool, Liverpool 14 3PE, UK
* Correspondence: vijay.prakash@thapar.edu; Tel.: +356-917988402900

Abstract: Workflow scheduling is one of the significant issues for scientific applications among virtual machine migration, database management, security, performance, fault tolerance, server consolidation, etc. In this paper, existing time-based scheduling algorithms, such as first come first serve (FCFS), min–min, max–min, and minimum completion time (MCT), along with dependency-based scheduling algorithm MaxChild have been considered. These time-based scheduling algorithms only compare the burst time of tasks. Based on the burst time, these schedulers, schedule the sub-tasks of the application on suitable virtual machines according to the scheduling criteria. During this process, not much attention was given to the proper utilization of the resources. A novel dependency and time-based scheduling algorithm is proposed that considers the parent to child (P2C) node dependencies, child to parent node dependencies, and the time of different tasks in the workflows. The proposed P2C algorithm emphasizes proper utilization of the resources and overcomes the limitations of these time-based schedulers. The scientific applications, such as CyberShake, Montage, Epigenomics, Inspiral, and SIPHT, are represented in terms of the workflow. The tasks can be represented as the nodes, and relationships between the tasks can be represented as the dependencies in the workflows. All the results have been validated by using the simulation-based environment created with the help of the WorkflowSim simulator for the cloud environment. It has been observed that the proposed approach outperforms the mentioned time and dependency-based scheduling algorithms in terms of the total execution time by efficiently utilizing the resources.

Keywords: workflow management; workflow scheduling; scientific applications; cloud computing; workflowSim; MaxChild and Scheduling Algorithms

check for updates

Citation: Prakash, V.; Bawa, S.; Garg, L. Multi-Dependency and Time Based Resource Scheduling Algorithm for Scientific Applications in Cloud Computing. *Electronics* **2021**, *10*, 1320. https://doi.org/10.3390/electronics10111320

Academic Editors: Eva Marín-Tordera, Vitor Barbosa Souza and Jordi Garcia

Received: 22 April 2021
Accepted: 26 May 2021
Published: 31 May 2021

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Copyright: © 2021 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

Most of the business processes [1] can be represented in terms of a workflow. Workflow can be defined as a non-directed acyclic graph (DAG) [2] based structure having a group of connected tasks in a parent to child relationship. There is no parent to parent or child to child relationship in a workflow. It means that the tasks on the same level are not connected to each other, and the connectivity of tasks can be done from higher levels to lower levels only. The task invocation, synchronization, and information flow between the different tasks can be represented in a specific order described by the workflow management [3]. A scientific workflow management system (WMS) [4] is used to specify and execute the processing of the complex data. The biggest problem for WMS is scheduling because it is very difficult to identify the resource availability in the central pool of resources at the time of execution. Workflow scheduling is a challenging job as a proper sequence of the workflow tasks for execution needs to be created. The mapping and management of a workflow's tasks on shared resources is done with the help of scheduling [5,6].

Workflow management systems [3,7] are basically used for appropriate management and execution of the workflow tasks. The WMS mainly consists of five major entities, including workflow design, information retrieval techniques, scheduling of workflow tasks, fault tolerance, and data movement as depicted in Figure 1.

Figure 1. Elements of workflow management system [7].

A workflow structure [8] mainly describes the connection between different tasks of a workflow. It can be of two types such as DAG and Non-DAG based structures. DAG-based workflow structure can further be classified into three main categories: sequence, parallelism, and choice.

Non-DAG structures are superset of DAG and include the iteration pattern [9] in which the tasks can be executed iteratively. The overall design of a workflow can be explained with the help of workflow structure, its model and composition system as shown in Figure 2. The workflow model [10] is used to define the workflow structure and is of two types; abstract and concrete [11]. Workflow composition system helps user to add different components to the workflow [2,12,13].

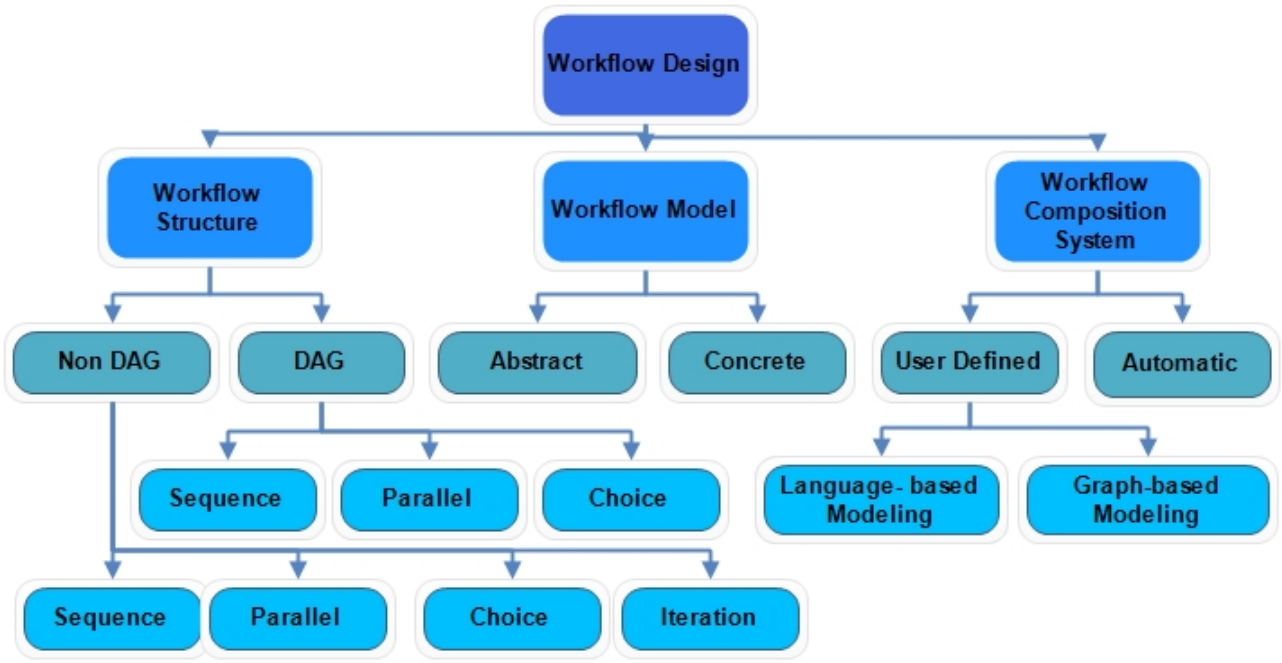


Figure 2. Aspects of workflow design [7].

1.1. Existing Challenges

Scheduling workflow's tasks to appropriate cloud resources is a challenging job as it depends on the QoS requirements of the cloud applications. In cloud computing, due to heterogeneity; uncertainty and resource mobilization; resource scheduling is a current area of research in demand. Different criteria for scheduling various resources and the parameters, requires different categories of resource scheduling techniques [14]. Further, major challenges in the workflow scheduling are: (1) how to assign appropriate cloud services to each workflow's task; (2) how to deal with cloud infrastructure variability; (3) how to consider the limits of concurrency in case of multiple tasks running in parallel; (4) how to solve the problem of data transfer between different workflow's task, etc. [15]. According to the authors [11] the challenges are: (5) resource management, economic barriers, such as costs for migrating workflow's task from one cloud service provider (CSP) to another; (6) legal issues such as data location or destination can be defined in advance; (7) security issues such as single sign-on authentication method in the inter-cloud environments, monitoring cloud resources, portability to move them from one cloud to another, and service-level agreement (SLA) having global SLAs between a federation and its customers are common issues.

1.2. Research Contribution

The major contribution toward resource scheduling in cloud computing is the proposed P2C algorithm. The proposed P2C algorithm is used to overcome the limitations of the existing time and dependency-based schedulers for scientific applications. The proposed algorithm considers the following parameters at a time:

1. Parent to child node dependencies (which parent has the maximum number of child nodes);
2. Child to parent node dependencies (which child has the minimum number of parent odes);
3. Time of different tasks (in case of a tie in the second condition, as mentioned above) present at different levels in the workflow. The task having maximum time in the ready queue will be scheduled in case of the same dependency ratio.

The key objectives of the proposed algorithm are: (1) to reduce the overall execution time and (2) to utilize the resources efficiently. The proposed P2C algorithm has been

validated by using the simulation-based environment created with the help of the WorkflowSim simulator. It has been observed that the proposed P2C algorithm outperforms the existing time and dependency based schedulers in terms of total execution time by utilizing the resources efficiently for the existing scientific applications.

1.3. Paper Organization

The rest of the paper is divided into six sections. In Section 2,the existing literature review and all the components of the workflow scheduling along with their execution environment are discussed. Section 3 describes in detail various scientific applications and their architectures. In Sections 4, the problem formulation is discussed. A detailed description of the methodology used to solve the problem, along with the working of proposed P2C algorithm is given in Section 5. Section 6 describes the experimental setup and the workflows in detail with the statistical result analysis. Finally, Section 7 concludes the overall work with the future scope of the paper.

2. Workflow Scheduling

One of the main issues of cloud computing is resource scheduling. Users can use the cloud services anytime from anywhere with the help of a stable internet connection. However, users will not have direct access to the cloud resources; instead, special application programming interfaces (API) should assist the resources on-demand. Allocating resources for cloud computing to a significantly changing request for the resources based on end-user application's usage pattern is a big challenge. The primary purpose is not only to optimize the resource allocation applications but also to improve resource utilization. There are numerous resource allocation algorithms, models, frameworks, and policies. These assist in allocating or transferring the resources that have proven to be helpful for both cloud service consumers (CSC) and cloud service providers (CSP). Although there are some conditions for resource scheduling, which are not appropriate for resource allocation to customers, these conditions can be highlighted in terms of factors such as: (1) if the cloud has a limited number of resources, then resource shortage may occur; (2) if CSPs has fewer resources for the end-user based on the policies and procedures; (3) allocation of additional resources based on customer's ongoing applications may violate processing strategies; (4) if more than two end-users try to get the same resource simultaneously, then resource congestion may occur; (5) if substantial resources are available within the cloud, but the cloud applications does not assign them to the relevant CSCs request, then resources may be lost [16].

Workflow scheduling [17] finds a correct sequence of task execution by following the scheduling criteria. The components of workflow scheduling are shown in Figure 3. Scheduling architecture is very important when it comes to the quality, efficiency, and effectiveness of a project. The layout of scheduling architecture is organized into three categories: centralized, decentralized, and hierarchical. In the centralized work environment, one central scheduler takes all scheduling decisions for all the workflow based activities.Whereas in a decentralized approach, there are multiple schedulers but no central controller for these different schedulers to help them to communicate with each other. However, in the case of hierarchical scheduling, there is a central controller which is used to control not only the workflow execution but also the sub-workflows assigned to the lower-level schedulers [18,19].

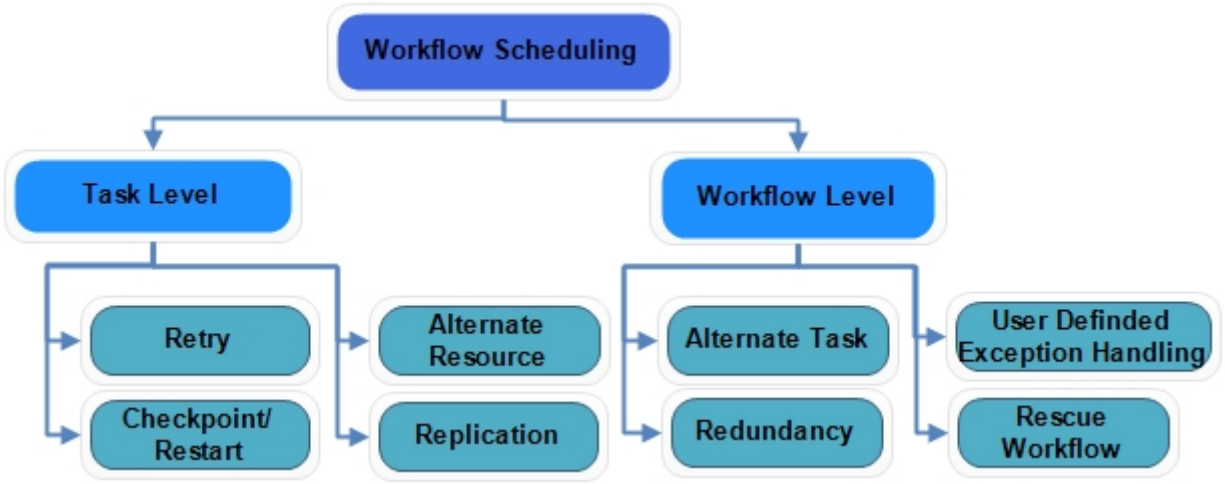


Figure 3. Components of workflow scheduling [20].

Scheduling decisions [21] are divided into two types, such as local and global. Decisions made based on the work or sub-employment are known as local decisions, whereas the decisions based on all employment are called global decisions. Global decision-making processes provide better overall results because only one job or fewer tasks in the workflow are considered in a local decision-making process [22].

Abstract workflow models can be transformed into concrete models by using two schemes: static and dynamic. In the static scheme, the dynamic changes in resources are not considered. Whereas in the case of dynamic schemes, concrete models are generated before execution of workflow with the help of static information, such as the execution environment [23]. Static schemes can further be categorized into two types: user-directed and simulation-based [24]. The dynamic scheme can further be divided into prediction based as well as just-in-time. This scheme considers both static and dynamic information about resources used for making scheduling decisions at run-time. Prediction-based planning scheme considers dynamic information along with some results based on prediction and just-in-time scheme makes a decision at the execution time only [8,25,26].

Performance driven scheduling strategies [27] achieve the highest performance for the user-defined QoS parameters as the workflow's tasks mapped with the resources that give the optimal performance [28]. Most operational scheduling strategies focus on increasing the scope of the workflow. Market-driven scheduling strategies focus on resource availability, cost allocation, quality budget, and time frames. The market model used to organize the workflow has become an available resource that leads to lower costs. The trustees' strategy focuses on security and reputation for the resources [29,30].

Effective resource scheduling techniques are used to reduce the execution cost and the execution time; power consumption and deal with other QoS requirements [31]. The QoS requirements [32] may consist of quality attributes, such as reliability, security, availability, and scalability, etc. Resources are seen to be more challenging because both the CSCs and CSPs are not ready to share information. The first objective of resource scheduling is to identify adequate resources requirements for appropriate and effective utilization of resources. The second objective of resource scheduling is to identify the sufficient and proper workload and resources to support scheduling multiple workloads to meet many QoS requirements. Therefore, resource scheduling considers the execution time of different workloads. The overall performance depends on the different types of workloads (heterogeneous) and QoS requirements with similar workload (homogeneous) [14].

The authors [33] considered the task scheduling on clouds as a three queue (TQ) process based on three queues and dynamic preferences. First of all, scheduling has been decided on all the tasks in the available queue based on the priority of tasks. Secondly, subsequent tasks are divided separately into different groups based on input data and output data volume, the number of nodes currently running, tasks completion time, disk

I/O rate, etc. Finally, the tasks are rearranged into the ready queue by considering all the parameters defined in the second queue.

The authors [34] proposed a solution to maintain a random cloud computing network. The goal is to satisfy QoS parameters by maximizing resource utilization. The proposed scheme increases the resource utilization, as well as reduces the resource consumption and execution time of the applications. However, there is a need for a mapping mechanism between the already allocated resources and the minimum requirement of resources to complete the execution process.

The authors [35] proposed a game-theoretical framework for real-time task scheduling in the cloud computing. In the proposed model, the task acts as a player, and the VM acts as a tactic, and the player's payoff indicates the player's completion time and waiting time. The proposed model is very effective in reducing the total time and waiting time of all the tasks in the workflow.

A new convenient and dynamic scheduling algorithm is required in the cloud environment for efficient resource utilization. Some of the existing static algorithms include first in first out (FIFO) [36], shortest job first (SJF) [37], round robin (RR) [38], etc. In contrast, dynamic algorithms do not require any advanced information about VMs and tasks; however, the VM requires constant monitoring. These algorithms are more accurate, efficient, and appropriate in cloud environments. When any VM is overloaded, so the work being done on this VM can be intentionally transferred to an under-loaded VM [39]. Dynamic RR [40], heterogeneous early finish time (HEFT) [41], etc., are a few examples of dynamic scheduling algorithms widely applied in cloud environments.

A novel multi-objective workflow scheduling approach in IaaS clouds [42] is proposed to optimize the cost and workflow's makespan for real-world scientific applications. The authors [43] proposed a multi-purpose workflow-scheduling algorithm based on decomposition. The proposed model uses Pareto front solutions to achieve at least as good as scheduling results instead of repeatedly implementing the single-objective scheduling algorithm with multiple constraints. Furthermore, the authors [44] proposed a dynamic fault-tolerant workflow scheduling (DFTWS) approach with hybrid spatial and temporary re-implementation schemes. Firstly, DFTWS calculates the time characteristics of each task and predicts the critical path of the workflows. Secondly, DFTWS identifies the appropriate virtual machine (VM) for each task according to the task requirement and budget quota in the initial resource allocation phase. Finally, DFTWS manages online scheduling, making real-time-error-tolerant decisions based on the failure type and task critique during the workflow execution.

The authors [45] proposed a new, budget deadline aware scheduling (BDAS) algorithm that addresses the scheduling of workflows under the budget and deadline constraints for scientific workflows in the IaaS cloud. The proposed heuristic satisfies the budget and time constraints, while the cost–time trade-off is introduced over heterogeneous cases.

The authors [46] proposed a novel resource prediction-based scheduling technique that automates the allocation of resources for scientific application in a virtualized cloud environment. Firstly, the proposed predictive model is trained on datasets that simultaneously generate the functions of a scientific application in the cloud. Secondly, resources are determined based on the production of the proposed estimation model. The main goal of the resource prediction-based scheduling technique is to efficiently manage resources for virtual machines, reduce execution time, and improve error rates and accuracy. Additionally, to manage fluctuating demand for resources, resources need to be managed efficiently. Furthermore, the authors [47] focuses on the design of a prediction-based scheduling approach that maps the function of a scientific application with appropriate VMs by combining the features of swarm intelligence and a multi-objective decision-making process. The proposed approach is used to improve accuracy rate, execution time, cost, and SLA breach rate.

The initial idea of Spring scheduling in terms of computer architecture has been proposed by the authors [48,49]. According to the authors, Spring scheduling co-processor

or multiprocessor have been considered as very large-scale integration (VLSI) accelerator for real-time system. Co-processors can be used for both static and dynamic scheduling. Many different approaches and their combinations can be used with the help of Spring scheduling, such as highest value first, earliest deadline first, and earliest available time first, etc. The Spring scheduler works on parallel structure for standard scheduling of several tasks, number of resources, and the internal criteria of scheduling. Trakadas et al. [50] defined the basic building parts and levels of a decentralized hybrid cloud MEC architecture that results in a platform-as-a-service (PaaS). The stakeholder ecosystem is also examined in order to provide a wide view on the business prospects of the platform.

3. Scientific Applications

Scientific applications are used to simulate real-world activities using mathematics. Real-world objects are turned into mathematical models, and their actions are simulated by executing the formulas. In this section, the various scientific applications used for the evaluation purpose are described in detail. These scientific applications are represented in terms of the workflows.

3.1. CyberShake

CyberShake workflow [51] is used to identify earthquake hazards by identifying the earthquake ruptures having a moment magnitude value greater than 6. CyberShake workflow is parallel and can be represented in 5 levels as shown in Figure 4.

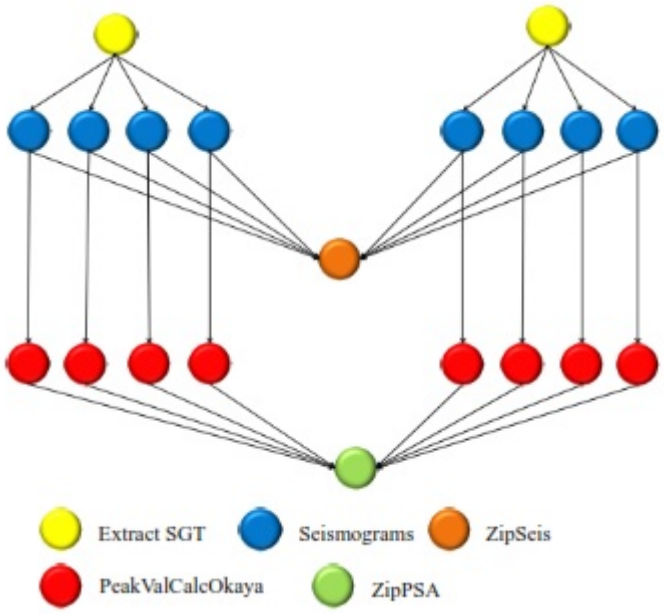


Figure 4. CyberShake workflow.

3.2. Montage

The montage application [52,53] combines together many input images to create sky mosaics using input images in the flexible image transport system (FITS) format. The montage application has pipeline structure and comprising of nine levels, as shown in Figure 5.

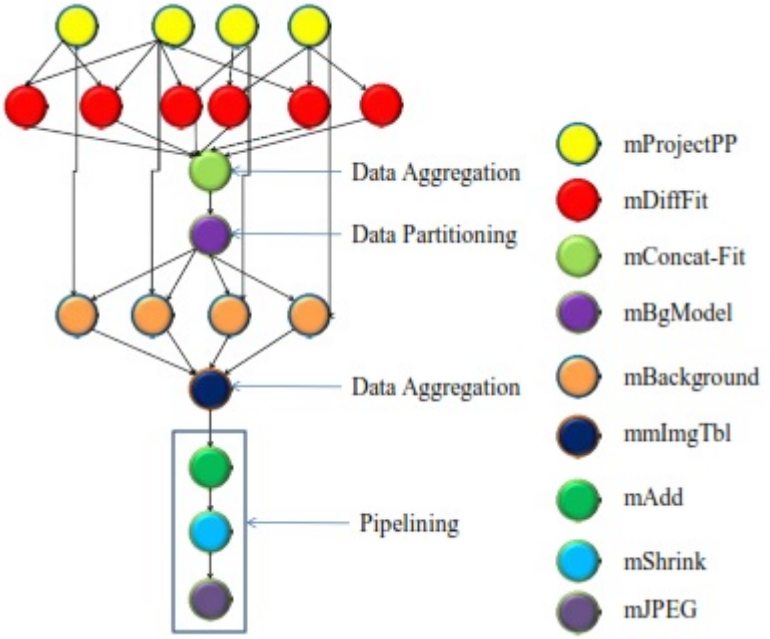


Figure 5. Montage workflow.

3.3. *Epigenomics*

The epigenomics [52,53] workflow is used to automate various operations in genome sequence processing. Epigenomics workflow also has a pipeline structure and has eight levels, as shown in Figure 6. The overall input to the workflow are sequential data obtained for multiple "lanes" from the genetic analysis process.

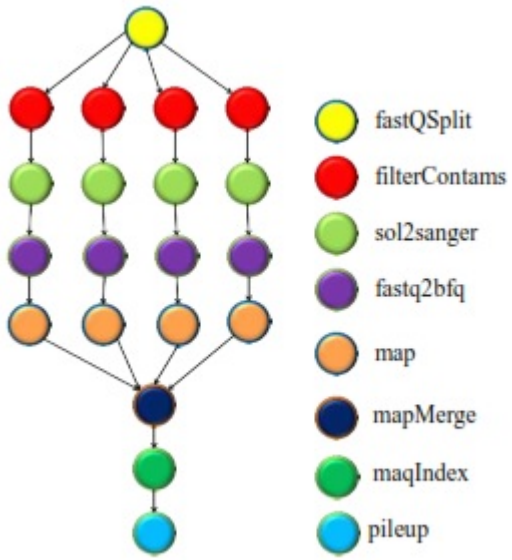


Figure 6. Epigenomics workflow.

3.4. *Inspiral*

Inspiral workflow [52] is used to generate and analyze dynamic gravitational waveformats from data collected during the integration of integrated binary systems. Inspiral workflow has a parallel and pipeline structure and has six levels, as shown in Figure 7.

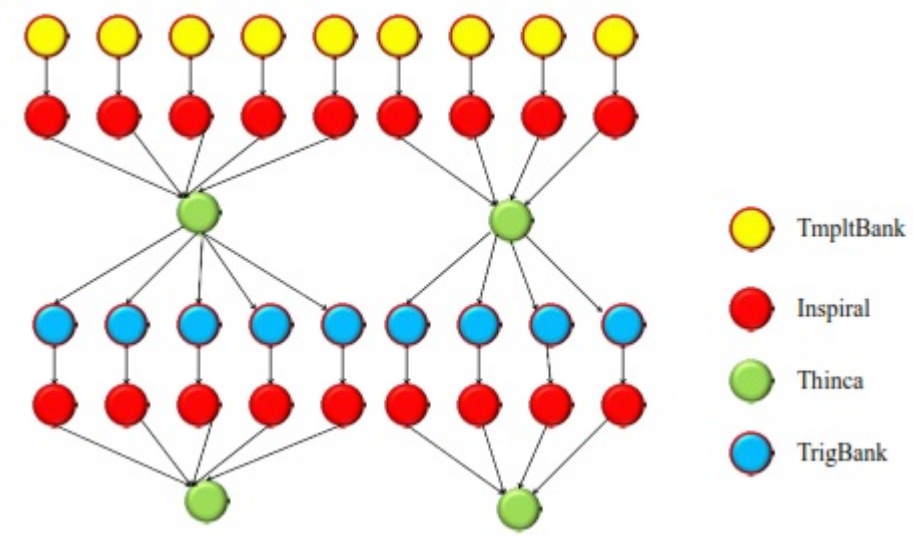


Figure 7. Inspiral workflow.

3.5. SIPHT

The SIPHT workflow [52,53] is used to automatically perform searches for uninterrupted RNA (sRNA) viruses in the National Center for Biotechnology Information (NCBI) database. All SIPHT workflows have almost identical structures, and larger workflows can be made by combining smaller independent workflows. The only difference is in the structure of any two events in the number of Patser's works. The SIPHT workflow has a pipeline structure, as shown in Figure 8.

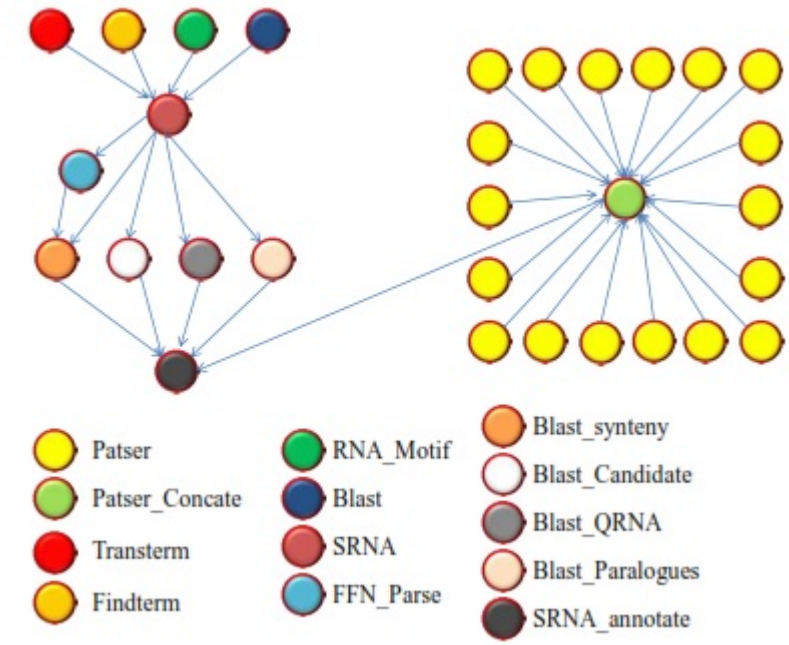


Figure 8. SIPHT workflow.

4. Problem Formulation

In this section, the research problem has been formulated using the following notations:

Let us consider that W denotes the list of workflows where individual ith workflow is denoted by w_i. Each workflow consists of a group of jobs where ith job at kth level is denoted by j_i^k.

Let us consider that the list of virtual machines (resources) be denoted by VM where individual ith virtual machine is denoted by vm_i.

Let 'C' be the category list of workflows and jobs, where c_i denotes the individual category.

Definition 1. *Let us consider that f_1 denotes one to one mapping function between a job and its burst time.*

$$f_1 : \{j_i^k \rightarrow \Delta t_b j_i^k \in w_k\} \tag{1}$$

where Δt_b is the burst time of job j_i. There exists dependencies between different jobs present in a workflow.

Definition 2. *Let us consider that f_2 be a one to many mapped function between jobs at different levels.*

$$f_2 : \{j_i^{k1} \rightarrow j_l^{k2} j_i^{k1}, j_l^{k2} \in w_m \ \ \wedge \ \ k_1 < k_2\} \tag{2}$$

Definition 3. *Let us consider that f_3 be a one to one mapped function between workflow and its execution time.*

$$f_3 : \{w_i \rightarrow \Delta t_e^{w_i} w_i \in W\} \tag{3}$$

where Δt_e^w is the total execution time of the workflow w_i.

Definition 4. *Let us consider that f_4 be a one to one mapped function between job and its execution time.*

$$f_4 : \{j_i \rightarrow \Delta t_e^{j_i} j_i \in w_k\} \tag{4}$$

where Δt_e^j is the total execution time of the job j_i.

Definition 5. *Let us consider that f_5 be a one to one mapped function between a job and its execution status at time 't'*

$$f_5 : \{j_i^t \rightarrow b1 j_i^t \in w_k \ \ \wedge \ \ b1 \in \{0, 1, 2\}\} \tag{5}$$

Definition 6. *Similarly, f_6 is a one to one mapped function between workflow and its execution status at time 't'*

$$f_6 : \{w_i^t \rightarrow b2 w_i^t \in W \ \ \wedge \ \ b2 \in \{0, 1, 2\}\} \tag{6}$$

where '0' indicates not started yet, '1' indicates pending and '2' indicates executed successfully.

Definition 7. *Let us consider that f_7 be a one to one mapped function between ith virtual machine and its computational power*

$$f_7 : \{vm_i \rightarrow c_k vm_i \in VM\} \tag{7}$$

Definition 8. *Let us consider that f_8 be a one to one mapped function between workflow and its category 'C'*

$$f_8 : \{w_i \rightarrow c_j w_i \in W \ \ \wedge \ \ c_j \in C\} \tag{8}$$

Definition 9. *Similarly, let f_9 be a one to one mapped function between job and its category 'C'*

$$f_9 : \{j_i \rightarrow c_k j_i \in w \ \ \wedge \ \ c_k \in C\} \tag{9}$$

Definition 10. *Let us consider that f_{10} be a one to one mapped function between virtual machine and its occupancy status*

$$f_{10} : \{vm_i \rightarrow b3 vm_i \in VM \ \ \wedge \ \ b3 \in \{0, 1\}\} \tag{10}$$

where '0' indicates not occupied, '1' indicates occupied.

Definition 11. *Let us consider that f_{11} be a one to many mapped function between workflow and its jobs*

$$f_{11} : \{w_i \rightarrow j_k w_i \in W \ \wedge \ j_k \in w_i \ \wedge \ num(j_k) \geq 1\} \tag{11}$$

Definition 12. *Similarly, f_{12} is a one to one mapped function between workflow and its job, where job is the starting job*

$$f_{12} : \{w_i \rightarrow j_k w_i \in W \ \wedge \ j_k \in w_i\} \tag{12}$$

Definition 13. *Similarly, f_{13} is a one to one mapped function between a job and a virtual machine, where the job is assigned to a virtual machine for its execution*

$$f_{13} : \{j_k \rightarrow vm_i j_k \in w_l \ \wedge \ vm_i \in VM\} \tag{13}$$

Definition 14. *Similarly, f_{14} is a one to many mapped function between workflow jobs*

$$f_{14} : \{j_i^{k_1} \rightarrow j_l^{k_2} j_i^{k_1}, j_l^{k_2} \in w_m \ \wedge \ k_1 > k_2\} \tag{14}$$

The objective function of the research work is given in Equation (15)

$$\forall k \ min \sum_{i=1}^{num(w_k)} \Delta t_e^{j_i} \tag{15}$$

subject to constraints

$$c_1 : \forall i, j \quad f_7(vm_i) = f_7(vm_j) \tag{16}$$

$$c_2 : \forall i, m \quad \exists f_9(j_i^k) = f_9(j_l^k) \tag{17}$$

$$c_3 : (f_5(j_i^t) == 2) \Rightarrow f_5(f_2(j_i^t)) = 2 \tag{18}$$

$$c_4 : (f_5(j_i^t) == 1) \Rightarrow f_5(f_2(j_i^t)) = 2 \tag{19}$$

$$c_5 : j_i \in w, w_i \in W, c_i \in C, vm_i \in VM \tag{20}$$

In this section, the problem formulation has been described with the main objectives of the paper, i.e., (1) to reduce the overall execution time of any scientific application, (2) to utilize the resources efficiently, with some constraints such as computational power of each virtual machine is considered to be the same as represented in Equation (16). Furthermore, any job represented as a node in the workflow can only be scheduled for the execution whenever all the predecessors of that node have been executed successfully represented with the help of Equations (18) and (19).

5. Proposed Solution

In this paper, mainly time based scheduling algorithms, such as FCFS, min–min, max–min [54], and (MCT) [55], along with dependency-based scheduler, such as MaxChild [56], have been considered. The MaxChild [56] algorithm is another one-way dependency-based scheduling algorithm that considers the dependency between the parent to child node only. However, there was no consideration given to the dependency between a child to a parent node. To overcome the limitation of these time-based schedulers, multi-dependency, and time-based scheduling algorithm is proposed and compared with the existing approaches. The proposed P2C approach considers the parent to child node relationship (dependencies), child to parent node relationship (dependencies), and the burst time of different tasks present at different levels in the workflow. The main objective of the proposed approach is to reduce the overall execution time of scientific applications, such as CyberShake [51], montage, epigenomics, inspiral, and SIPHT [52], etc., by efficiently utilizing the resources.

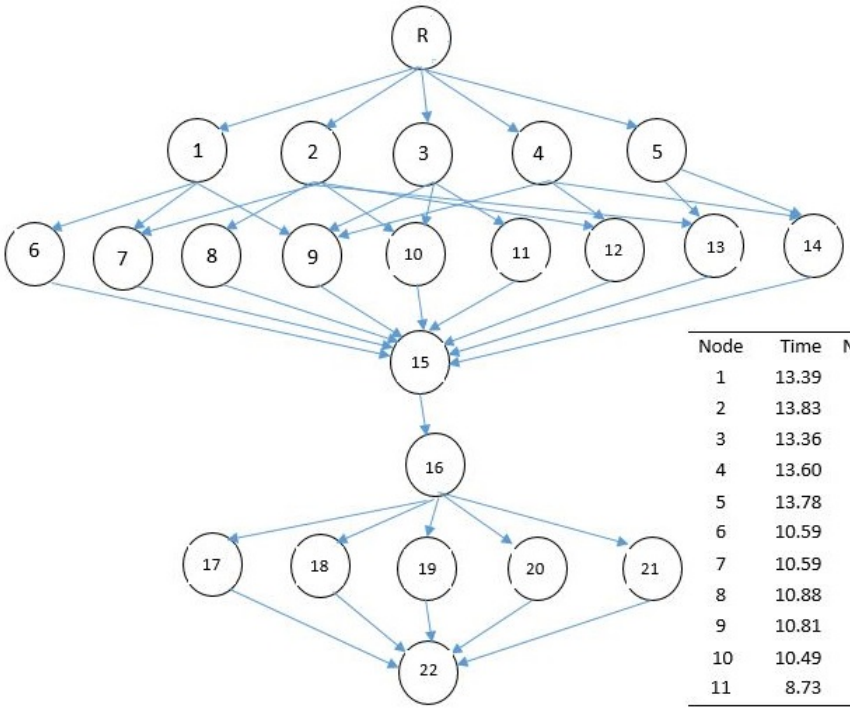
The proposed P2C algorithm is used to overcome the limitations of the existing time and dependency-based schedulers for scientific applications. The proposed approach considers the following parameters at a time:

1. Parent to child node dependencies (which parent has the maximum number of child nodes);
2. Child to parent node dependencies (which child has the minimum number of parent nodes);
3. Time of different tasks (in case of a tie in the second condition, as mentioned above) present at different levels in the workflow. The task having maximum time will be scheduled in case of the same dependency ratio.

The proposed P2C algorithm initially checks whether the tasks in the ready queue are less than available resources, or are very much greater than available resources. In cases such as this, the proposed approach will execute the task in the decreasing order of time. Then the P2C algorithm will schedule those child nodes whose parents have already been successfully completed. Maximum dependencies are considered from parent to child nodes, whereas the minimum number of dependencies are considered from child to parent nodes.

Working of the Proposed P2C Algorithm

The step by step working of the proposed P2C algorithm is explained with the help of Figure 9 and according to the steps given in Algorithm 1. Further, to check the status of the ready queue, the procedure READYQUEUESTATUS is described in Algorithm 2.



Node	Time	Node	Time
1	13.39	12	10.51
2	13.83	13	10.62
3	13.36	14	10.37
4	13.60	15	1.42
5	13.78	16	10.39
6	10.59	17	10.64
7	10.59	18	10.64
8	10.88	19	10.83
9	10.81	20	10.93
10	10.49	21	10.76
11	8.73	22	8.73

Figure 9. Montage Workflow for Dataset File Montage_25.

Montage workflow has been considered to have 25 nodes from the dataset file Montage_25. Further, an execution overhead of 0.21 ms has been added to execute the root node represented by R. The tasks numbered from 22 to 25 have been combined into a single task due to the montage workflow pipeline structure in Figure 5. The step by step description of the P2C algorithm for the execution process is given as follows:

Execution 1:

Step 1: Creation of the parent–child table to find the child nodes of each task in the workflow at level 1.

First of all, the proposed P2C algorithm will check the parent to child dependencies from the first level. In level 1, there are five tasks numbered from 1 to 5. Therefore, the parent–child table, Table 1 has been created to represent the parent–child dependencies.

Table 1. Creation of the parent-child table at level 1.

Parent	1			2					3			4			5	
Child	6	7	9	7	8	10	12	13	9	10	11	9	12	14	13	14

Step 2: Sorting the ready queue based on child to parent dependencies.

For *Montage*_25 Dataset file, there are 5 VMs available at a time to execute the tasks of a workflow. In level 1, there are only five tasks to be scheduled on 5 VMs. Further, there is a need to sort the tasks according to child–parent node dependencies. Whereas, in the case of more than one task, there is the need to sort the tasks based on the burst time in decreasing order, as depicted in Table 2.

Table 2. Sorting of parent in decreasing order of children at level 1.

Parent	2					4			1			3			5	
Child	7	8	10	12	13	9	12	14	6	7	9	9	10	11	13	14

Correct sequence of tasks in the ready queue will become in the order {2, 4, 1, 3, 5} after sorting the tasks based on the number of parents, as well as decreasing order of time.

Step 3: Scheduling of tasks to appropriate VM.

With the help of Step 1 and Step 2, the ready queue of tasks have been found for the scheduling. At the same time the final list of VMs have also been finalized. The next step is to schedule task 2 $\rightarrow VM1$, task 4 $\rightarrow VM2$, task 1 $\rightarrow VM3$, task 3 $\rightarrow VM4$, task 5 $\rightarrow VM5$. After the first run, tasks numbered from 1 to 5 have been executed and tasks 6–14 are in the available queue. Now, execute all the steps of the algorithm until all the tasks have finished their execution successfully.

Execution 2:

Step 1–2: Creation of the parent–child table to find the child nodes of each task in the workflow at level 2

Each task has only one child, i.e., task 15. Therefore, there is a need to sort the tasks numbered from 6 to 14 in decreasing order of time. Hence, the correct sequence of the tasks in the ready queue will become in the order {8, 9, 13, 6, 7, 12, 10, 14, 11}.

Step 3: Scheduling of tasks to appropriate VM

Schedule task 8 $\rightarrow VM5$, task 9 $\rightarrow VM4$, task 13 $\rightarrow VM3$, task 6 $\rightarrow VM2$, task 7 $\rightarrow VM1$. In this step, we have reversed the VM allocation sequence due to the availability of both free VM and un-allocated tasks in the same order. After the second run, tasks numbered from 1 to 9 and task 13 have been executed successfully and tasks numbered from 10 to 12 and 14 are in the available queue. Hence, the correct sequence for the task in the ready queue will appear in the order {12, 10, 14, 11}.

Execution 3:

Step 1–3: Schedule task 12 $\rightarrow VM1$, task 10 $\rightarrow VM2$, task 14 $\rightarrow VM3$ and task 11 $\rightarrow VM2$

After the third execution, tasks numbered from 1 to 14 have been completed successfully, and tasks 15 is available in available as well as ready queue. There is a need to schedule task 15 on VM1.

Execution 4:

Step 1: Creation of the parent-child table to find the child nodes of each task in the workflow at level 4

After the fourth execution, tasks numbered from 1–15 have been executed, and tasks 16 is available in available and ready queue. The initial level parent–child table for level 4 can be shown in Table 3.

Table 3. Creation of the parent-child table at level 4.

Parent			**16**		
Child	17	18	19	20	21

Step 2: Sorting the ready queue based on child to parent dependencies

After sorting the tasks based on the number of parents and decreasing order of time, updated table can be represented in Table 4.

Table 4. Sorting of parent in decreasing order of children at level 4.

Parent			**16**		
Child	20	19	21	17	18

Execution 5:

After the fifth execution, tasks numbered from 1–16 have been executed, and tasks numbered from 17–21 are available to execution. Hence, the correct sequence of the tasks in the ready queue will become in the order {20, 19, 21, 17, 18}.

Step 1–3: Scheduling of tasks to appropriate VM.

Schedule task 20 $\rightarrow VM1$, task 19 $\rightarrow VM2$, task 21 $\rightarrow VM3$, task 17 $\rightarrow VM4$, task 18 $\rightarrow VM5$. After the sixth execution, tasks numbered 1–21 have been executed, and task 22 is available in available queue and ready queue. Further, there is need to schedule the task 22 on any of the free VM.

After seventh execution, tasks numbered from 1–22 have been executed and the P2C algorithm has also been stopped due to execution of all the tasks. Hence, overall execution time will be the sum of execution time consumed at each step of the algorithm.

$$ET\_Total = ET_1 + ET_2 + ET_3 + ET_4 + +ET_5 + ET_6 + ET_7 + Root\ node\ overhead \quad (21)$$

$$ET\_Total = 13.83 + 10.88 + 10.51 + 1.42 + 10.39 + 10.93 + 8.73 + 0.21$$

$$ET\_Total = 63.9 \text{ ms}$$

The simplified step by step description of the algorithm is represented with the help of flowchart as shown in Figure 10.

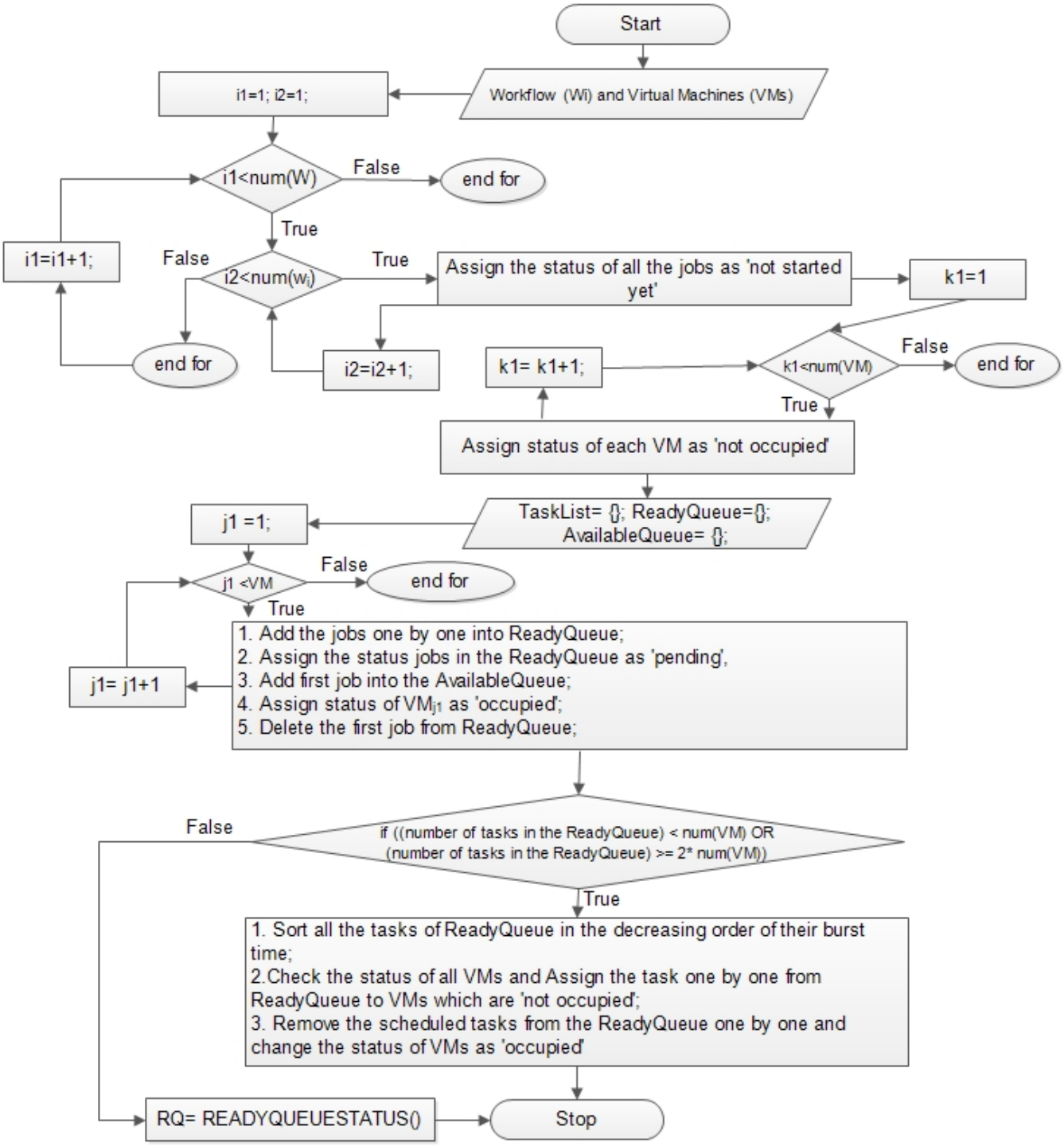


Figure 10. Flowchart of proposed parent to child (P2C) Algorithm.

6. Experimental Setup and Results

Cloud applications have different requirements for configuration and deployment [57]. All the experiments have been conducted on a workstation having a 64 bit Windows 10 operating system, 6 GB memory, and Intel (R) Core (TM) i5-3367U @ 1.8GHz CPU. This section describes the details related to the cloud resources, simulation environment, workflows dataset, and results analysis.

Algorithm 1 Proposed P2C Algorithm.

Input: Workflow (w_i) and Virtual Machines (VMs)
Output: Execution Time (Δt_e)

for $i1 \leftarrow 1$ to $num(W)$ **do**
  for $i2 \leftarrow 1$ to $num(w_i)$ **do**
    $f_5(j_{i2} = 0)$
  end for
end for
for $i1 \leftarrow 1$ to $num(VM)$ **do**
  $f_{10}(vm_{i1} = 0)$
end for
TL={}
RQ={}
AQ={}
for $i1 \leftarrow 1$ to $num(W)$ **do**
  RQ = RQ + $f_{12}(w_i)$
  $vm_1 \leftarrow rq_1$
  $f_5(rq_1) = 1$
  AQ = AQ+ {rq1}
  $f_{10}(vm_1) = 1$
  RQ = RQ-{rq1}
  while $f_5(f_{11}(w_i))!$ = 2 **do**
    if (num(RQ) - $\sum_{\forall k}(f_{10}(vm_k)) \leq 0$) (num(RQ) $\geq 2 \times \sum_{\forall k}(f_{10}(vm_k))$) **then**
      RQ = READYQUEUESTATUS()
      if num(RQ) $\neq 0$ **then**
        $RQ = Sort(RQ, num(f_1(rq)), 1)$
        for $i1 \leftarrow 1$ to $num(VM)$ **do**
          if $f_{10}(vm_{i1}) == 0$ **then**
            $vm_{i1} \leftarrow rq_1$
            $f_5(rq1) = 1$
            $RQ = RQ - \{rq_1\}$
            $f_{10}(vm_{i1}) = 1$
            $AQ = AQ + \{rq1\}$
          end if
        end for
      end if
    else
      RQ = READYQUEUESTATUS()
      if num(RQ) $\neq 0$ **then**
        $RQ = Sort(RQ, num(f_2(rq)), 1)$
        for $i1 \leftarrow 1$ to $num(VM)$ **do**
          if $f_{10}(vm_{i1}) == 0$ **then**
            $vm_{i1} \leftarrow rq_1$
            $f_5(rq1) = 1$
            $RQ = RQ - \{rq_1\}$
            $f_{10}(vm_{i1}) = 1$
            $AQ = AQ + \{rq1\}$
          end if
        end for
      end if
    end if
  end while
end for

Algorithm 2 Ready queue check condition.

```
procedure READYQUEUESTATUS
    for i1 ← 1 to num(AQ) do
        if aq_i1 = job's execution completed then
            f5(aq_i1) = 2
            f10(f13(aq_i1)) = 0
            AQ = AQ − aq_i1
            TQ = TQ + f2(aq_i1)
        end if
    end for
    for i1 ← 1 to TQ do
        RQ = RQ + [f5(f14(tq_i1)) == 2 ? tq_i1 : {}]
    end for
end procedure
```

6.1. Cloud Resources

A resource can be defined as a physical or logical component that is connected to a computer system. Cloud resources are mainly classified as fast computing, storage, communication, power, and security, etc. Every device which is connected to a computer system is considered to be a resource. Figure 11 represents the detailed view of various cloud resources which are provided on the user's request through internet.

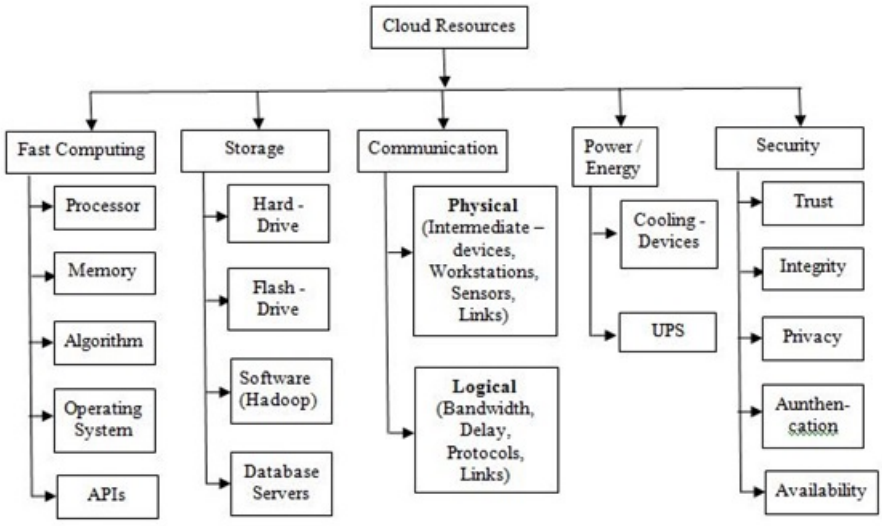


Figure 11. Classification of cloud resources [58].

These resources are mainly classified in two categories: physical resources and logical resources. Physical resources consist of the system's hardware such as processor, memory, and other peripheral devices connected to the system. Logical resources control the physical resources on a temporary basis and mainly consist of an operating system, power sources, APIs, databases, networking bandwidth, and protocols, etc., [58,59]. Cloud framework can be heterogeneous for the proper utilization of homogeneous, as well as heterogeneous resources. Further, quality of service (QoS), specific runtime and virtual technology, can be utilized to provide application's objectivity and the virtualization.

6.1.1. Fast Computing Utility

These resources provide the computational power to the users for the execution of their applications. Cloud computing provides computation as a service (CaaS) as a utility to cloud users. These resources can be treated as the collection of physical machines, mainly processing power, memory, algorithms, operating system, and APIs. To provide these resources to users, the physical machines are deployed on the virtual environment, which is considered as a virtual machine [58,59].

6.1.2. Storage Utility

One of the main issues of end-users is to store their data into any of the convenient storage media, such as hard disk, floppy disk, and flash drive, etc. However, instead of purchasing their own storage space, it is better to take the storage space from CSPs on rent. Therefore, the cloud users store the data and information into an external remote database servers. Data can be accessed and transferred from the database servers to the user's system through internet connectivity [58,59].

6.1.3. Communication Utility

It is also known as network utility or network as a service (NaaS). Communication utility consists of physical resources, such as intermediate devices, sensors, workstations, and logical resources, such as bandwidth, delay, protocols, and communication links. From the networking perspective, intermediate devices and communicating devices consist of modems, hubs, routers, and switches, etc. Further, the networking resources installed on physical machines within a data center are mainly organized in clusters [58,59].

6.1.4. Power Utility

Cloud computing consists of thousands of data servers and various types of other interconnecting devices. Energy efficiency is one of the main issues due to a lot of power consumption for the overall functioning of these service utilities. The power is consumed by the data servers and by cooling and supporting infrastructure, power distribution equipment, and networking equipment. Data centers usually take power from power utility providers, such as local power storage, especially from renewable energy sources including wind and solar energy [58,59].

6.1.5. Security

Cloud users demand various service utilities like IaaS, PaaS, and SaaS from the cloud provider. Therefore, these services must be highly secure, reliable, and available. Trust, integrity, privacy, authentication, and availability must be considered for the perspective of issues in security [58,59].

6.2. Workflows Dataset

There is no cost of infrastructure and services needed to test these applications in a repeatable and controlled environment. Initially, less number of virtual machines are created for the execution of smaller workflows. Later on, the number of virtual machines can be increased dynamically according to the execution of larger workflows. The different test cases are created by using the smaller as well as larger workflows as listed in Table 5.

Table 5. Creation of Test Cases for different size of different workflows.

Workflow	Number of Task	Number of VM	Dataset File
CyberShake	30	5	CyberShake_30
	50	10	CyberShake_50
	100	20	CyberShake_100
	1000	50	CyberShake_1000
Montage	25	5	Montage_25
	50	10	Montage_50
	100	20	Montage_100
	1000	50	Montage_1000
Epigenomics	24	5	Epigenomics_24
	46	10	Epigenomics_46
	100	20	Epigenomics_100
	997	50	Epigenomics_997
Inspiral	30	5	Inspiral_30
	50	10	Inspiral_50
	100	20	Inspiral_100
	1000	50	Inspiral_1000
SIPHT	30	5	SIPHT_30
	60	10	SIPHT_60
	100	20	SIPHT_100
	1000	50	SIPHT_1000

6.3. Simulation Environment

The simulation environment evaluates a different kind of resource leasing on the provider's side under different conditions with different load distribution. Implementation of the existing and proposed scheduling algorithms have been done on the WorkflowSim [60] simulator by setting it up on Net Beans IDE. The homogeneous virtual machines used in the simulation process have 512 MB memory, a CPU with 1000 MIPS, a bandwidth of 1000 BPS and 10 GB of image size. WorkflowSim extends the existing CloudSim [61] simulator that allows modeling and matching the cloud environment, data centers, virtual machines, and cloudlets, tackling only one load but is not suitable for scheduling workflows as many tasks need to be integrated. WorkflowSim, therefore, provides a higher level of workflow management over the CloudSim layer.

6.4. Statistical Analysis

Statistical analysis is the collection and interpretation of data to uncover patterns and trends. It is a part of data analytics. Statistical analysis can be used in collecting research descriptions, statistical modeling or designing surveys and studies. There are two main types of statistical analysis, i.e., descriptive and inference. After collecting the data, there is a need to analyze it by extracting the data, such as creating a pie chart, line graph, bar graph, and histogram, etc. All of these come down to using the right methods for statistical analysis, i.e., processing and collecting data samples to uncover patterns and trends. For this analysis, there are five different heuristics, such as average, standard deviation, regression, hypothesis test, and sample size determination [62]. The following terms are used in the descriptive analysis for this research:

Definition 15. *Mean, Standard Deviation, and Variance The mean (average) of a dataset is obtained by adding all the numbers in the dataset divided by the number of values in the dataset. A standard deviation is a measure of the distribution of a dataset from its mean. It measures the absolute diversity of a distribution; the higher the distribution or variance, the higher will be the standard deviation and the greater will be the magnitude of deviation of the value from its mean. The variance measures how well the dataset is distributed. A variance of zero indicates that all data values are the same. However, a high variance indicates that data points are still widely distributed from the mean and each other. The variance can be defined as the average of the squared distances from each point to the mean.*

Definition 16. *Confidence Interval and Margin of Error The confidence interval indicates that the parameter is likely to fall between a pair of values near the mean. Confidence interval measures the level of uncertainty or certainty of samples They are usually built using 95% or 99% confidence levels. The margin of error tells about the percentage of points in results that differ from the actual population. For example, a 95% confidence interval with a 5% margin of error indicates that the actual statistics will be between 5% of the substantial population by 95% of the time. The following null hypothesis is considered for this research:*

H0 = There is a significant difference of size, shape, and structure of the workflow on the proposed scheduling algorithm.

To verify the validity of the proposed algorithm and null hypothesis, the statistic "Two-factor ANOVA without Replication" has been selected and executed successfully at a confidence interval of 95% with 5% a margin of error. The basic difference between the two-factor ANOVA with and without replication is that the sample size is different [63]. In the replication technique, all the sample are mostly unique, and if this happens, then there is a need to calculate the mean independently. In this experiment also, we have different sized workflows as depicted in Table 5.

Definition 17. *Two factor ANOVA without Replication A factor is an independent variable. A level is some aspect of a factor; these are also called groups or treatments.The two factors considered are: factor A (Algorithms) and factor B (Workflow Size). Factor 'Algorithms' has 6 levels (FCFS, min–min, max–min, MCT, MaxChild, and P2C). Factor 'Workflow Size' has 4 levels (e.g., for montage workflow: Montage_25, Montage_50, Montage_100, Montag_1000) as depicted in Table 5. The levels for the factor 'Algorithms' are organized as rows and the levels for factor 'Workflow Size' are organized as columns. The two-factor ANOVA will either test for the main effects of factor 'Algorithms' or factor 'Workflow Size' as represented in Equation (22) or Equation (23) and depicted in Table 6.*

$$H0 : \mu_1 = \mu_2 = \mu_3 \ldots\ldots \mu_m (Factor\ Algorithms) \tag{22}$$

or

$$H0 : \mu_1 = \mu_2 = \mu_3 \ldots\ldots \mu_n (Factor\ Workflow\ Size) \tag{23}$$

Table 6. Two way ANOVA Summary results for CyberShake Workflow.

Factors	SUMMARY	Sum	Average	Variance
Algorithms	FCFS	1566.79	391.69	48,472.73
	Min-min	1608.55	402.13	57,146.5
	Max-min	1499.74	374.93	46,607.81
	MCT	1566.79	391.69	48,472.73
	MaxChild	1563.36	390.84	47,722.96
	P2C	**1456.5**	**364.12**	**49,551.19**
Workflow Size	CyberShake_30	1579.98	263.33	172.31
	CyberShake_50	1581.96	263.66	55.93
	CyberShake_100	1785.53	297.58	305.15
	CyberShake_1000	4314.26	719.04	511.36

Bold numbers are results of proposed approach.

Table 6 indicates that total (sum of all the tasks of CyberShake workflow in all four variants) execution time and average execution time for the proposed P2C algorithm is less as compared to other comparative scheduling. Further, the value of variance for the proposed approach is higher than the other scheduling algorithms. The descriptive results of Two factor ANOVA test are represented in Table 7.

Table 7. Two way ANOVA descriptive results for CyberShake Workflow.

Source of Variation	*SS*	*df*	*MS*	*F*	*p*-Value	*F-crit*
Algorithms	3798.654	5	759.73	7.996	0	2.901
Workflow Size	892,496.7	3	297,498.91	3131.242	3.39E+00	3.287
Error	1425.148	15	95.009			
Total	897,720.5	23				

Since the *p*-Value for the factor Algorithms (0.000 < 0.05) or (*F* = 7.996 > 2.901 = *F-crit*) as depicted in Table 7. Therefore, the null hypothesis is rejected, and there is no significant difference of size, shape, and structure of the CyberShake workflow on the proposed P2C algorithm. Similarly, the summary as well as descriptive results of two-factor ANOVA for montage workflow are depicted in Tables 8 and 9.

Table 8. Two way ANOVA Summary results for Montage Workflow.

Factors	SUMMARY	Sum	Average	Variance
Algorithms	FCFS	866.94	216.73	75,982.15
	Min-min	870.54	217.63	76,802.02
	Max-min	866.08	216.52	75,850.97
	MCT	866.94	216.73	75,982.15
	MaxChild	866.63	216.65	75,903.93
	P2C	**833.23**	**208.3**	**74,158.48**
Workflow Size	Montage_25	340.79	56.79	1.73
	Monatge_50	458.18	76.36	7.33
	Montage_100	606.7	101.11	19.26
	Montage_1000	3764.69	627.44	32.63

Bold numbers are results of proposed approach.

Table 8 indicates that total and average execution time for montage workflow of proposed P2C algorithm is less than other comparative scheduling algorithms. Since the *p*-Value for the factor Algorithm (0.000 < 0.05) or (*F* = 12.752 > 2.901 = *F-crit*) as depicted in Table 9. Therefore, the null hypothesis is rejected, and there is no significant difference of size, shape, and structure of the montage workflow on the proposed P2C algorithm.

Table 9. Two way ANOVA descriptive results for Montage Workflow.

Source of Variation	*SS*	*df*	*MS*	*F*	*p*-Value	*F-crit*
Algorithms	246.771	5	49.354	12.752	5.85373E−05	2.901
Workflow	1,363,981.131	3	454,660.38	117,479.76	5.34794E−33	3.287
Error	58.051	15	3.87			
Total	1,364,285.955	23				

Table 10 indicates that total and average execution time for epigenomics workflow for the proposed P2C algorithm is less or equal to other comparative scheduling algorithms. Since the *p*-Value for the factor Algorithms (0.024 < 0.05) but (*F* = 2.526 < 2.901 = *F-crit*) as depicted in Table 11. Therefore, the null hypothesis for comparative scheduling algorithms with respect to proposed P2C algorithm rejected in some cases while accepted in a majority of the remaining cases. Due to this variable behavior, a significant difference of size, shape and structure of the epigenomics workflow on proposed scheduling algorithm can not

be determined. Further, by carefully examining the total and average execution time for epigenomics workflow, it has been observed that either the proposed P2C algorithm outperforms or behaves similar to some existing algorithms. It could be possible due to the parallel, as well as pipeline, structure of the epigenomics workflow.

Table 10. Two way ANOVA Summary results for Epigenomics Workflow.

Factors	SUMMARY	Sum	Average	Variance
Algorithms	FCFS	161,211.02	40,302.75	2,520,942,674
	Min-min	174,101.15	43,525.28	2,741,892,698
	Max-min	171,923.1	42,980.77	2,713,164,848
	MCT	161,211.02	40,302.75	2,520,942,674
	MaxChild	161,211.02	40,302.75	2,520,942,674
	P2C	161,211.33	40,302.33	2,520,942,674
Workflow Size	Epigenomics_24	33,578.68	5596.44	0.04
	Epigenomics_46	46,459.45	7743.24	0
	Epigenomics_100	230,865.9	38,477.65	15,052,972.16
	Epigenomics_997	682,378.61	113,729.76	10,729,351.4

Table 11. Two way ANOVA descriptive results for Epigenomics Workflow.

Source of Variation	*SS*	*df*	*MS*	*F*	*p*-Value	*F-crit*
Algorithms	43,480,779.16	5	8,696,155.832	2.526	0.024	2.901
Workflow	45,928,839,780	3	15,309,613,260	2688.071	0	3.287
Error	85,430,838.92	15	5,695,389.261			
Total	46,057,751,398	23				

Table 12 clearly indicates that total and average execution time for inspiral workflow of proposed P2C algorithm is less as compared to other comparative scheduling algorithms.

Table 12. Two way ANOVA Summary results for inspiral workflow.

Factors	SUMMARY	Sum	Average	Variance
Algorithms	FCFS	10,800.33	2700.08	4,953,313
	Min-min	12,054.6	3013.65	4,808,698
	Max-min	10,771.88	2692.97	4,884,867
	MCT	10,800.33	2700.08	4,953,313
	MaxChild	10,802.75	2700.68	4,957,883
	P2C	**10,748.33**	**2687.08**	**4,884,225**
Workflow	Inspiral_30	10,469.64	1744.94	17,561.79
	Inspiral_50	9741.8	1623.63	31,777.85
	Inspiral_100	9350.48	1558.41	10,026.64
	Inspiral_1000	36416.3	6069.38	12,884.62

Bold numbers are results of proposed approach.

Since the *p*-Value for the factor Algorithm ($0.000 < 0.05$) or ($F = 40.834 > 2.901 = F\text{-}crit$) as depicted in Table 13. Therefore, the null hypothesis is rejected, and there is no significant difference of size, shape, and structure of the inspiral workflow on proposed scheduling algorithm.

Table 13. Two way ANOVA descriptive results for inspiral workflow.

Source of Variation	*SS*	*df*	*MS*	*F*	*p*-Value	*F-crit*
Algorithms	336,530.6	5	67,306.1	40.834	0	2.901
Workflow Size	88,302,172.05	3	2.9E+07	17,857.59	0	3.287
Error	24,723.993	15	1648.27			
Total	88,663,426.64	23				

Table 14 indicates that total and average execution time of SIPHT Workflow for the proposed P2C algorithm is less compared to other comparative scheduling algorithms. Since the *p*-Value for the factor Algorithm (0.025 < 0.05) or (*F* = 3.471 > 2.901 = *F-crit*) as depicted in Table 15. Therefore, the null hypothesis is rejected, and there is no significant difference of size, shape, and structure of the SIPHT workflow on the proposed scheduling algorithm.

Table 14. Two way ANOVA summary results for SIPHT workflow.

Factors	SUMMARY	Sum	Average	Variance
Algorithms	FCFS	20,100.22	5025.05	1,058,606.8
	Min-min	20,575.48	5143.87	1,589,971.89
	Max-min	18,846.06	4711.51	172,348.68
	MCT	20,100.22	5025.05	1,058,606.8
	MaxChild	20,199.08	5049.77	1,162,325.93
	P2C	**18,375.08**	**4593.77**	**296,019.17**
Workflow	SIPHT_30	26,058.77	4343.12	31,144.96
	SIPHT_60	27,849.2	4641.53	30.09
	SIPHT_100	26,859.25	4476.54	48.18
	SIPHT_1000	37,428.92	6238.15	549,233.12

Bold numbers are results of proposed approach.

Table 15. Two way ANOVA descriptive results for SIPHT workflow.

Source of Variation	*SS*	*df*	*MS*	*F*	*p*-Value	*F-crit*
Algorithms	955,130.555	5	191,026.111	3.471	0.0256	2.901
Workflow Size	14,066,486.6	3	4,688,828.875	36.12	0	3.287
Error	1,947,151.29	15	129,810.086			
Total	16,968,768.5	23				

6.5. Results

Initially, all the scheduling algorithms have been implemented for existing scientific applications such as CyberShake, montage, epigenomics, inspiral and SIPHT with different tasks and different numbers of VMS but having the same structure. The experimental results have been compared with a different number of virtual machines for different sizes of workflows. The existing, as well as proposed, algorithms have been validated on all the test cases listed in Table 5 and results are shown in Figures 12–16, etc.

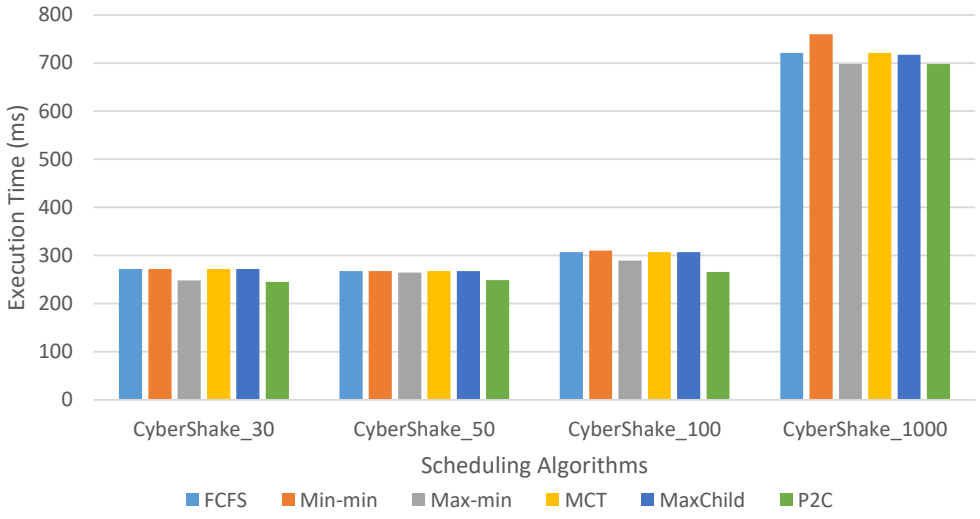


Figure 12. Comparison of execution time for CyberShake workflow.

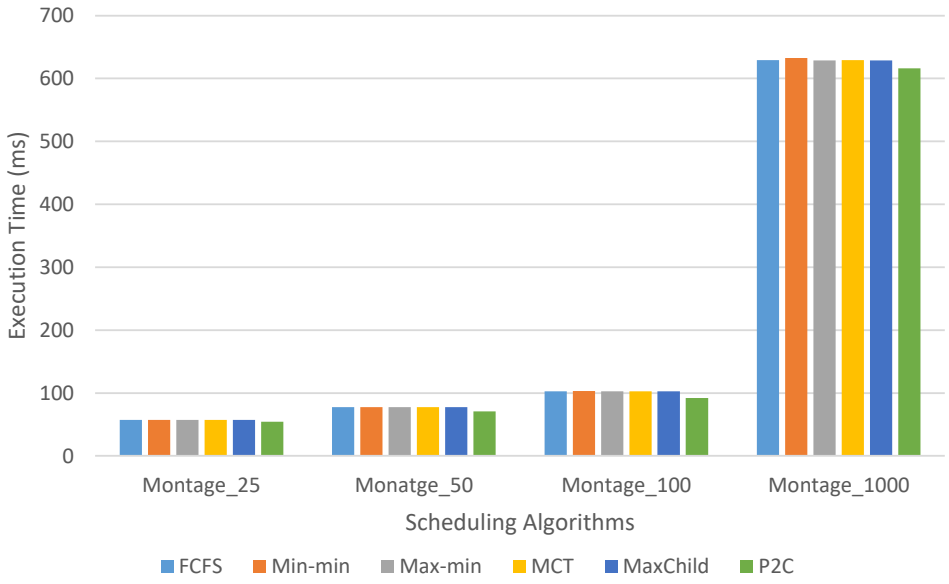


Figure 13. Comparison of execution time for montage workflow.

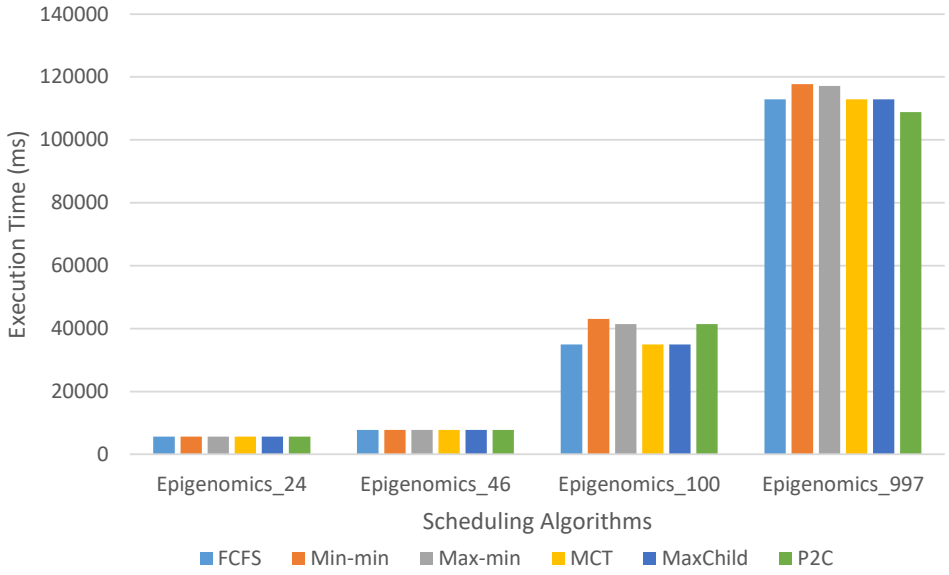


Figure 14. Comparison of execution time for epigenomics workflow.

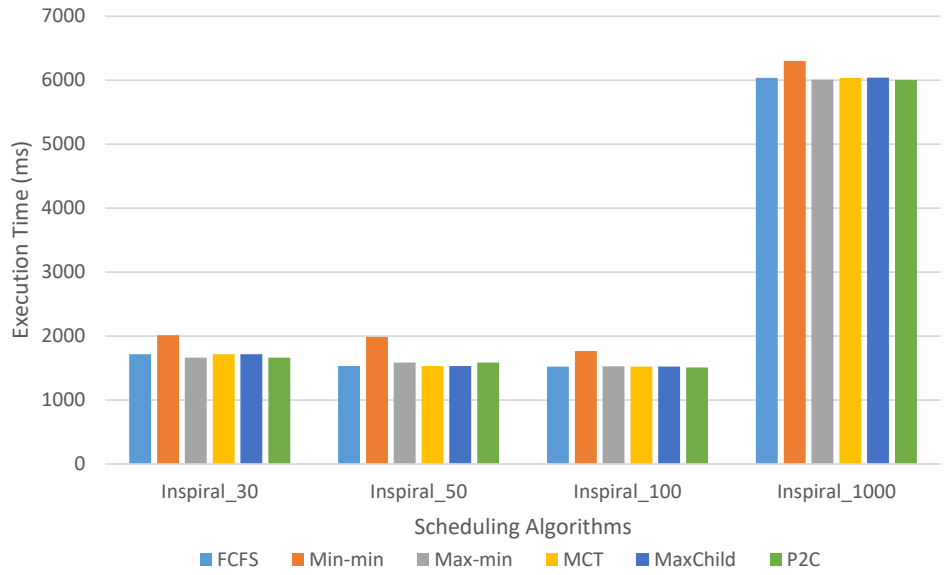


Figure 15. Comparison of execution time for inspiral workflow.

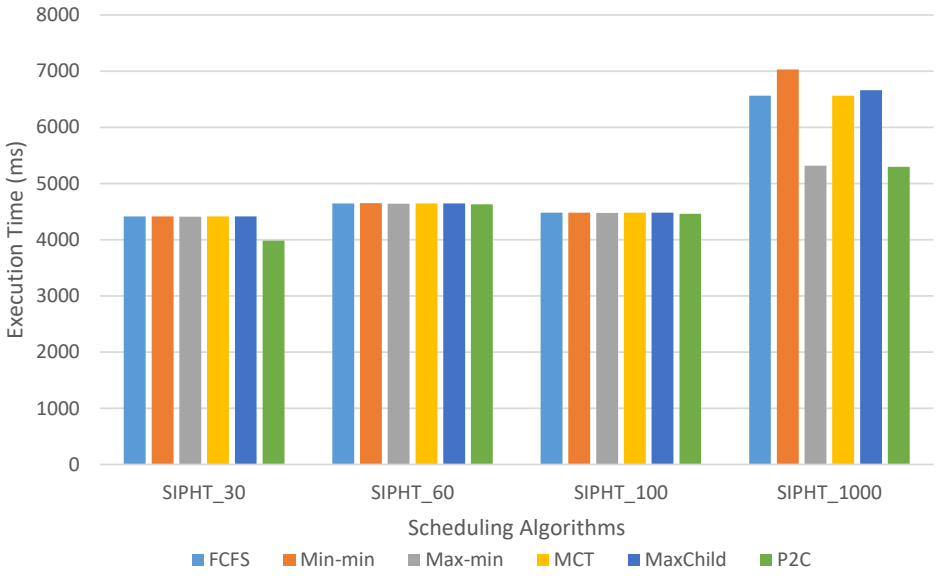


Figure 16. Comparison of execution time for SIPHT workflow.

It is clearly visible in Figure 12 that for CyberShake workflow, the proposed approach outperforms the existing approaches in terms of total execution time for less number of tasks. However, due to its parallel nature, in case of availability of more number of tasks, the proposed approach behaves similar to max–min scheduler and takes the same time as of max–min scheduler. Similarly, in some cases of Inspiral and Epigenomics workflow, the proposed algorithms behaves similar to max–min algorithms. It is clearly visible in Figure 13 for montage workflow and Figure 16 for SIPHT workflow, the proposed algorithm P2C outperforms the existing scheduling in terms of total execution time.

6.6. Complexity Analysis

The task scheduling problem in the cloud environment takes a large solution space. Moreover, the scheduling of n tasks onto m resources have been considered as an NP-Hard problem. Thus, it will take $O(n^m)$ time which is non-polynomial since, there is no existence of any algorithm that can find the optimal solution in a polynomial run time. The proposed

P2C algorithm is based on a priority queue structure. In general, a priority queue ranks their tasks by a particular key with an order relation. Here, each element has its key, and such keys are not necessarily unique. Some major operations have been associated with priority queue such as *TaskList*, *ReadyQueue*, *AvailableQueue*, $READYQUEUESTATUS()$, etc. The priority-based task scheduling consists of the following properties:

1. Each task has a priority associated with it;
2. A task with high priority must precedes the low priority tasks;
3. Two tasks can have same priority, however, such tasks will be scheduled as per their order in the queue.

Insertion time: Implementing priority queue without sorting would take $O(1)$ time. However, implementing priority queue with linear sorting would take $O(n)$ time.

Removal time: The functions like *TaskList*, *ReadyQueue*, *AvailableQueue* over unsorted priority queue would take $O(n)$ time. However, in case of linear sorted list it would take $O(n^2)$ time. Here, first *remove*() task takes $O(n)$, the second $O(n-1)$ and so on until the last removal task takes only $O(1)$ time. The total time needed to complete the first pass is: $O(n + (n-1) + (n-2) + ... + 2 + (1)) = O(\sum_{i=1}^{n} i) = \frac{n(n+1)}{2}$. Hence the total computational complexity for *removal*() is $O(n^2)$.

In case of the non-linear sorting, if priority queue is using heap then, *insert*() and *remove*() each take $O(logn)$, where n is the number of tasks. Here, each pass takes $O(nlogn)$ time for *insert*(), as well as *remove*().

7. Conclusions and Future Scope

Workflow scheduling is one of the significant issue in cloud computing among other popular issues, such as virtual machine migration, databases management, security, performance, fault tolerance, and server consolidation, etc. Workflow scheduling is a challenging job to find the proper sequence of workflow tasks for execution. It further depends upon the QoS requirements of the cloud applications. Due to heterogeneity; uncertainty and resource mobilization; resource scheduling is one of the hotspot area of research in demand. Different criteria for scheduling various resources and the parameters, requires different categories of resource scheduling techniques. In this paper, existing time-based scheduling algorithms such as first come first serve (FCFS), min–min, max–min, and minimum completion time (MCT), along with dependency-based scheduling algorithm MaxChild have been considered. The main objective of the existing time based schedulers is to reduce the workflow's execution time, but there is no importance given to resource utilization. The proposed (P2C) algorithm mainly focuses on utilization of the resources efficiently. Further, The proposed P2C algorithm outperforms existing time and dependency based scheduling algorithms in terms of total execution time. The experimental results conclude that the existing schedulers have varying execution time based on the size, shape, and number of resources and virtual machines available. From the statistical analysis, it has been analyzed that proposed algorithm has no significance of size, shape, and structure of the workflow. There are still many challenges that need to be overcome to achieve more effective and comprehensive results. Further, at present, the results have been computed for the standard CyberShake, montage, epigenomics, inspiral, and SIPHT workflow with varying number of tasks and virtual machines. However, in the future, the work could be extended to other scientific systems beyond the natural environment, and verification can be tested over real-time cloud space.

Author Contributions: The initial writing on the ideas; Preparation, creation and/or presentation of the published work by those from the original research group, specifically critical review, commentary or revision—including pre-or post-publication stages have been done by first author V.P. The other authors, S.B. and L.G. have played the role of supervisors with oversight and leadership responsibility for the research activity planning and execution, including mentorship external to the core team. Further, they have provided valuable suggestions specifically on the critical review, commentary

or revision—including pre-or post-publication stages. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

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

References

1. Harmon, P. *Business Process Change: A Business Process Management Guide For Managers and Process Professionals*; Morgan Kaufmann: San Francisco, CA, USA, 2019.
2. Li, J.; Li, X.; He, D. A directed acyclic graph network combined with CNN and LSTM for remaining useful life prediction. *IEEE Access* **2019**, *7*, 75464–75475. [CrossRef]
3. Deelman, E.; Vahi, K.; Rynge, M.; Mayani, R.; da Silva, R.F.; Papadimitriou, G.; Livny, M. The evolution of the pegasus workflow management software. *Comput. Sci. Eng.* **2019**, *21*, 22–36. [CrossRef]
4. de Carvalho Silva, J.; de Oliveira Dantas, A.B.; de Carvalho Junior, F.H. A Scientific Workflow Management System for orchestration of parallel components in a cloud of large-scale parallel processing services. *Sci. Comput. Program.* **2019**, *173*, 95–127. [CrossRef]
5. Pandey, S.; Vahi, K.; da Silva, R.F.; Deelman, E.; Jiang, M.; Harrison, C.; Chu, A.; Casanova, H. Event-Based Triggering and Management of Scientific Workflow Ensembles. In Proceedings of the HPC Asia, Tokyo, Japan, 28–31 January 2018.
6. Senkul, P.; Toroslu, I.H. An architecture for workflow scheduling under resource allocation constraints. *Inf. Syst.* **2005**, *30*, 399–422. [CrossRef]
7. Yu, J.; Buyya, R. A taxonomy of workflow management systems for grid computing. *J. Grid Comput.* **2005**, *3*, 171–200. [CrossRef]
8. Ma, X.; Gao, H.; Xu, H.; Bian, M. An IoT-based task scheduling optimization scheme considering the deadline and cost-aware scientific workflow for cloud computing. *EURASIP J. Wirel. Commun. Netw.* **2019**, *2019*, 249. [CrossRef]
9. 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]
10. Khennaoui, R.; Belala, N. Towards a Formal Context-Aware Workflow Model for Ambient Environment. In Proceedings of the International Conference on Smart Homes and Health Telematics, Hammamet, Tunisia, 24–26 June 2020; Springer: Berlin/Heidelberg, Germany, 2020; pp. 415–422.
11. Talwani, S.; Singla, J. Comparison of Various Fault Tolerance Techniques for Scientific Workflows in Cloud Computing. In Proceedings of the 2019 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COMITCon), Faridabad, India, 14–16 February 2019; IEEE: Piscataway, NJ, USA, 2019; pp. 454–459.
12. Deelman, E.; Mandal, A.; Jiang, M.; Sakellariou, R. The role of machine learning in scientific workflows. *Int. J. High Perform. Comput. Appl.* **2019**, *33*, 1128–1139. [CrossRef]
13. Deelman, E.; Vahi, K.; Juve, G.; Rynge, M.; Callaghan, S.; Maechling, P.J.; Mayani, R.; Chen, W.; Da Silva, R.F.; Livny, M.; et al. Pegasus, a workflow management system for science automation. *Future Gener. Comput. Syst.* **2015**, *46*, 17–35. [CrossRef]
14. Singh, S.; Chana, I. A survey on resource scheduling in cloud computing: Issues and challenges. *J. Grid Comput.* **2016**, *14*, 217–264. [CrossRef]
15. Kijak, J.; Martyna, P.; Pawlik, M.; Balis, B.; Malawski, M. Challenges for scheduling scientific workflows on cloud functions. In Proceedings of the 2018 IEEE 11th International Conference on Cloud Computing (CLOUD), San Francisco, CA, USA, 2–7 July 2018; IEEE: Piscataway, NJ, USA, 2018; pp. 460–467.
16. George, S.S.; Pramila, R.S. A review of different techniques in cloud computing. In *Materials Today: Proceedings*; Elsevier: Amsterdam, The Netherlands, 2021.
17. Zhou, J.; Wang, T.; Cong, P.; Lu, P.; Wei, T.; Chen, M. Cost and makespan-aware workflow scheduling in hybrid clouds. *J. Syst. Archit.* **2019**, *100*, 101631. [CrossRef]
18. Barrett, E.; Howley, E.; Duggan, J. A learning architecture for scheduling workflow applications in the cloud. In Proceedings of the 2011 IEEE Ninth European Conference on Web Services, Lugano, Switzerland, 14–16 September 2011; IEEE: Piscataway, NJ, USA, 2011; pp. 83–90.
19. Isaac, E.U.; Izuchukwu, A.C. Development of a Model Architecture for Job Scheduling. *Sci. J. Circuits Syst. Signal Process.* **2020**, *9*, 16. [CrossRef]
20. Yu, J.; Buyya, R. A taxonomy of scientific workflow systems for grid computing. *ACM Sigmod Rec.* **2005**, *34*, 44–49. [CrossRef]
21. Kintsakis, A.M.; Psomopoulos, F.E.; Mitkas, P.A. Reinforcement learning based scheduling in a workflow management system. *Eng. Appl. Artif. Intell.* **2019**, *81*, 94–106. [CrossRef]
22. Varalakshmi, P.; Ramaswamy, A.; Balasubramanian, A.; Vijaykumar, P. An optimal workflow based scheduling and resource allocation in cloud. In Proceedings of the International Conference on Advances in Computing and Communications, Kochi, India, 22–24 July 2011; Springer: Berlin/Heidelberg, Germany, 2011; pp. 411–420.
23. Zhou, X.; Zhang, G.; Sun, J.; Zhou, J.; Wei, T.; Hu, S. Minimizing cost and makespan for workflow scheduling in cloud using fuzzy dominance sort based HEFT. *Future Gener. Comput. Syst.* **2019**, *93*, 278–289. [CrossRef]
24. Prakash, V.; Bala, A.G. An Efficient Workflow Scheduling Approach in Cloud Computing. Ph.D. Thesis, Thapar Institute of Engineering and Technology, Patiala, India, 2014.

25. 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]
26. Tong, Z.; Chen, H.; Deng, X.; Li, K.; Li, K. A scheduling scheme in the cloud computing environment using deep Q-learning. *Inf. Sci.* **2020**, *512*, 1170–1191. [CrossRef]
27. Mansouri, N.; Javidi, M.M. Cost-based job scheduling strategy in cloud computing environments. In *Distributed and Parallel Databases*; Springer: Berlin/Heidelberg, Germany, 2019; pp. 1–36.
28. Kumar, A.; Bawa, S. DAIS: Dynamic access and integration services framework for cloud-oriented storage systems. In *Cluster Computing*; Springer: Berlin/Heidelberg, Germany, 2020; pp. 1–20.
29. Buyya, R.; Yeo, C.S.; Venugopal, S.; Broberg, J.; Brandic, I. Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility. *Future Gener. Comput. Syst.* **2009**, *25*, 599–616. [CrossRef]
30. Hu, J.; Gu, J.; Sun, G.; Zhao, T. A scheduling strategy on load balancing of virtual machine resources in cloud computing environment. In Proceedings of the 2010 3rd International Symposium on Parallel Architectures, Algorithms and Programming, Dalian, China, 18–20 December 2010; IEEE: Piscataway, NJ, USA, 2010; pp. 89–96.
31. Prakash, V.; Bala, A. Workflow Scheduling Algorithms in Grid and Cloud Environment—A Survey. *J. Comput. Technol.* **2014**, *3*, 2278–3814.
32. Njenga, K.; Garg, L.; Bhardwaj, A.K.; Prakash, V.; Bawa, S. The cloud computing adoption in higher learning institutions in Kenya: Hindering factors and recommendations for the way forward. *Telemat. Inform.* **2019**, *38*, 225–246. [CrossRef]
33. Yu, Y.; Su, Y. Cloud Task Scheduling Algorithm Based on Three Queues and Dynamic Priority. In Proceedings of the 2019 IEEE International Conference on Power, Intelligent Computing and Systems (ICPICS), Shenyang, China, 12–14 July 2019; IEEE: Piscataway, NJ, USA, 2019; pp. 278–282.
34. Xia, W.; Shen, L. Joint resource allocation using evolutionary algorithms in heterogeneous mobile cloud computing networks. *China Commun.* **2018**, *15*, 189–204. [CrossRef]
35. Patra, M.K.; Sahoo, S.; Sahoo, B.; Turuk, A.K. Game theoretic approach for real-time task scheduling in cloud computing environment. In Proceedings of the 2019 International Conference on Information Technology (ICIT), Bhubaneswar, India, 19–21 December 2019; IEEE: Piscataway, NJ, USA, 2019; pp. 454–459.
36. Li, J.; Ma, T.; Tang, M.; Shen, W.; Jin, Y. Improved FIFO scheduling algorithm based on fuzzy clustering in cloud computing. *Information* **2017**, *8*, 25. [CrossRef]
37. Nazar, T.; Javaid, N.; Waheed, M.; Fatima, A.; Bano, H.; Ahmed, N. Modified shortest job first for load balancing in cloud-fog computing. In Proceedings of the International Conference on Broadband and Wireless Computing, Communication and Applications, Taichung, Taiwan, 15 July–15 August 2018; Springer: Berlin/Heidelberg, Germany, 2018; pp. 63–76.
38. Devi, D.C.; Uthariaraj, V.R. Load balancing in cloud computing environment using improved weighted round robin algorithm for nonpreemptive dependent tasks. *Sci. World J.* **2016**, *2016*, 3896065. [CrossRef] [PubMed]
39. Mazumder, A.M.R.; Uddin, K.A.; Arbe, N.; Jahan, L.; Whaiduzzaman, M. Dynamic task scheduling algorithms in cloud computing. In Proceedings of the 2019 3rd International Conference on Electronics, Communication and Aerospace Technology (ICECA), Coimbatore, India, 12–14 June 2019; IEEE: Piscataway, NJ, USA, 2019.
40. Ghosh, S.; Banerjee, C. Dynamic time quantum priority based round robin for load balancing in cloud environment. In Proceedings of the 2018 Fourth International Conference on Research in Computational Intelligence and Communication Networks (ICRCICN), Kolkata, India, 22–23 November 2018; IEEE: Piscataway, NJ, USA, 2018; pp. 33–37.
41. Samadi, Y.; Zbakh, M.; Tadonki, C. E-HEFT: Enhancement heterogeneous earliest finish time algorithm for task scheduling based on load balancing in cloud computing. In Proceedings of the 2018 International Conference on High Performance Computing & Simulation (HPCS), Orleans, France, 16–20 July 2018; IEEE: Piscataway, NJ, USA, 2018; pp. 601–609.
42. Gao, Y.; Zhang, S.; Zhou, J. A hybrid algorithm for multi-objective scientific workflow scheduling in IaaS Cloud. *IEEE Access* **2019**, *7*, 125783–125795. [CrossRef]
43. Bugingo, E.; Zheng, W.; Zhang, D.; Qin, Y.; Zhang, D. Decomposition based multi-objective workflow scheduling for cloud environments. In Proceedings of the 2019 Seventh International Conference on Advanced Cloud and Big Data (CBD), Suzhou, China, 21–22 September 2019; IEEE: Piscataway, NJ, USA, 2019; pp. 37–42.
44. Wu, N.; Zuo, D.; Zhang, Z. Dynamic fault-tolerant workflow scheduling with hybrid spatial-temporal re-execution in clouds. *Information* **2019**, *10*, 169. [CrossRef]
45. Arabnejad, V.; Bubendorfer, K.; Ng, B. Budget and deadline aware e-science workflow scheduling in clouds. *IEEE Trans. Parallel Distrib. Syst.* **2018**, *30*, 29–44. [CrossRef]
46. Kaur, G.; Bala, A. An efficient resource prediction–based scheduling technique for scientific applications in cloud environment. *Concurrent Eng.* **2019**, *27*, 112–125. [CrossRef]
47. Kaur, G.; Bala, A. Prediction based task scheduling approach for floodplain application in cloud environment. In *Computing*; Springer: Berlin/Heidelberg, Germany, 2021; pp. 1–22.
48. Niehaus, D.; Ramamritham, K.; Stankovic, J.A.; Wallace, G.; Weems, C.; Burleson, W.; Ko, J. The Spring scheduling co-processor: Design, use, and performance. In Proceedings of the 1993 Proceedings Real-Time Systems Symposium, Raleigh, NC, USA, 1–3 December 1993; IEEE: Piscataway, NJ, USA, 1993; pp. 106–111.
49. Burleson, W.; Ko, J.; Niehaus, D.; Ramamritham, K.; Stankovic, J.A.; Wallace, G.; Weems, C. The spring scheduling coprocessor: A scheduling accelerator. *IEEE Trans. Very Large Scale Integr. Syst.* **1999**, *7*, 38–47. [CrossRef]

50. Trakadas, P.; Nomikos, N.; Michailidis, E.T.; Zahariadis, T.; Facca, F.M.; Breitgand, D.; Rizou, S.; Masip, X.; Gkonis, P. Hybrid clouds for data-intensive, 5G-enabled IoT applications: An overview, key issues and relevant architecture. *Sensors* **2019**, *19*, 3591. [CrossRef] [PubMed]
51. Graves, R.; Jordan, T.H.; Callaghan, S.; Deelman, E.; Field, E.; Juve, G.; Kesselman, C.; Maechling, P.; Mehta, G.; Milner, K.; et al. CyberShake: A physics-based seismic hazard model for southern California. *Pure Appl. Geophys.* **2011**, *168*, 367–381. [CrossRef]
52. Bharathi, S.; Chervenak, A.; Deelman, E.; Mehta, G.; Su, M.H.; Vahi, K. Characterization of scientific workflows. In Proceedings of the 2008 Third Workshop on Workflows in Support of Large-Scale Science, Austin, TX, USA, 17 November 2008; IEEE: Piscataway, NJ, USA, 2008; pp. 1–10.
53. Juve, G.; Chervenak, A.; Deelman, E.; Bharathi, S.; Mehta, G.; Vahi, K. Characterizing and profiling scientific workflows. *Future Gener. Comput. Syst.* **2013**, *29*, 682–692. [CrossRef]
54. Braun, T.D.; Siegel, H.J.; Beck, N.; Bölöni, L.L.; Maheswaran, M.; Reuther, A.I.; Robertson, J.P.; Theys, M.D.; Yao, B.; Hensgen, D.; et al. A comparison of eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed computing systems. *J. Parallel Distrib. Comput.* **2001**, *61*, 810–837. [CrossRef]
55. Dong, F.; Akl, S.G. *Scheduling Algorithms for Grid Computing: State of the Art and Open Problems*; Technical Report; Queen's University: Kingston, ON, Canada, 2006.
56. Prakash, V.; Bala, A. A novel scheduling approach for workflow management in cloud computing. In Proceedings of the 2014 International Conference on Signal Propagation and Computer Technology (ICSPCT 2014), Ajmer, India, 12–13 July 2014; IEEE: Piscataway, NJ, USA, 2014; pp. 610–615.
57. Di Cosmo, R.; Lienhardt, M.; Treinen, R.; Zacchiroli, S.; Zwolakowski, J.; Eiche, A.; Agahi, A. Automated synthesis and deployment of cloud applications. In Proceedings of the 29th ACM/IEEE International Conference On Automated Software Engineering, Vasteras, Sweden, 15–19 September 2014; pp. 211–222.
58. Jennings, B.; Stadler, R. Resource management in clouds: Survey and research challenges. *J. Netw. Syst. Manag.* **2015**, *23*, 567–619. [CrossRef]
59. Parikh, S.M.; Patel, N.M.; Prajapati, H.B. Resource management in cloud computing: Classification and taxonomy. *arXiv* **2017**, arXiv:1703.00374.
60. 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; IEEE: Piscataway, NJ, USA, 2012; pp. 1–8.
61. Calheiros, R.N.; Ranjan, R.; Beloglazov, A.; De Rose, C.A.; Buyya, R. CloudSim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Software Pract. Exp.* **2011**, *41*, 23–50. [CrossRef]
62. Glen, S. Statistical Analysis. Available online: https://www.statisticshowto.com/statistical-analysis/ (accessed on 15 April 2021).
63. Zaiontz, C. Two Factor ANOVA without Replication. Available online: https://www.real-statistics.com/two-way-anova/two-factor-anova-without-replication/ (accessed on 15 April 2021).