Hindawi
Wireless Communications and Mobile Computing
Volume 2022, Article ID 1637614, 12 pages
https://doi.org/10.1155/2022/1637614

Research Article

Energy-Efficient Scientific Workflow Scheduling Algorithm in Cloud Environment

Neha Garg[1], **Neeraj**[1], **Manish Raj**[1], **Indrajeet Gupta**[1], **Vinay Kumar**[2] and **G. R. Sinha**[3]

[1]*School of CSET, Bennett University, Greater Noida, India*
[2]*Department of Computer Engineering and Applications, GLA University, Mathura, India*
[3]*Myanmar Institute of Information Technology, Mandalay, Myanmar*

Correspondence should be addressed to G. R. Sinha; gr_sinha@miit.edu.mm

Received 4 February 2022; Accepted 21 February 2022; Published 14 March 2022

Academic Editor: Deepak Kumar Jain

Copyright © 2022 Neha Garg et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Scheduling extensive scientific applications that are deadline-aware (usually referred to as workflow) is a difficult task. This research provides a virtual machine (VM) placement and scheduling approach for effectively scheduling process tasks in the cloud environment while maintaining dependency and deadline constraints. The suggested model's aim is to reduce the application's energy consumption and total execution time while taking into account dependency and deadline limitations. To select the VM for the tasks and dynamically deploy/undeploy the VM on the hosts based on the jobs' requirements, an energy-efficient VM placement (EVMP) algorithm is presented. Demonstrate that the proposed approach outperforms the existing PESVMC (power-efficient scheduling and VM consolidation) algorithm.

1. Introduction

Large-scale complex scientific applications/workflow are executed and analyzed in the multi-disciplinary area of research such as astronomy and physics [1]. The workflow contains a large number of mutually dependent tasks which are executed according to their dependency constraint [2]. Due to dependency constraint, the child task can start its execution only when parent task finishes its execution. A directed acyclic graph (DAG) is used to represent workflows. These workflows often have disparate requirements (such as storage and CPU) and constraints (dependency) that need to be accounted during their execution. For example, the scientific workflow Panoramic Survey Telescope and Rapid Response System (Pan-STARRS) [3] is a resource-intensive workflow with a good degree of scalability [4]. The strict necessity of the computing infrastructure makes the execution of scientific applications difficult and costly [5]. Cloud computing provides virtualized cloud resources as a service, on-demand, and pay-per-use basis [6, 7]. The characteristics of cloud computing such as elasticity and flexibility make this environment a major trend for computation and storage services. These characteristics motivate to execution of scientific applications in the cloud environment [8].

Scientific workflows are the constitution of distinct tasks with complex dependency. Resource provisioning and the order in which workflow tasks are executed are challenging problems. The inefficient utilization of resources while executing the workflows wastes a tremendous number of resources. The inefficient utilization of resources increases the number of unused provisioned resources. These unused resources increase energy consumption without performing any useful operation [9]. The resource utilization can be increased by efficient resource provisioning. An energy-efficient scheduling algorithm can be used to manage the resources that are required by the task while executing these scientific workflow tasks. In the literature, numerous workflow scheduling algorithms have been proposed. These scheduling algorithms focus on diminishing makespan and cost with inadequate resources. The selection and designing

of a competent and operative workflow scheduling algorithm are also challenging tasks [10]. The energy-aware scheduling algorithm must be selected which can provision a proper resource from the offered resources which are efficient enough to complete the workflow tasks within their deadline constriction, and it can decrease the energy consumption. To minimize the energy consumption Dynamic Power Management (DPM) [11, 12], Dynamic Voltage and Frequency Scaling (DVFS) [12–15], resource consolidation with migration techniques [6, 16], virtualization [6], and green policies [17], technologies are used. Energy consumption has also been minimized by reducing the computational power of the resources. The reduction of computational power has increased the workflow makespan.

An amalgamation of software and hardware-based techniques is necessary to reduce energy consumption. In this paper, the EVMP algorithm is proposed to schedule a scientific workflow on virtual machines (VMs). The EVMP algorithm integrates both hardware and software policies to minimize energy consumption. Virtualization technology is exploited to create VMs on a server. DVFS technique is used to save energy when the server/core of the CPU is idle. Dynamic provisioning of heterogeneous types of available resources is considered to show the infrastructure-as-a-Service (IaaS) cloud service. An energy model is presented to monitor and calculate the energy consumption. During the scheduling of tasks, server overloading is also prevented by monitoring the server status [18].

1.1. Paper Outline. The next section presents the cloud workflow model, task model, and energy model to execute the workflow. In Section 3, the workflow task scheduling algorithm is presented. The experimentation and performance metrics are presented in Section 4. Section 5 demonstrates the simulation results and discussion. The conclusion of the paper along with future directions is presented in the last section.

2. System Model

This section describes the cloud model, workflow model, task model, and energy model.

2.1. Cloud Model. In this research paper, large and nonhomogenous hosts or physical servers are deployed. In this paper, host/physical server host_k (k^{th} host) is depicted as $\text{host}_k = \{\text{cpu}_k, \text{pre}_k, \text{ram}_k, \text{network}_k, \text{storage}_k\}$, where cpu_k, pre_k, ram_k, network_k, storage_k is Central Processing Unit (CPU) capacity, number of processing elements, Random Access Memory (RAM) capacity, network bandwidth capacity, and storage capacity on the host_k, respectively. cpu_k is equally divided into pre_k. Million instructions per second (MIPS) [6], megabytes (MB), gigabits per second (Gbps), and gigabyte (GB) measurement units are used to measure the capacity of CPU, RAM, network bandwidth, and storage, respectively [19]. VMs are used to execute the workflow, and more than one VM can be deployed on the host. Let j be the number of VMs deployed on the host_k, and it is depicted as $\text{VM}_{jk} = \{\text{vm}_{1k}, \text{vm}_{2k}, \cdots, \text{vm}_{jk}\}$. VMs are dynamically deployed/undeployed on the host as per the workflow demands. To execute the workflow, the fraction of host resources are allocated to the VM, and it is depicted as $\text{vm}_{jk} = \{f\text{cpu}_{jk}, f\text{pre}_{jk}, f\text{ram}_{jk}, f\text{network}_{jk}, f\text{storage}_{jk}\}$, where $\text{vm}_{jk}, f\text{cpu}_{jk}, f\text{pre}_{jk}, f\text{ram}_{jk}, f\text{network}_{jk}$, and $f\text{storage}_{jk}$ are j^{th} VM on k^{th} host, fraction of CPU, processing elements, RAM, network bandwidth, and storage, respectively. In this paper, hosts are switched on/off dynamically. Based on the host utilization, hosts are characterized into three categories, i.e., underloaded, overloaded, and normal. If the resource utilization is less than the lower threshold value, then the host is categorized as an underloaded host. If any host is underloaded, then try to migrate the deployed VMs and switch off the host. This strategy is useful to minimize energy consumption. If the resource utilization is more than the upper threshold value, then, the host is categorized as an overloaded host. Migrate some of the VMs from the overloaded host because overloaded hosts consume more energy. Otherwise, the host is in the normal category.

2.2. Workflow Model. Workflow (W) is described as a set of interdependent computational tasks [20]. In the literature [21], many scientific workflows such as LIGO, Montage, Cybershake, Epigenomics, and Pan-STARRS exist. In this paper, Pan-STARRS scientific workflow is considered for task execution. Pan-STARRS project continuously monitors the entire sky to detect moving or variable objects. PS1 telescope is used to monitor the sky. John Hopkins University and Microsoft manage the generated astronomy data using two types of workflows, i.e., PSLoad workflow and PSMerge workflow. PSLoad workflow is used to collect the data from the telescope and store data in the database. PSMerge workflow is used to update the database. PSLoad and PSMerge workflows are described in Figures 1 and 2, respectively. Table 1 describes the detailed characteristics of the workflows.

2.3. Task Model. A workflow task is an activity that is carried out as part of the workflow description [20]. Workflow task has needed resources for the complete execution of the workflow with a set of constraints. For example, task length/size in million instruction (MI), number of processing elements, deadline in seconds, data transfer file size in MB, list of child tasks, and list of parent tasks, these are modeled as $t_i = \{l_i, \text{pe}_i, d_i, \text{tr}_i, c_i, p_i\}$, respectively. Based on the task resource requirement and constraints (such as deadline, length, and dependency), VMs are dynamically deployed. The execution time, transfer time, start time, finish time, and makespan time are defined as

(1) Execution time: the task's execution time is measured in seconds, and it is determined by the task's length and the processing capacity of the VM that is used to execute it. The execution time (et_{ijk}) of the task t_i on vm_{jk} at host_k is calculated as

$$\text{et}_{ijk} = \frac{l_i}{f\text{cpu}_{jk}.f\text{pre}_{jk}}. \quad (1)$$

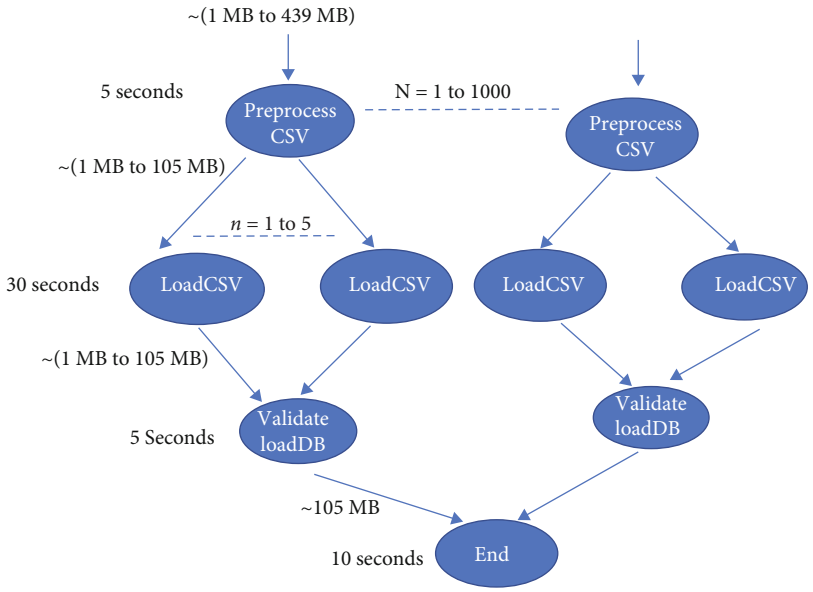

Figure 1: PSLoad workflow.

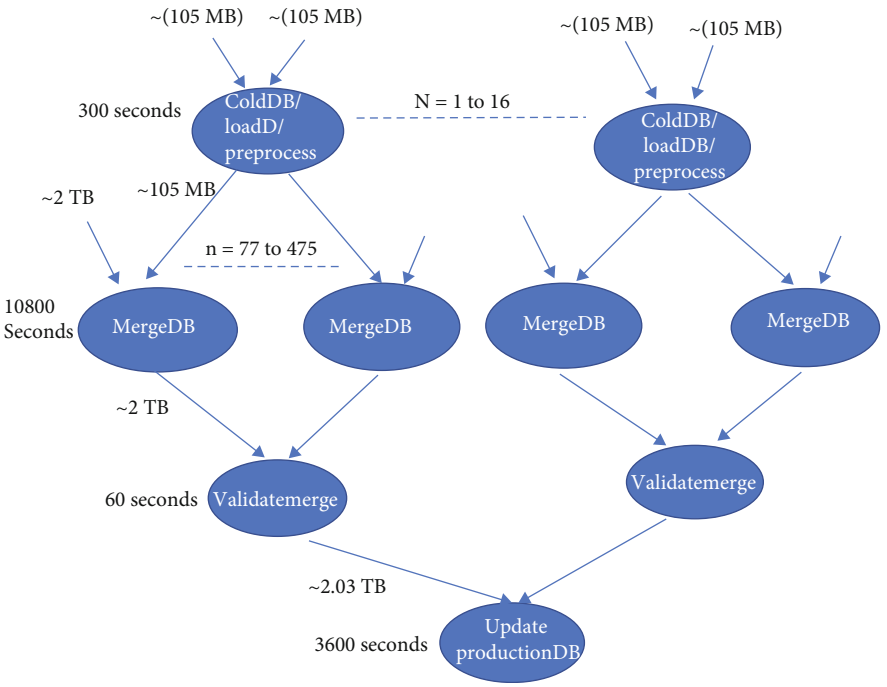

Figure 2: PSMerge workflow.

(2) Transfer time: if the child and parent tasks are not executed on the same VM, then output data (i.e., tr_i) is transferred from the parent task to the child task for its execution. Let t_p, t_c, vm_p,vm_c, and cst be the parent task, child task, VM which is deployed for a parent task, VM which is deployed about to execute the child task, and communication startup time, respectively. The transfer time (tt_p^c) from task t_p to t_c is calculated as

$$\mathrm{tt}_p^c = \begin{cases} \mathrm{cst} + \dfrac{\mathrm{tr}_i}{\text{Bandwidth between } \mathrm{vm}_p \text{ and } \mathrm{vm}_c}, & \text{if } (\mathrm{vm}_p \neq \mathrm{vm}_c), \\ 0, \quad \text{otherwise.} \end{cases} \tag{2}$$

Table 1: Workflow characterization.

Workflow categories	Total number of tasks	Total number of child tasks	Total number of edges	Type of tasks	Number of tasks	Length of tasks (in MI)	Deadline (in seconds)	Input file size (in MB)
PSLoad_ Small	4	3	4	PreprocessorCSV	1	2500-12500	5	97.52
				LoadCSV	1	15000-75000	30	97.52
				ValidateLoadDB	1	2500-12500	5	97.52
				End	1	5000-25000	10	104.86
PSLoad_ Medium	489	389	776	PreprocessorCSV	100	2500-12500	5	6.29-362.81
				LoadCSV	288	15000-75000	30	1.05-104.86
				ValidateLoadDB	100	2500-12500	5	1.05-104.86
				End	1	5000-25000	10	104.86
PSLoad_ Large	5084	4084	8166	PreprocessorCSV	1000	2500-12500	5	1.05-438.3
				LoadCSV	3083	15000-75000	30	1.05-104.86
				ValidateLoadDB	1000	2500-12500	5	1.05-104.86
				End	1	5000-25000	10	104.86
PSMerge_ Small	80	79	234	ColdDB/LoadDB/ Preprocess	1	150000-750000	300	104.86 and 104.86
				MergeDB	77	540000-27000000	10800	104.86 and 2199023.256
				ValidateMerge	1	30000-150000	60	2199023.256
				UpdateProductionDB	1	1800000-9000000	3600	2232008.604
PSMerge_ Medium	841	836	2505	ColdDB/LoadDB/ Preprocess	5	150000-750000	300	104.86 and 104.86
				MergeDB	830	540000-27000000	10800	104.86 and 2199023.256
				ValidateMerge	5	30000-150000	60	2199023.256
				UpdateProductionDB	1	1800000-9000000	3600	2232008.604
PSMerge_ Large	7622	7606	22815	ColdDB/LoadDB/ Preprocess	16	150000-750000	300	104.86 and 104.86
				MergeDB	7589	540000-27000000	10800	104.86 and 2199023.256
				ValidateMerge	16	30000-150000	60	2199023.256
				UpdateProductionDB	1	1800000-9000000	3600	2232008.604

(3) Start time: $t_i$be the entry task; then start time(st_{ijk})of a task$t_i$onvm_{jk}at$host_k$is calculated as

$$st_{ijk} = rt_{jk}, \tag{3}$$

where rt_{jk} is the ready time of the vm_{jk} at $host_k$.

Task is not an entry task, and the same VM is used to execute the child task and its parent task; then, the start time of the child task is calculated as

$$st_{ijk} = \max\left(rt_{jk}, ft_{pjk}\right), \tag{4}$$

where ft_{pjk} is the finish time of the parent task on vm_{jk} at $host_k$.

If task t_i is not an entry task and is allocated on the different VM on which its parent is not executed, then, the start time of the task is calculated as

$$st_{ijk} = \max\left(rt_{jk}, ft_{pjk}\right) + tt_p^i. \tag{5}$$

If task t_i is not an entry task and a new VM (vm_{jk}) is deployed for its execution, then, st_{ijk} is calculated as

$$\mathrm{st}_{ijk} = \mathrm{ft}_{pxk} + \mathrm{tt}_p^i + \mathrm{ct}_{jk}, \tag{6}$$

where ct_{jk} is the creation time of the vm_{jk} at host_k. If the VM v_{xk} is migrated to a new host and a new VM v_{jk} is positioned on the host, then the start time of the task is estimated as

$$\mathrm{st}_{ijk} = \mathrm{ct}_{jk} + \mathrm{ft}_{pxk} + \mathrm{tt}_p^i + \sum_{x=1}^{|x|} \mathrm{mt}(v_{xk}), \tag{7}$$

where $\mathrm{mt}(v_{xk})$ is the migration time of the vm_{xk}. If a new host is activated, then the start time of the task is evaluated as

$$\mathrm{st}_{ijk} = \mathrm{st}(\mathrm{host}_k) + \mathrm{ct}_{jk} + \mathrm{ft}_{pxk} + \mathrm{tt}_p^c, \tag{8}$$

where $\mathrm{st}(\mathrm{host}_k)$ is the start time of the host_k.

(4) Finish time: finish time (ft_{ijk}) of the task t_i on v_{jk} at host_k is calculated as

$$\mathrm{ft}_{ijk} = \mathrm{st}_{ijk} + \mathrm{et}_{ijk}. \tag{9}$$

(5) Makespan time: workflow makespan (w_{makespan}) is the total time that is taken to complete the execution of the workflow and is calculated as

$$w_{\mathrm{makespan}} = \max\left(\mathrm{ft}_{ijk}\right) - \mathrm{subTime}_{\mathrm{workflow}}, \tag{10}$$

where $\mathrm{subTime}_{\mathrm{workflow}}$ is the submission time of the workflow.

2.4. Energy Model. CPUs, network interfaces, memory, and storage devices are the most energy-intensive components of host servers. The CPU consumes approximately 37% to 43% of total server energy [22, 23], and network devices consume approximately 33% of total data center energy [24]. In the proposed work, the energy consumption of the CPU [6] and data transfer between VMs [24] are taken into account, and the total energy consumption is calculated in five different scenarios. These scenarios are defined as follows.

Scenario 1. This scenario is used to calculate the energy consumption during the execution of the task t_i on vm_{jk} on host_k (i.e., ec_{ijk}) and is calculated as

$$\mathrm{ec}_{ijk} = \mathrm{ecr}_{jk} \bullet \mathrm{et}_{ijk}, \tag{11}$$

where ecr_{jk} is the energy consumption rate of the vm_{jk} on host_k. Energy consumption to execute the whole workflow is

$$\mathrm{ec}^{\mathrm{exec}} = \sum_{k=1}^{|\mathrm{Host}_a|} \sum_{j=1}^{|\mathrm{VM}_k|} \sum_{i=1}^{T} x_{ijk} \bullet \mathrm{ecr}_{ijk} \bullet \mathrm{et}_{ijk}, \tag{12}$$

where x_{ijk} symbolizes the mapping of task t_i on vm_j at host h_k. The x_{ijk} remains "1" if the task t_i is scheduled on VM v_j at host_k for execution; otherwise, x_{ijk} is equal to "0."

Scenario 2. This scenario is used when the server/host is active, but no VM is running on it; this situation is used to reduce energy consumption by switching the host to low voltage and frequency over some time (up to a threshold duration). Energy consumption of the idle hosts (i.e., $\mathrm{ec}^{\mathrm{all\ Idle}}$) is calculated as

$$\mathrm{ec}^{\mathrm{all\ Idle}} = \sum_{k=1}^{|\mathrm{Host}_a|} \mathrm{ecr}'_k \bullet \mathrm{it}_k, \tag{13}$$

where ecr'_k and it_k is energy consumption rate of host_k at idle mode and idle time of host_k.

Scenario 3. This scenario is used when the server/host is partially idle such as some idle VM is installed on the host. The VM is left idle up to the threshold period. The energy consumption of the partially idle host (i.e., $\mathrm{ec}^{\mathrm{part\ Idle}}$) is calculated as

$$\begin{aligned} \mathrm{ec}^{\mathrm{part\ Idle}} &= \sum_{k=1}^{|\mathrm{Host}_a|} \sum_{j=1}^{|\mathrm{VM}_k|} \mathrm{ecr}_{jk}.t_j^{\mathrm{part\ Idle}}, \\ t_j^{\mathrm{part\ Idle}} &= \mathrm{dt}_{jk} - \mathrm{ct}_{jk} - \sum_{i=1}^{|T|} x_{ijk} \bullet \mathrm{et}_{ijk}, \end{aligned} \tag{14}$$

where $t_j^{\mathrm{part\ Idle}}$, dt_{jk}, and ct_{jk} are the idle time of vm_j at host_k, time at which vm_j is un-deployed from host_k, and time at which vm_j is deployed at host_k, respectively.

Scenario 4. This scenario is used to calculate the energy of unused resources of the servers/hosts. Energy consumption is minimized by applying core-level DVFS. It is evident from the paper [25] about 50% energy usage is minimized by reducing the voltage at 70% from its peak voltage. Minimum time is taken during scaling in which the operating frequency of the resources is in nanoseconds [6]. Therefore, during the calculations, scaling time of frequency is neglected. Energy consumption of unused resources of the hosts (i.e., ecur) is calculated as

$$\mathrm{ecur} = \sum_{k=1}^{|\mathrm{Host}_a|} \sum_{j=1}^{|\mathrm{VM}_k|} \sum_{p=1}^{s} \mathrm{ecr}'_{jk} \bullet t_p, \tag{15}$$

where s is the time in which the reckoning of VMs in a host is distinct from the former time.

Total computational energy consumption ($\mathrm{ec}^{\mathrm{computational}}$) is the addition of the above four scenarios as shown in Equation (16).

$$\mathrm{ec}^{\mathrm{computational}} = \mathrm{ec}^{\mathrm{exec}} + \mathrm{ec}^{\mathrm{allIdle}} + \mathrm{ec}^{\mathrm{partIdle}} + \mathrm{ecur}. \tag{16}$$

Scenario 5. This scenario is used to calculate the energy

consumption during the data transfer from one VM to another VM when parent and child tasks are not executed on the same VM. The energy consumption to transfer data ($ec^{transfer}$) is calculated as

$$ec^{transfer} = \begin{cases} ecrBW_{xj} * tt_p^c, & \text{if } (vm_x \neq v_{mj}), \\ 0, & \text{otherwise,} \end{cases} \quad (17)$$

where $ecrBW_{xj}$ and tt_p^c are energy consumption rate of network bandwidth and transfer time of data from one VM to another VM, respectively. Total energy consumption of a data center during workflow execution is calculated by using Equations (16) and (17) as

$$ec^{total} = ec^{computational} + ec^{transfer}. \quad (18)$$

3. Energy Efficient VM Placement (EVMP) Algorithm

This section describes the proposed algorithm which is used to execute the workflow in an energy-efficient manner and within the deadline constraint as shown in Figure 3. To execute the workflow, there is a need to follow some set of rules, and these rules are presented in the form of the algorithm. The following steps are used during the workflow scheduling:

(Step 1) On the arrival of a new workflow, it is analyzed to get the type of the workflow, number of tasks, and dependency between them in the workflow. After that tasks are stored in the *task pool queue* ($task_{Pool}$). Check the parent tasks of the tasks. If the task is an entry task, then activate the new host and create a new VM on it based on the task requirement and allocate the task to VM for its execution. After that, update the start time, execution time, finish time of the task, and ready time of the VM

(Step 2) When any task executes successfully then check its child tasks. If any child task is ready for execution, then transfer the child task from $task_{Pool}$ to *ready queue* ($ready_{task}$)

(Step 3) When any task is in $ready_{task}$, then, check the relationship of that task with its parent tasks. If the task can be executed on the same VM on which its parent task(s) are executed without violation of its deadline, then allocate the task to that VM

(Step 4) If step 3 is not possible, then, sort the already deployed VM based on their energy consumption rate. If any VM fulfills the task requirement and the deadline is not violated, then, allocate the task to that VM

(Step 5) If step 4 is not possible, then, a new VM is created based on task requirement and allocated the task to the newly created VM. There are three cases to deploy the new VM on the host. In the first case, a new VM is deployed on the already active host. If this case is not possible, then, try to migrate any VM from one host to another and deploy the new VM on that host. If this is also not possible, then, try to activate the new host and deploy the VM on the newly created host

(Step 6) System status is updated such as energy consumption, makespan, and resource utilization

These scheduling steps are used to execute the workflow and are described in Algorithm 1.

Algorithm 1 is used to get the ready tasks for their execution. In this proposed algorithm, initially, all the tasks are stored in the task pool queue ($task_{Pool}$) and set ready task queue ($ready_{task}$) to null (see lines 1 and 2). If all the immediate parents of the task finish their execution or task is the entry task, then, that task is ready for its execution. Store that task in the $ready_{task}$, and remove it from $task_{Pool}$ (see lines 4-7). If there is any task in $ready_{task}$, then, the EVMP algorithm is used to schedule the tasks for their execution (see lines 8-10). This algorithm is automatically called on the arrival of a new workflow or completion of any task within the workflow.

Algorithm 2 is used to schedule the tasks. Initially make the tags such as findVM and findFlag, null and false, respectively (see lines 2 and 3). If the task is an entry task, then, select the VM type which can fulfill the task requirement. After that, start a new host and add this host to the active list H_a. Deploy the VM to the new host, and schedule the task on the new deployed VM. Also, update the ready time of the VM (see lines 4-12). If the task is not an entry task, then, firstly try to execute the task on the same VM on which its parent is executed. If it is possible, then schedule the task on the parent VM and update the ready tome, the transfer time (see lines 13-23). If this step is not possible, then call the alreadyDeployedVM() function (see lines 24-26).

Algorithm 3 is proposed to use the deployed VMs for workflow execution to save the VM creation time as well as energy consumption. In this function, firstly sort the deployed VM according to energy consumption rate (see line 2). If any VM can execute the task without violating the deadline, then, schedule task on that VM and update the system parameters such as ready time, the transfer time (see lines 3-10). If this step is not possible, then, call the scaleUp() function (see lines 11-13). The scale-down function is adopted from [18] to shut down the VMs and host to save energy consumption.

Algorithm 4 is used to add new resources for workflow execution. When already deployed VMs are unable to complete the workflow tasks then the scheduler calls this algorithm to install a new VM. This function is implemented from [26] with some variations. In this algorithm, firstly VM is selected which can fulfill the task requirement (see line 1). The new VM may be positioned on an already active host without migration based on the host resources (see lines

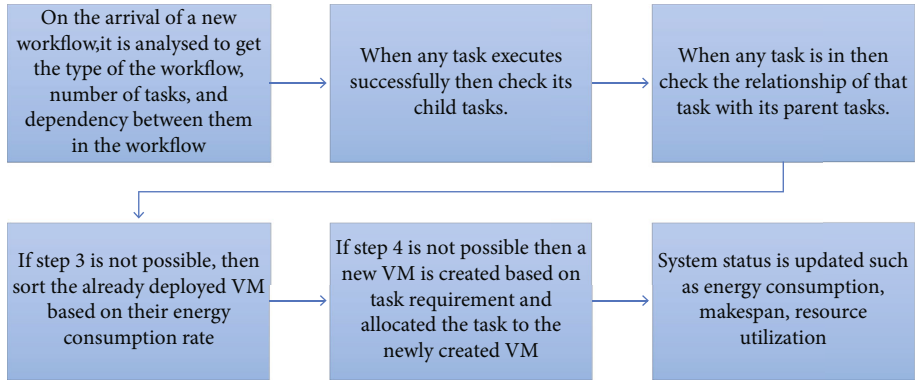

Figure 3: Flow diagram of energy-efficient VM placement (EVMP) algorithm.

1. add all the workflow tasks $(t_1, t_2, \cdots \cdots \cdots .t_n)$ to $task_{Pool}$
2. initiate $ready_{task} \longleftarrow null$;
3. for all tasks in the $task_{Pool}$
4. if immediate parents of the tasks are executed or task is entry task then
5. add t_i to $ready_{task}$ and remove it from $task_{Pool}$;
6. end if
7. end for
8. if $ready_{task}$ is not empty
9. schedule tasks by EVMP algorithm;
10. end if

Algorithm 1: Pseudocode to initiate the workflow scheduling.

5-8). If this is not possible, then, a new VM may be deployed on the already active host with live VM migration (see lines 9-19). If migration is not possible, then, a new host is triggered and a new VM is installed on it (see lines 20-24). Allocate the task to new VM, and remove the task from $\text{ready}_{\text{task}}$ (see line 25). VM ready time and if this task has parent, then, data transfer time from parent task to child task is restructured (see line 25).

4. Experimentation and Performance Metrics

In this section, the workflow model, simulation parameters, and performance metrics used in the proposed model are presented.

4.1. Considered Workflow Model. Workflow (W) is defined as a set of interdependent computational tasks [20]. In the literature [21], many scientific workflows such as LIGO, Montage, Cybershake, Epigenomics, and Pan-STARRS exist. In this paper, Pan-STARRS scientific workflow is considered for task execution. Pan-STARRS project continuously monitors the entire sky to detect moving or variable objects. PS1 telescope is used to monitor the sky. John Hopkins University and Microsoft manage the generated astronomy data using two types of workflows, i.e., PSLoad workflow and PSMerge workflow. PSLoad workflow is used to collect the data from the telescope and store data in the database. PSMerge workflow is used to update the database. PSLoad and PSMerge workflows are described in Figures 1 and 2, respectively. Table 1 describes the detailed characteristics of the workflows.

4.2. Simulation Parameters. CloudSim framework is exploited to simulate the cloud environment [27] and to check the usefulness of the anticipated scheduling model. Detailed simulation parameters are described below:

(i) HP ProLiant ML110 G4 and HP ProLiant ML110 G5 are two types of hosts are deployed [28]

(ii) The energy consumption rates of these two different types of hosts are 117 Watts per second (Ws^{-1}) and 135 Ws^{-1} [28]

(iii) The energy consumption rate to transfer 1GB of data is 2.3 W [29]

(iv) Four types of VM [19] are deployed with varying RAM (in MB) capacity and CPU speed (in MIPS). The configurations of different types of VMs are as follows: VM Type 1-500 MIPS with 613 MB RAM, VM Type 2-1000 MIPS with 1740 MB RAM, VM Type 3-2000 MIPS with 1740 MB, and VM Type 4-2500 MIPS with 870 MB RAM to execute the scientific workflow

(v) As per workflow requirements, the average VM start-up time is 96.9 s [30]

1. for all tasks in $ready_{task}$
2. $findVM \longleftarrow null$;
3. $findTag \longleftarrow FALSE$;
4. if the task is an entry task then
5. select vm_j type to execute the task within its deadline;
6. start a new host and add it to H_a;
7. deploy vm_j on the newly created host;
8. $findFlag \longleftarrow TRUE$;
9. schedule t_i to VM vm_j;
10. remove t_i from $ready_{task}$
11. update ready time of the VM;
12. end if
13. if the task is not an entry task then
14. $findFlag \longleftarrow FALSE$;
15. For all t_i in the $ready_{task}$
16. if parent vm_{jk} can execute the task without violating the deadline and $host_k$ is not overloaded then
17. schedule t_i on vm_{jk} ;
18. remove t_i from $ready_{task}$;
19. $rt_{jk} = rt_{jk} + et_{ijk} + tt_p^c$;
20. $findFlag \longleftarrow TRUE$;
21. $transferTime = transferTime + (calculate\ data\ transfer\ time\ using$ Eq. (2));
22. end if
23. end for
24. for each task t_i from $ready_{task}$
25. call alreadyDeployedVM();
26. end for
27. end if
28. end for

Algorithm 2: Pseudocode of EVMP algorithm.

1. initialize findFlag$\longleftarrow FALSE$;
2. sort the deployed VMs based on energy consumption rate in increasing order;
3. for all deployed vm_{jk}
4. if $d_i \geq rt_{jk} + et_{ijk} + tt_p^i$ then
5. $transferTime = transferTime + calculate\ data\ transfer\ time\ using\ using$ Eq. (5);
6. schedule t_i on vm_{jk} after time tt_p^c and remove it from $ready_{task}$;
7. $rt_{jk} = rt_{jk} + et_{ijk} + tt_p^c$;
8. findFlag$\longleftarrow TRUE$; break;
9. end if
10. end for
11. if findFlag== $FALSE$ then
12. call scaleUp();
13. end if

Algorithm 3: Pseudocode to schedule the task on existing VMs (alreadyDeployedVM).

1. select vm_j type to execute the task within its deadline;
2. sort all the hosts in the H_a list in decreasing order as per their utilization level;
3. $findTag \longleftarrow FALSE$;
4. for all $host_k$ in H_a
5. if host utilization does not exceed the upper threshold limit after VM allocation then
6. deploy $vm_j host_k$; $findTag \longleftarrow TRUE$; break;
7. end if
8. end for
9. if $findTag == FALSE$ then
10. select the $host_m$ which has minimum utilization level;
11. select the vm_n from $host_m$ which has minimum CPU capacity;
12. if vm_n cab be migrated to another host except $host_m$ then
13. migrate vm_n ;
14. end if
15. if $host_m$ utilization does not exceed the upper threshold limit after vm_j allocation then
16. deploy vm_j on $host_m$;
17. $findTag \longleftarrow TRUE$;
18. end if
19. end if
20. if $findTag == FALSE$ then
21. start a new host and add it to H_a;
22. deploy vm_j on the newly started host;
23. $findTag \longleftarrow TRUE$;
24. end if
25. allocate t_i to vm_j and remove t_i from $ready_{task}$;
26. update the ready time and transfer time;

Algorithm 4: Pseudocode to deploy new VM (scaleUp()).

(vi) In between VM, the average bandwidth is set to 20 MBPS, which is the imprecise bandwidth offered by Amazon Web Services [31]

(vii) Pan-STARRS real-world scientific workflow is considered. Each scientific workflow is divided into three groupings based on the number of tasks as defined in Table 1 [21]

4.3. Performance Metrics

4.3.1. Average Resource Utilization (ARU). ARU is defined as the ratio of assigned computing resources to accomplish the scientific workflow tasks and total computing resources available on the server. ARU is intended as:

$$\text{ARU} = \sum_{k=1}^{|H_a|} \sum_{j=1}^{|v_k|} \sum_{i=1}^{|T|} l_i.x_{ijk} \div \sum_{k=1}^{|H_a|} c_k \bullet \text{at}_k, \tag{19}$$

where at_k is the active time of the host h_k.

4.3.2. Total Energy Consumption (TEC). It defines the total energy which is consumed by the servers to execute a scientific workflow. TEC is computed using Equation (19).

4.3.3. Makespan or Total Execution Time. Makespan is the time taken to execute the scientific workflow from start tasks to the end task. It is computed using Equation (11).

5. Results and Discussion

The proposed EVMP algorithm is compared with an existing algorithm PESVMC algorithm [32] to establish the enhanced performance. In the existing PESVMC algorithm, the workflow tasks are allocated to the VM which depletes less energy. The deadline of tasks was not considered while assigning to the VM. Tasks were selected as per their parent-child relationship but during VM allocation for the task; the parent-child task relationship was not considered. As a result, the execution time and data transfer time both were increased which also affected both makespan as well as energy consumption. The performance of the EVMP algorithm is evaluated based on the ARU, total energy consumption, and workflow makespan.

5.1. Performance Impact on Resource Utilization. ARU of EVMP and PESVMC is observed for PSLoad and PSMerge scientific workflows with varying numbers of workflow tasks. Experimental result in terms of average resource utilization is shown in Figure 4. The result shows that EVMP performs better in terms of resource utilization in comparison to PESVMC. EVMP performs better because of its dynamic nature. In the proposed algorithm, when currently deployed VMs are not sufficient to complete the tasks within the deadline, then, only new VMs are created. So, resources are properly utilized. VM migration policy is also used to consolidate the resources which impressively increases

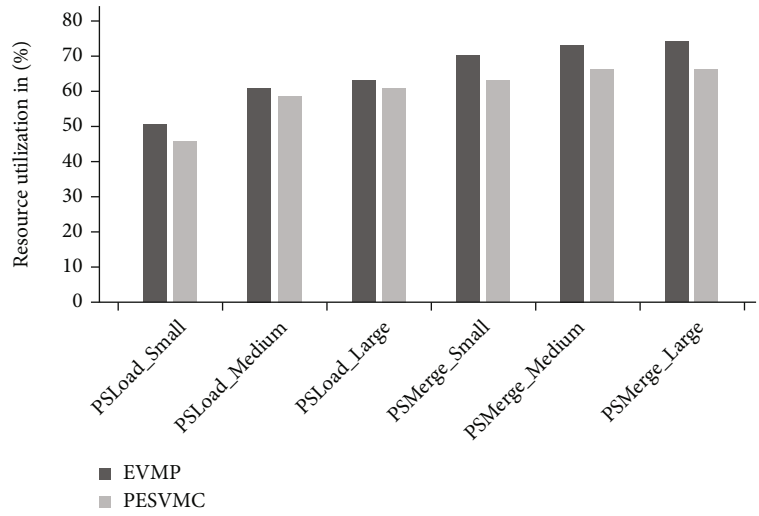

Figure 4: Performance impact on resource utilization.

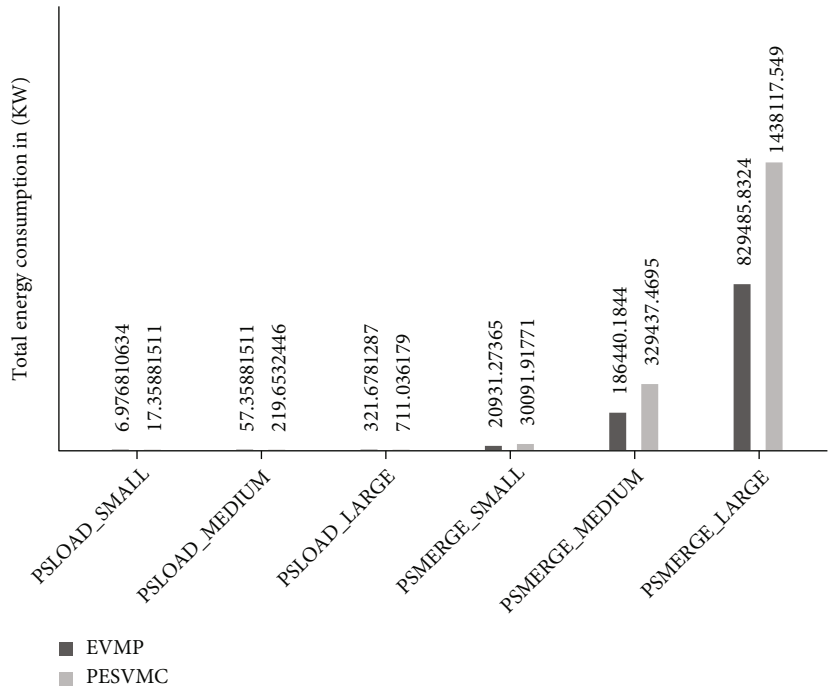

Figure 5: Performance impact on total energy consumption.

resource utilization. On average, 8.6% resource utilization is increased in comparison to the existing algorithm.

5.2. Performance Impact on Total Energy Consumption. The total energy consumption of EVMP and PESVMC is observed for PSLoad and PSMerge scientific workflows with the varying number of workflow tasks. Experimental result in terms of total energy consumption (measured in Kilowatt (kW)) is shown in Figure 5. In the existing algorithm, all the resources are active which consumes more energy without doing any useful work. But the EVMP algorithm deploys the resources as per the need of workflow tasks which impressively reduces the energy consumption. During the scheduling of workflow tasks, the existing algorithm does not consider the parent-child relationship which leads to the high data transfer energy consumption. But the proposed algorithm considers the parent-child relationship during task scheduling VM which helps to reduce the data transfer energy consumption and workflow makespan. On average, 42.3% of energy consumption is reduced by the EVMP algorithm in comparison to the PESVMC algorithm.

5.3. Performance Impact on Makespan. The makespan of EVMP and PESVMC is observed for PSLoad and PSMerge scientific workflows with the varying number of workflow tasks. Experimental result in terms of makespan (measured

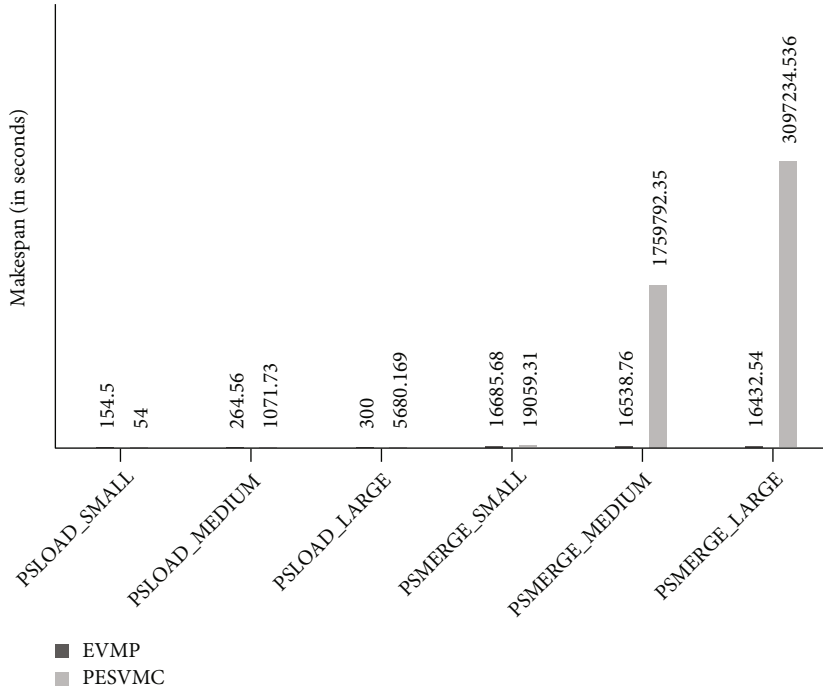

Figure 6: Performance impact on makespan.

in seconds (s)) is shown in Figure 6. Makespan result shows that EVMP performs better in terms of makespan. On average, 98% makespan is reduced in comparison to the PESVMC algorithm. This is due to limited resources being considered in the PESVMC algorithm which impressively reduces the parallel execution of tasks. In the existing algorithm, the parent-child relationship is not considered during task scheduling to the VMs which has also affected the makespan of the workflow. Hence, makespan of PESVMC is significantly increased for a large dataset of workflow.

6. Conclusion

The paper presents an energy aware VM placement model for the dependent scientific workflows in the cloud which achieves scheduling objectives and energy efficiency and improves the system performance for real-world scientific workflows. The proposed EVMP algorithm has reduced the energy consumption by applying DVFS (hardware technique) for the VMs/hosts which are not performing any work or idle computing resources, and software techniques for VMs and hosts which are idle beyond the preestablished threshold time. The data transfer energy consumption is minimized by scheduling tasks on or around the parent VM (where parent task is executed), and it also helped in reducing the execution delay by decreasing the transfer time and VM creation time. The EVMP algorithm is implemented on the CloudSim framework. The Pan-STARRS real-world scientific workflows are considered for evaluating the performance of the EVMP algorithm. The EVMP algorithm has increased resource utilization by 8.6% in comparison to the PESVMC algorithm. The energy consumption has been decreased by 42.3%, and makespan has been reduced by 98% in comparison to PESVMC algorithms. The proposed EVMP algorithm will also be implemented on a public cloud platform along with the evaluation of additional performance metrics of security and fault tolerance in the future.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest regarding the publication of this study.

References

[1] J. Sahni and D. P. Vidyarthi, "A cost-effective deadline-constrained dynamic scheduling algorithm for scientific workflows in a cloud environment," *IEEE Transactions on Cloud Computing*, vol. 6, no. 1, pp. 2–18, 2018.

[2] K. Kanagaraj and S. Swamynathan, "Structure aware resource estimation for effective scheduling and execution of data intensive workflows in cloud," *Future Generation Computer Systems*, vol. 79, pp. 878–891, 2018.

[3] L. Ramakrishnan and D. Gannon, *A Survey of Distributed Workflow Characteristics and Resource Requirements*, Indiana University, 2008.

[4] Y. C. Lee, H. Han, A. Y. Zomaya, and M. Yousif, "Resource-efficient workflow scheduling in clouds," *Knowledge-Based Systems*, vol. 80, pp. 153–162, 2015.

[5] J. Wang, A. Taal, P. Martin et al., "Planning virtual infrastructures for time critical applications with multiple deadline

constraints," *Future Generation Computer Systems*, vol. 75, pp. 365–375, 2017.

[6] N. Garg and M. S. Goraya, "Task deadline-aware energy-efficient scheduling model for a virtualized cloud," *Arabian Journal for Science and Engineering*, vol. 43, no. 2, pp. 829–841, 2018.

[7] M. Kumar and S. C. Sharma, "PSO-COGENT: cost and energy efficient scheduling in cloud environment with deadline constraint," *Sustainable Computing: Informatics and Systems*, vol. 19, pp. 147–164, 2018.

[8] I. M. Ibrahim, "Task scheduling algorithms in cloud computing: a review," *Turkish Journal of Computer and Mathematics Education*, vol. 12, no. 4, pp. 1041–1053, 2021.

[9] L. Jia, K. Li, and X. Shi, "Cloud computing task scheduling model based on improved whale optimization algorithm," *Wireless Communications and Mobile Computing*, vol. 2021, Article ID 4888154, 2021.

[10] X. Chen, L. Cheng, C. Liu et al., "A WOA-based optimization approach for task scheduling in cloud computing systems," *IEEE Systems Journal*, vol. 14, no. 3, pp. 3117–3128, 2020.

[11] L. Benini, A. Bogliolo, and G. D. Micheli, "A survey of design techniques for system-level dynamic power management," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 8, no. 3, pp. 299–316, 2000.

[12] N. Garg, D. Singh, and M. S. Goraya, "Energy aware hardware and software approaches in cloud environment," *International Journal of Computer Science & Communication Networks*, vol. 7, no. 3, pp. 66–69, 2017.

[13] H. Kimura, M. Sato, Y. Hotta, T. Boku, and D. Takahashi, "Emprical study on reducing energy of parallel programs using slack reclamation by DVFS in a power-scalable high performance cluster," in *Proceedings of the 2006 IEEE International Conference on Cluster Computing (2006)*, pp. 1–10, Barcelona, Spain, 2006.

[14] I. Pietri and R. Sakellariou, "Energy-aware workflow scheduling using frequency scaling," in *Proceedings of the 2014 43rd International Conference on Parallel Processing Workshops (2014)*, pp. 104–113, Minneapolis, MN, USA, 2014.

[15] Z. Tang, L. Qi, Z. Cheng, K. Li, S. U. Khan, and K. Li, "An energy-efficient task scheduling algorithm in DVFS-enabled cloud environment," *Journal of Grid Computing*, vol. 14, no. 1, pp. 55–74, 2016.

[16] X. Xu, W. Dou, X. Zhang, and J. Chen, "EnReal: an energy-aware resource allocation method for scientific workflow executions in cloud environment," *IEEE Transactions on Cloud Computing*, vol. 4, no. 2, pp. 166–179, 2016.

[17] A.-C. Orgerie, L. Lefèvre, and J.-P. Gelas, "Save watts in your grid: green strategies for energy-aware framework in large scale distributed systems," in *Proceedings of the 2008 14th IEEE International Conference on Parallel and Distributed Systems (2008)*, pp. 171–178, Melbourne, VIC, Australia, 2008.

[18] N. Garg, D. Singh, and M. S. Goraya, "Energy and resource efficient workflow scheduling in a virtualized cloud environment," *Cluster Computing*, vol. 24, no. 2, pp. 767–797, 2021.

[19] A. Beloglazov and R. Buyya, "Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in cloud data centers," *Concurrency and Computation: Practice and Experience*, vol. 24, no. 13, pp. 1397–1420, 2012.

[20] J. Liu, E. Pacitti, P. Valduriez, and M. Mattoso, *Parallelization of Scientific Workflows in the Cloud*, HAL, 2014.

[21] "Workflow generator," https://confluence.pegasus.isi.edu/display/pegasus/WorkflowGenerator.

[22] W. Lin, H. Wang, Y. Zhang, D. Qi, J. Z. Wang, and V. Chang, "A cloud server energy consumption measurement system for heterogeneous cloud environments," *Information Sciences*, vol. 468, pp. 47–62, 2018.

[23] D. Kliazovich, P. Bouvry, and S. U. Khan, "GreenCloud: a packet-level simulator of energy-aware cloud computing data centers," *The Journal of Supercomputing*, vol. 62, no. 3, pp. 1263–1283, 2012.

[24] K. Bousselmi, Z. Brahmi, and M. M. Gammoudi, "Energy efficient partitioning and scheduling approach for scientific workflows in the cloud," in *Proceedings of the 2016 IEEE International Conference on Services Computing (2016)*, pp. 146–154, San Francisco, CA, USA, 2016.

[25] Z. Li, J. Ge, H. Hu, W. Song, H. Hu, and B. Luo, "Cost and energy aware scheduling algorithm for scientific workflows with deadline constraint in clouds," *IEEE Transactions on Services Computing*, vol. 11, no. 4, pp. 713–726, 2015.

[26] X. Zhu, L. T. Yang, H. Chen, J. Wang, S. Yin, and X. Liu, "Real-time tasks oriented energy-aware scheduling in virtualized clouds," *IEEE Transactions on Cloud Computing*, vol. 2, no. 2, pp. 168–180, 2014.

[27] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. F. De Rose, and R. Buyya, "CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," *Software: Practice and Experience*, vol. 41, no. 1, pp. 23–50, 2011.

[28] N. Garg, D. Singh, and M. S. Goraya, "Power and resource-aware VM placement in cloud environment," *Proceedings of the 8th international advance computing conference (IACC)*, 2018, pp. 113–118, Greater Noida, India, 2018.

[29] S. Sharma, *Trends in Server Efficiency and Power Usage in Data Centers*, SPEC 2016 ASIA SUMMIT, Auckland, 2016.

[30] M. Mao and M. Humphrey, "A performance study on the VM startup time in the cloud," in *Proceedings of the 2012 IEEE Fifth International Conference on Cloud Computing (2012)*, pp. 423–430, Honolulu, HI, USA, 2012.

[31] M. Palanker, A. Lamnitchi, M. Ripeanu, and S. Garfinkel, "Amazon S3 for science grids: a viable solution?," in *Proceedings of the 2008 international workshop on data-aware distributed computing ACM*, pp. 55–64, New York, NY, United States, 2008.

[32] N. Mohanapriya, G. Kousalya, P. Balakrishnan, and C. Pethuru Raj, "Energy efficient workflow scheduling with virtual machine consolidation for green cloud computing," *Journal of Intelligent Fuzzy Systems*, vol. 34, no. 3, pp. 1561–1572, 2018.