Wireless Personal Communications (2021) 119:1301–1320
https://doi.org/10.1007/s11277-021-08263-z

Energy Efficient and Reliability Aware Workflow Task Scheduling in Cloud Environment

Rambabu Medara[1] · **Ravi Shankar Singh**[1]

Accepted: 4 February 2021 / Published online: 19 February 2021

Abstract
With the rising demand for cloud services, the high energy consumption of cloud data centers is a significant problem that needs to be handled. The Dynamic Voltage and Frequency Scaling approach has been identified as one of the efficient techniques to conserve energy, particularly while scheduling real-world scientific workflows. Moreover, scientific workflows demand high-availability of the system. The computational systems in the cloud data centers are not failure-free and further frequency scaling impacts negatively on the reliability of the system by increasing the transient fault rate. A trade-off is required between energy conservation and the reliability of the computational machine. In this paper, we propose an energy-efficient and reliability aware workflow task scheduling in a cloud environment (EERS) algorithm, which conserves energy and maximizes the system reliability. The EERS comprises five sub-algorithms. First, we apply a task rank calculation algorithm to preserve the task dependencies. Second, a task clustering algorithm to reduce the communication cost which reduces energy consumption. The third is the sub-target time distribution algorithm to define the sub_makespan for each task. Further, we propose a cluster-VM mapping algorithm that reduces energy and maximizes system reliability and finally, a slack algorithm to reclaim slack associated with the non-critical tasks. The performance of the EERS evaluated on the WorkflowSim simulator using two real-world scientific workloads CyberShake and Montage. The results indicate that it surpasses the related existing approaches.

Keywords Cloud computing · Workflow scheduling · Enenrgy-efficiency · Reliability

1 Introduction

Cloud computing is the on-demand delivery of computing services such as servers, storage, databases, networking, software, analytics, intelligence over the Internet to offer faster innovation, flexible resources, and economies of scale [1]. Because of the sustained growth rate for cloud services, there has been substantial growth in energy

✉ Rambabu Medara
medararambabu.rs.cse18@iitbhu.ac.in

[1] Department of Computer Science and Engineering, Indian Institute of Technology (BHU), Varanasi, Uttar Pradesh 221005, India

consumption (EC) of the cloud data centers. This demand for energy consumption results leads to high operating costs and higher carbon emissions [2]. As a consequence, the data centers become unsustainable [3]. The information technology (IT) infrastructure is the main contributor to energy consumption in a data center, which includes servers and other IT components. Optimization of energy consumed by the cloud-related infrastructure is the major research deal in recent years [4].

Most of the business and complex scientific applications used workflows to analyze complex data sets and to conduct simulation experiments effectively [5]. A scientific workflow involves individual data transformations and analysis steps, and mechanisms to link them according to the data dependencies among them [6]. Several scientific applications comprise numerous tasks with precedence constraints, including chemistry, physics, earth science, astronomy, computer science, and bioinformatics. The scientific workflow applications are highly complex with various sizes of tasks, which are in the form of a Directed Acyclic Graph (DAG). Therefore, scientific workflow management mostly deals with the large-volumes of data. Such complex workflow applications demand high computing power and high system reliability. With the rapid proliferation of accessing scientific applications, it is necessary to focus on developing the energy-aware workflow scheduling model in a cloud environment.

The scheduling of workflow tasks in distributed platforms such as grids and cloud environments has been extensively considered for many years. Researchers have developed algorithms geared towards different environments: from small-scale homogeneous clusters to large-scale community grids, to the contemporary paradigm, heterogeneous, utility-based, and resource-rich cloud computing [5]. Clouds provide unlimited computing power which is a more relevant platform to execute complex workflow applications. But the computational systems in cloud data centers are not failure-free and any type of fault may be critical to the running application [7]. Particularly, the transient faults increase by the Dynamic Voltage and Frequency Scaling (DVFS) approach. Such failures may impact the execution nature of the program and provoke unpredictable results [8]. In this work, we proposed an efficient heuristic for energy efficient and reliability aware workflow tasks scheduling in cloud environment (EERS) which maximizes the reliability for workflow tasks while conserving energy.

The primary contributions of this article are summarized as follows:

- An energy and reliability aware workflow scheduling algorithm, called EERS is proposed to reduce the application energy consumption while maximizing the system reliability.
- The proposed EERS approach consists of five sub-algorithms such as task rank calculation, task clustering, sub-target time distribution, cluster-VM mapping algorithm, and slack algorithm to meet the stated objectives.
- Along with energy and reliability aware resources management of our approach, the task clustering algorithm reduced communication cost and the slack algorithm reduced a significant amount of energy consumption.
- The performance of the proposed EERS approach was evaluated through several numerical experiments. And the experimental results show that the proposed approach can achieve a significant amount of energy-saving while maximizing the application reliability without regard to the diverse workflow structures.

The rest of the article is organized as follows. The related work is discussed in Sect. 2. We propose system architecture and models in Sect. 3. Then Sect. 4 presents the five

sub-algorithms and EERS algorithm implementation. Section 5 gives the algorithm performance evaluation and finally the conclusion and future work discussed in Sect. 6.

2 Related Works

In recent years, there has been significant attention in developing the models to provide workflow scheduling and energy-efficient resource management in the cloud environment. The optimal workflow scheduling and resource management depend on various factors such as the arrival rate of the requests, availability of the resources, and the workload of workflow tasks.

Some of the task scheduling mechanisms for the cloud environment addressed optimization parameters such as schedule length and cost. A fuzzy dominance-based HEFT algorithm to address the cost and makespan of workflow applications in clouds [9], which uses real-world pricing and resource models. A cost-aware large-scale workflow scheduling approach in [10] with DAG splitting mechanism reduces the monetary cost of application by maximizing the VM utilization. A list-based heuristic called Heterogeneous Earliest Finish Time (HEFT) [11] approach gives the best makespan. A heuristic called Jaya algorithm in [12] is used to reduce both the computation and communication costs while scheduling the workflow tasks in the cloud environment. A Case Library and Pareto Solution-based hybrid Genetic Algorithm (CLPS-GA) [13] has focused on two objectives such as makespan and energy conservation while scheduling the workflow tasks. It relies on a case library and multiparent crossover operator to effectively ensure the stability, diversity, and convergence in the solution. An approach in [14] has employed an immune genetic algorithm to resolve the QoS constraint satisfaction by considering five objectives for workflow scheduling in the cloud environment. The deadline-based workflow scheduling algorithm [15] employs the Particle Swarm Optimization (PSO) technique to diminish the overall execution cost while scheduling the scientific workflow applications in the Infrastructure-as-a-Service (IaaS) clouds without violating the deadline constraints. However, these works are not considered either energy or reliability objectives.

Many efficient scheduling techniques to maximize system reliability have been studied [16–18]. Some of the task scheduling approaches developed for on-demanding computing environments addressed the reliability of the system and performance. A reliability model and load-balanced scheduling approach introduced in [19] by using Colored Petri Nets (CPN). A task scheduling model is developed in [20] using Ant Colony Optimization (ACO) which is carried out by first estimating the availability of system so that the most reliable nodes are selected for task transactions. In [21] an efficient task allocation approach developed using Social Spider Optimization (SSO) to maximize the system performance and reliability. A Honey Bee Optimization (HBO)–based technique proposed in [22] to solve the load-balanced transaction scheduling. The above works ignored the objectives of energy conservation.

A green energy-efficient scheduling approach [23] employs the DVFS technique which enables the computing processors to run the task at low voltages and low frequencies. It effectively utilizes the cloud data center resources through task to-VM allocation based on the task dependencies of an application to reduce energy consumption and makespan. An energy-efficient heuristic is proposed in [24] to schedule real-time workflow applications in clouds. It saves energy by effectively utilizing the schedule gaps using per-core DVFS and approximate computations. Similar work in [25] addressed energy, monetary cost, and

quality of service objectives. A polynomial-time multi-objective heuristic proposed in [26] to schedule time-constrained tasks on the cloud environment, which optimize energy, cost, and resource utilization while maintaining Service Level Agreements (SLAs). Stackelberg game based Game-Score simulator developed in [27] to schedule the tasks in the cloud environment, which trade-off between energy consumption and schedule length of a workflow. A load-balancing based approach for scheduling tasks in a distributed cloud environment in [28], optimizes resource utilization by estimating task execution time based on the cloud status and queuing model used to improve response time. A PSO based multi-objective approach for workflow scheduling in clouds optimizes cost, schedule length, and resource utilization while considering system reliability [29]. From the review of the existing state-of-art scheduling strategies, it is observed that most of these strategies focus on multi-optimization without considering the reliability of the system.

3 System Architecture and Models

Our problem subsists in scheduling the workflow tasks with user-specified deadline. Our approach aims to minimize energy utilization and maximizing system reliability. To propose such an efficient technique we discussed our system architecture and models in this section. We propose system models such as the cloud datacenter model, application model, energy model, cloud system architecture, and system reliability model. For ease of insight Table 1 outlines, the primary notations with their meanings used all over in this work.

Table 1 Primary notations

Notation	Definition
PM	Set of PMs (physical machines), ($pm_k \in PM, 1 \leq k \geq PM$)
VM	Set of virtual machine ($vm_k \in VM, 1 \leq k \geq v$)
W	Workflow with set of tasks T_W ($t_i \in T_W, 1 \leq i \geq n$)
t_i	ith task of W
C	Set of communication edges in DAG ($c_{ij} \in C, 1 \leq I, j \geq n \quad and \quad i \neq j$)
t_{entry}	Entry task of W
t_{exit}	Exit task of W
$T(t_i, vm_k)$	Time to execute task t_i on vm_k
$\overline{T}(t_i)$	Average execution time of task t_i on various VMs
c_{ij}	An edge from task t_i to task t_j, ($i, j \in C \quad and \quad i \neq j$)
$w(c_{ij})$	Weight of the edge c_{ij}
$T(t_{ij})$	Data transfer time from tasks t_i to t_j
T_M	Makespan of W
T_D	Deadline
$E_{Dynamic}$	Dynamic energy
E_{Static}	Static energy
C_{eff}	Effective loading capacitance
f_{op}^{k}	Operating frequency of vm_k
$f_{op}^{i,k}$	Operating frequency of the t_i on vm_k
R_W	Application reliability

3.1 Datacenter Model

The cloud datacenter model considered in this work comprises M heterogeneous physical machines (PM) $PM = \{pm_1, pm_2, \ldots, pm_M\}$. All PMs supports DVFS, each of which can operate at k number of frequency(f) levels $(f_1, f_2, \ldots, f_k)$ and the time to switch among frequencies approximately takes 10–150 μm which is negligible [30]. The DVFS approach operates processor frequencies under various voltage levels. Every PM is describing with different types of resources such as CPUs, memory capacity, bandwidth, network I/O, and the storage size. These physical machines virtualized into v number of virtual machines (VMs) each of which considered operating at a different frequency level (DVFS enabled) and PM operating frequency attribute to its VMs. A VM can be characterized with maximum computing performance in a Million Instructions Per Second (MIPS), bandwidth (B) etc. Hence, the cloud workflow scheduler has distinct possibilities in selecting the suitable VM to execute a task by meeting workflow constraints. Generally, we consider computing performance relates to the processor frequency. A kth virtual machine vm_k at some level of operating frequency represented as f_{op}^k.

3.2 Application Model

In general, a workflow W with dependencies among tasks $Tw = \{t_1, t_2, \ldots, t_n\}$ is modeled as a DAG. A DAG $W = (T_w, C)$ where C is the set of edges or directed arcs represents the dependencies among tasks. An edge c_{ij} is the dependency from t_i to t_j where t_i is one of the parents of t_j and t_j is one of the children of t_i. A task without a parent(s) or predecessor(s) is called an entry task t_{entry} and a task without child(s) or successor(s) is called exit task t_{exit}. A task is prompt to execute when all required resources of its available. When the job t_i completes its execution and its generated output transfers to its children. The data transferred (in MB) from task t_i to its child t_j has represented the weight on the edge c_{ij} as $w(c_{ij})$. We denote the overall execution time of workflow (makespan) as T_M and associated workflow deadline as T_D. For any workflow execution deadline T_D is describe as a time constraint and is user-defined.

The data communication time $T(t_{ij})$ between two precedence constraint tasks is calculated as in Eq. (1)

$$T(t_{ij}) = \frac{w(c_{ij})}{B} \tag{1}$$

where B is the bandwidth and $w(c_{ij})$ is the data (in MB) communicated between tasks t_i and t_j. The execution cost of any task t_i calculated as in Eq. (2)

$$T(t_i, vm_k) = \frac{w_i}{f_{max}^k} + T(t_{ij}) \tag{2}$$

where $T(t_i, vm_k)$ is the task t_i execution time on vm_k and it includes effective execution time and data communication time. The mean execution time of task t_i is the mean of execution times on various available VMs and is calculated as follows

$$\overline{T}(t_i) = \sum_{k=1}^{n} \frac{T(t_i, vm_k)}{n} \tag{3}$$

where n is the number of VMs. The earliest start time EST and as well as earliest finish time EFT of task t_i are calculated as follows

$$EST(t_i) = \begin{cases} 0 & \text{if } t_i = t_{entry} \\ max_{t_p \in parent(t_i)} EFT(t_p) & \text{otherwise} \end{cases} \quad (4)$$

$$EFT(t_i) = EST(t_i) + T(t_i, vm_k) \quad (5)$$

The $EFT(t_{exit})$ is the minimum makespan of the W $minT_M = EST(t_{exit})$. Without loss of generality, we consider user defined deadline T_D should be greater than the $minT_M$ i.e. $T_D > minT_M$.

3.3 Energy Model

The power utilization of computational servers of cloud data centers is due to CPU, memory, network interfaces, storage disks, and other underlying circuits. In comparison with other computing resources, CPU dominates energy consumption. The power consumption of any workflow execution on cloud infrastructure is includes frequency-independent, dynamic and static power consumption:P_{ind}, $P_{Dynamic}$, and P_{Static}. The P_{Static} is the sleep or static power dissipation (to keep the clock circuit running, to maintain the basic circuits etc., which can be eliminated only by turning off the system) and it calculated as in Eq. (6),

$$P_{Static} = V * I \quad (6)$$

where I is the current into the device and V is the supply voltage, P_{ind} is independent power consumption, which is free from CPU frequency and supply voltage. The P_{ind} consists of components such as memory, storage disks, I/O and other network devices and it can be reduced to the negligible amount by keeping the system in standby mode [31].The dynamic power dissipation $P_{Dynamic}$ is the dominant component of energy consumption in widely attractive CMOS (Complementary Metal Oxide Semiconductor) circuits. It is the sum of the transient power consumption which is the sum of switching and through current and capacitive-load power consumption [32]. The $P_{Dynamic}$ depends on the frequency and voltage supply of the CPU [33] and it can be estimated as in Eq. (7)

$$P_{Dynamic} = C_{eff} V^2 f = C_{eff} f^3 \quad (7)$$

where C_{eff} is the effective loading capacitance, f is the clock frequency, and V is the supply voltage. The clock frequency of CPU f is directly proportional to supply voltage V. When application runs at lower frequency levels, then supply voltage reduces linearly. Our scheduling approach minimizes this component by maximizing the application reliability. To emphasize both dynamic and static power dissipation in a system, we adopt the system power model suggested in [31] and afterward used in [30, 34] and the system power (P) consumption have given by Eq. (8).

$$\begin{aligned} P &= P_{Static} + h(P_{ind} + P_{Dynamic}) \\ &= P_{Static} + h(P_{ind} + C_{eff} V^2 f) \end{aligned} \quad (8)$$

where h describes the state of the system, which is 0 to specify the system is in sleeping mode and 1 for active mode. As the dynamic power utilization is the most compelling component and the other factors are ignored in this paper. Energy is defined as a product of power and time then the energy consumption when the machine is running calculated by using Eq. (9).

$$E = P * t \tag{9}$$

The energy consumption of a specific task t_i with computation cost w_i, executing on vm_k with operating frequency f_{max} calculated as follows

$$E(f_{op}^{i,k}) = P_{ind}\left(\frac{w_{ik}}{f_{op}^{i,k}}\right) + C_{eff}\left(f_{op}^{i,k}\right)^2 T(t_i, vm_k) \tag{10}$$

and the total energy consumed for the application given by the sum of the energies of individual tasks in application and is calucted using Eq. (11).

$$E_{Total} = \sum_{i=1}^{n} E\left(f_{op}^{i,k}\right) \tag{11}$$

For simplicity, we considered only dynamic energy consumption in this work.

3.4 System Architecture

This paper aims to maximize system reliability and then reduce the energy consumption of cloud data centers. The works in [35–37] introduced cloud system architecture for implementing workflow applications in an energy-efficient way, and a reference architecture for Cloud Workflow Management System (CWfMS) in [5] empowers the establishment and execution of workflows. Based on these works, we introduce a system reference architecture for Energy-efficient Cloud Workflow Management System (eCWfMS) depicted in Fig. 1. The various components depicted in Fig. 1 are standard for most of CWfMSs.

User Interface enables the users to set up, modify, submit, and track their applications.

Workflow Engine is the essential component of the eCWfMS and is accountable for the execution of workflow application for this it can have various sub-components such as (a) workflow Parser is responsible for converting high-level workflow descriptions such as XML to internal specifications such as objects, tasks, parameters, and dependencies which are accessed by the scheduler component. (b) cloud scheduler works with the (c) resource provisioning modules for panning the execution of the actual workflow algorithm. The overall performance of the system, reliability, and energy consumption depends on the efficiency of the scheduler. For energy-efficient and reliability aware workflow execution, it needs to interact with different components such as:

Service analyzer is to analyze the service requirements of an application, reliability and energy designer to minimize the execution cost while meeting the QoS requirements.

Service scheduler is to allocate the requests to computing resources such as leased VMs and takes the decision when to start or stop the service of VMs. It often interacts with the *VM Manager* to update the status of VMs and as well as to provision new VMs across the physical machines of cloud infrastructure.

*Administration and monitoring tools*include monitoring modules for tracking the status and performance of workflow tasks, continuously and dynamically and enables leased resource management such as VMs. The collected data by these can be stored in historical databases, which can be useful for performance predictions of the system.

Cloud Information Service (CIS) provides information about different cloud service providers, their resource types, such as VM types, including pricing models. For example, Amazon EC2 giving different kinds of instances (a virtual server) for various instance

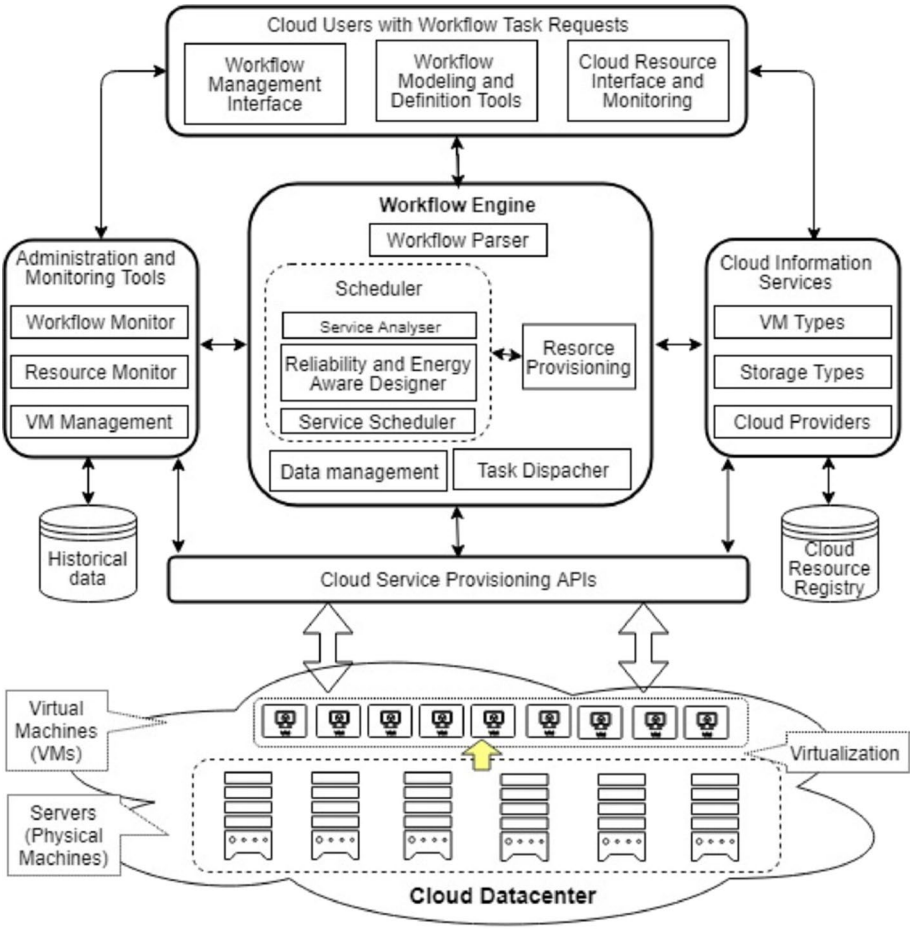

Fig. 1 Reference cloud system architecture

families such as small, medium, large and xlarge (extra-large), the computing capabilities such as CPU computing power, memory, and storage depends on the instance type.

Cloud Service Provisioning API is the type of application programming interface, which is a gateway to enable direct or indirect cloud services to users. In particular, for scheduling problems which we considered in this work, the cloud APIs can enable provisioning and removal of the VMs on-demand and monitor the provisioned VM's resources, storage, security, and network configuration.

3.5 System Reliability

During the execution of an application, faults may occur due to hardware breakdown, software failures, cosmic ray radiation, etc. As the frequency of transient faults significant than permanent and intermittent faults [30, 33], we focused on transient faults in this paper. Cloud provides sharable heterogeneous computing infrastructure through the internet. The

computing resources (in particular processors) failures are inevitable and have conflicting effects on application performance and energy consumption. These failures of processors are discrete events and assumed to follow a Poisson process [38]. The operating frequency of CPU influence the fault arrival rate λ [30]. This fault arrival rate λ influences the performance of a node where a computation-intensive application running and hence the reliability of such node is essential.

We assume that the transient faults happen while individual tasks are in execution. However, with the effect of DVFS, the systems operating frequency can influence the error arrival rate and its corresponding supply voltage. Therefore, the fault rate is represented as in [33] is given in Eq. (12)

$$\lambda(f_{op}^{k}) = \lambda_0 . g(f_{op}^{k}) \tag{12}$$

where λ_0 is the initial fault arrival rate at f_{max}^{k}, f_{op}^{k} is operating frequency of vm_k, $g(f_{op}^{k})$ is a decreasing function and $g(f_{max}^{k}) = 1$. Generally, Eq. (12) is known as an exponential relation between the λ and the circuit's critical cost. In our scheduling approach and experimental analysis, we assume the model proposed in [33] and afterward used in [30] expressed as exponential model as in Eq. (13)

$$\lambda(f_{op}^{k}) = \lambda_0 g(f_{op}^{k}) = \lambda_0 . 10^{\frac{d(1-f_{op}^{k})}{1-f_{min}^{k}}} \tag{13}$$

where λ_0, f_{op}^{k} and $g(f_{op}^{k})$ are same as mentioned before. The positive constant d stands for the faulty rate dependency on frequency scaling and corresponding voltage. It can be easy to perceive exponential increase in λ with frequency scaling for energy saving, i.e. λ is maximum at the minimum allowed CPU frequency.

$$\lambda_{max} = \lambda_0 . 10^{d}, \quad \text{for} \quad f_{op}^{k} = f_{min}^{k} \tag{14}$$

Considering the transient fault model in [30, 33], which follows Poisson distribution model, the reliability R of a task t_i running on vm_k is calculated as follows

$$R(f_{op}^{i,k}) = e^{-\lambda(f_{op}^{k}) . \frac{T(t_i, vm_k)}{f_{op}^{k}}} \tag{15}$$

where $f_{op}^{i,k}$ is the operating frequency of the node where task t_i running. The reliability of application W with n number of tasks is the product of individual task reliability

$$R_W = \prod_{i=1}^{n} R(f_{op}^{i,k}) \tag{16}$$

4 Proposed Algorithm

In this section, we discussed our proposed approach for workflow scheduling in a cloud environment. The proposed EERS approach is capable of minimizing energy consumption and maximizing system reliability while meeting the user-defined deadline. It includes four sub-algorithms, such as task rank algorithm, task clustering algorithm, sub-target time distribution algorithm, and cluster-VM mapping algorithm, which reduce energy consumption

and maximize the system reliability. This section presents an implementation of these sub-algorithms precisely.

4.1 Task Rank Calculation Algorithm

Workflow tasks rank order is established in this stage to fulfill the requirement of task scheduling. The task ranks are established in such a way to meet the precedence constraints and finds a topological order for scheduling. To prioritize tasks in W without disturbing dependencies, each task t_i is assigned a rank $rank(t_i)$, that can be computed recursively staring with the exit task t_{exit} [30, 34] as follows

Step-1 The exit tasks rank initialized to its average computing time

$$rank(t_{exit}) = \overline{T}(t_{exit}) \tag{17}$$

Step-2 For each task compute rank recursively according to the following expression

$$rank(t_i) = \overline{T}(t_i) + max_{t_j \in child(t_i)} rank(t_j) \tag{18}$$

where $\overline{T}(t_i)$ is the average computation time of t_i on different VMs. Estimate ranks of all tasks by repeating the above steps for each task in workflow.

4.2 Task Clustering Algorithm

Once the parent task completes execution, its generated output transfers to its child tasks. If both are scheduled on different VMs, then communication energy consumption incurs. A large amount of data communication energy is consumed during inter-processor communication. We can avoid this energy consumption by grouping the tasks and then schedule on the same machine. Consider three tasks t_i, t_{i+1} and t_{i+2} with the dependency shown in Fig. 2, the task t_{i+2} has two parents, t_i and t_{i+1} with communication costs 3 and 5, respectively. Communication energy can be saved by grouping t_{i+1} and t_{i+2} to schedule on the same VM.

We adopt the clustering approach in [30] for this paper and we extended it to minimize communication energy as follows:

Step-1 Staring from the entry task $t_i = t_{entry}$, for every task t_i, if task t_i not yet earmark for any cluster, then make a new cluster cl_l

Step-1a add t_i to cl_l and sort the children of t_i

Step-2 For each child t_j of t_i, if cluster not been assigned and parents are assigned to a cluster

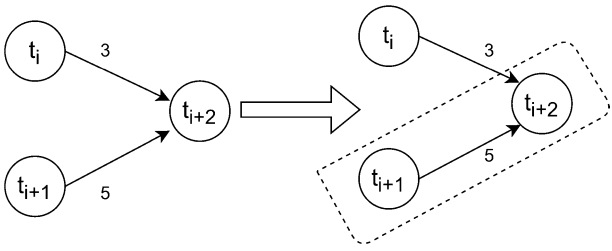

Fig. 2 Clustering of tasks

Step-2a if a task t_j has more than one parent, then check its parents which transfers more data to it. Such parent can find as follows

$$t_k = max_{t_k \in parent(t_j)} w_{kj}, \tag{19}$$

if $EFT(t_j) \leq EST(t_i)$ then $t_i \leftarrow t_k$,goto Step2

Step-2b $t_i \leftarrow t_j$

Repeat Step 2 for entire graph W, until each task assigned to some cluster.

4.3 Sub-target Time Distribution Algorithm

The target time to complete each task is based on the T_D is distributed to each task of the workflow. It is a proportionate increase in the effective execution time (makespan) of individual tasks and hence the application. This makespan extension can be done by reducing the frequency of the processor where it is running; hence, we can save energy. To implement sub-target time distribution we suggested a simple algorithm that can compute in polynomial time and steps for sub-target time distribution algorithm are as follows

Step-1 Each task t_i in a given workflow, calculate the sub-target time using Eq. (20). The sub-target time is the deadline to complete execution of task t_i.

$$T_{sub-target}(t_i) = T(t_i).\frac{T_D}{minT_M} \tag{20}$$

where $T(t_i)$ is the effective execution time of task t_i, T_D and $minT_M$ are deadline and minimum makespan of application respectively.

Step-2 For each task t_i update *EST* and *EFT* using Eqs. (4) and (5), respectively.

Every task is set with a new deadline or target completion time by repeating the above steps for each task in a workflow.

4.4 Cluster-VM Mapping Algorithm

Different from the clusters and utility grid computing environment, in a cloud environment, as long as the service available, then the cloud scheduling approach can provide a time-slot to map the task to avail the service [37]. However, the cloud has inexhaustible resources, and it can offer VMs with different characteristics for users. But it is not always good to meet several optimization constraints. Therefore, in this section, we propose cluster-VM mapping to execute cluster tasks to maximize system reliability and minimize energy consumption. The steps to select a more appropriate VM for a cluster to execute its tasks as follows.

Step-1 For each task t_i (where $t_i \in cl_l$), calculate the optimal frequency for energy conservation on different available VMs using the Eq. (10).

Step-2 Calculate reliability of each task t_i (where $t_i \in cl_l$), on each VM using Eq. (15)

Step-3 Map the tasks of cluster cl_l to the highest reliable VM vm_k^{opt} to complete its execution; we denote it as $cl_l \rightarrow vm_k^{opt}$.

Step-4 If vm_k^{opt} is idle or if it executes in communication mode then scale down its operating frequency to its minimum i.e. $f_{op}^{i,k} = f_{min}^{i,k}$

We can map each cluster to the most suitable VM for energy conservation and maximize task reliability by repeating the above steps for all the clusters.

4.5 Slack Algorithm

Consumption of electrical energy has developed into one of the primary interests of the cloud-data centers. In the context of workflow application scheduling, there is some idle time slots associated with VMs (slack time) while executing non-critical tasks. We can redeem this slack associated with non-critical tasks by scaling the supply voltage and frequency of the task, to conserve energy [37, 39]. First, we need to estimate the latest start time (LST) of each task before we introduce the slacking algorithm. The LST of task t_i $LST(t_i)$ is specified as follows

$$LST(t_i) = \begin{cases} T_D - T(t_{entry}), & \text{for } t_i = t_{exit}, \\ min_{t_p \in child(t_i)}(LST(t_p) - T(t_p)), & \text{otherwise} \end{cases} \tag{21}$$

The slack time of task t_i is computed by

$$T_{slack}(t_i) = LST(t_i) - EST(t_i) \tag{22}$$

A critical task t_i has no room to reclaim i.e. $T_{slack}(t_i) = 0$, and for non-critical tasks $T_{slack}(t_i) > 0$. To reduce the frequency $f_{op}^{i,k}$ of a non critical task t_i to conserve energy, we used the following four steps:

Step-1 Calculate the slack time of the task t_i using Eq. (22) and mark all critical tasks as "defined" as no idle slot to reclaim.

Step-2 Select a task t_i to change the frequency, which has the longest T_{path}, and its parents are marked "defined", where T_{path} is the sum of the execution time of the tasks on the path from t_i to the "defined" task.

Step-3 Reduce task frequency to extend the execution time and save energy as follows

$$f_{op}^{i,k} = \left(f_{max}^{i,k}\right) \cdot \frac{T_{path}}{T_{path} + T_{slack}(t_i)} \tag{23}$$

Step-4 Update execution time of t_i and mark it "defined".

Repeat the above steps for entire DAG W until all tasks are marked "defined".

4.6 EERS Algorithm

We propose the EERS algorithm, which enables the cloud scheduler to get through less energy cost to complete an application and also maximizes system reliability while meeting the user-defined deadline. The EERS algorithm comprises five sub-algorithms which were introduced in the above sections. We schedule a task to a specified VM when all of its predecessors finishes i.e a task becomes schedulable when all of its predecessors complete their execution. When a current task completes execution then its successor tasks become schedulable. We specified a sub-deadline for every task before mapping it to the most appropriate VM. Thus, each task can be complete its execution within its target time, and the entire application can be completed within a specified deadline.

The pseudo-code for our EERS algorithm is presented in Fig. 3. Firstly, the EERS algorithm calls the *task rank calculation* algorithm at line 2 in Fig. 3, to realize a reasonable order of tasks to execute without loss of precedence constraints of workflow. At line 3 in Fig. 3, the *task clustering* algorithm is called to minimize the communication cost which reduces energy consumption. Then *sub-target time distribution* algorithm for energy conservation by decreasing the task frequency while meeting user-defined quality

Fig. 3 The pseudo-code of the EERS algorithm

```
The EERS Algorithm
1   BEGIN
2   Call the Task rank calculation algorithm
3   Call the Task clustering algorithm
4   Call the Sub-Target Time Distribution algorithm
5   while the workflow is not complete
6       Call the Cluster-VM mapping algorithm
7       if the task t_i is a non-critical task of workflow
8           then Call the Slack algorithm
9       end if
10  end while
11  END
```

parameter(deadline) in line 4. The *cluster-VM mapping* algorithm to select optimal VM to conserve energy and maximize task reliability is called at line 6 in Fig. 3. Finally, a *slack algorithm* is called at line 7 in Fig. 3 to reclaim the slack of non-critical tasks to further reduce the energy consumption. The clustering algorithm, task-VM mapping algorithm, and task slacking algorithm reduce energy consumption without compromising the performance.

5 Performance Evaluations

We discussed the performance of the EERS workflow scheduling technique in this section. By conducting a series of simulation runs, we evaluated our proposed algorithm. We assume that the cloud data center has DVFS enabled virtual machines (VMs) and every VM has its computing resources, and bandwidth is constant between VMs instances. The computing performance of VMs and other simulation parameters present in Table 2 and are self-defined. The other parameter such as failure rates are 10^{-5} to 10^{-7} failures/s as in [30]. For ease, we suppose that the frequency of VM is directly proportional to its computing performance, which is realizable from the experimental point of view. Moreover, every VM operates at different levels of the frequency with minimum and maximum thresholds, and the DVFS takes advantage of these frequency levels to scale the task's operational frequency to conserve energy.

We used WorkflowSim [40] is a toolkit to simulate the cloud environment, which permits the IaaS cloud to model and simulate. The IaaS cloud provides a virtualized computing environment (VMs) to execute workflow applications. We choose real-world scientific workflows such as *CyberShake* and *Montage* to evaluate our proposed EERS algorithm. *Cybershake* is characterized as a memory and data-intensive earthquake science application. This model was used in Sothern California to study Seismic Hazards. It used physics-based 3-D ground simulations to study the seismic wave propagation effects. *Montage* is an I/O-intensive astronomy application developed for scientific research. This toolkit enables astronomers for assembling sky images in Flexible Image Transport System (FITS) within custom mosaics. Usually, very large scale datasets used in scientific workflow applications. Each workflow has distinct topological structures. The topological structure of the

CyberShake and the *Montage* workflows are shown in Figs. 4 and 5 respectively. Moreover, each one has its data requirements and computational characteristics [15, 41].

For experiments, purpose, we assumed that each task's workload of the workflow ranges from 1000 to 3000 MIPS, and the output generated data volume ranges between 100 and 1000 GB. Due to the complex and massive data volume of workflow applications, the deadline for workflow is over 10 h [37].

In our experiments, the main focus is on reliability and energy consumption. These will change with the varying workloads (number of tasks in workflow) and a varying number of VMs to execute the selected workload. To expose the strength of our proposed scheduling algorithm on reducing energy consumption and assured reliability the simulation results were compared with other existing works. We perform three well-known scheduling approaches HEFT [11], EES [39], and REEWS [30] to compare with our algorithm. The HEFT approach is a prominent list-based heuristic for workflow application scheduling to optimizing the makespan. In every step, HEFT chooses the highest priority (rank value) task to assign a processor, which reduces the earliest finish time (EFT) with an insertion-based approach. The Enhanced Energy-efficient Scheduling (EES) algorithm is a HEFT-based approach to conserve energy while meeting the quality parameters. The fundamental idea of the EES method exploits the slack time on non-critical tasks and globally allocates them to minimize energy. The REEWS is a heuristic algorithm to maximize the application reliability and minimize energy consumption while meeting user-defined quality constraints. It outperformed RHEFT [18] in reliability and PALS [42] in energy saving. REEWS works in four stages: (1) task priority calculation to preserve dependencies; (2) task clustering to minimize communication cost; (3) distribution of target time (user-defined deadline); and (4) mapping the cluster to VM with appropriate frequency/voltage levels.

5.1 Performance Evaluation with Different Workloads

First, we evaluated the performance of our algorithm for different workloads i.e by varying the number of tasks as 30, 50, 75, 100, 150, and 200 on *CyberShake* and 75, 100, 125, 150, 175, and 200 on *Montage* real-world scientific workflows. The simulation results for

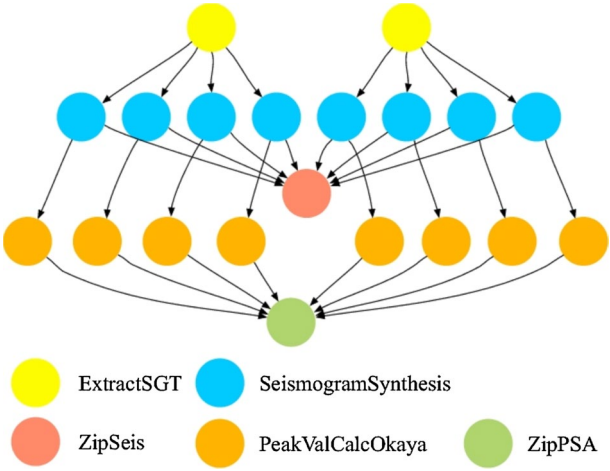

Fig. 4 CyberShake

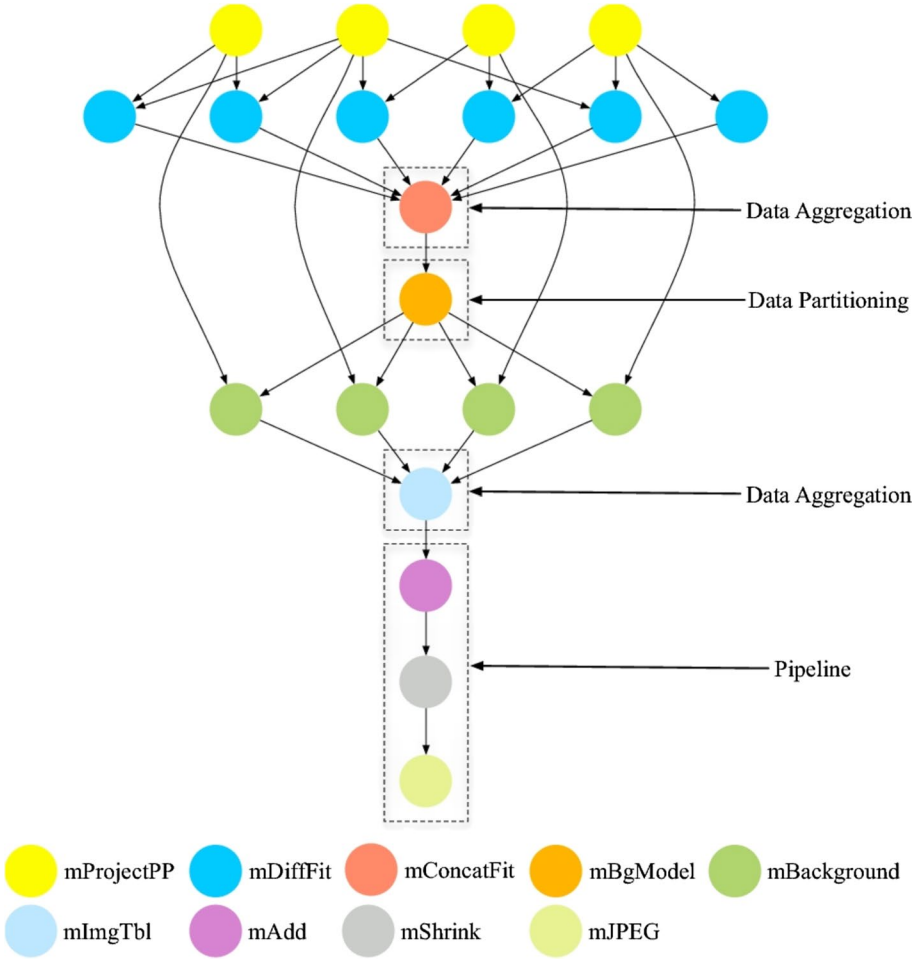

Fig. 5 Montage

Table 2 Simulation environment parameters

VM parameter	Value(s)
CPU frequency level (f_{max})	2.0–2.4 GHz
Computing capacity (MIPS)	1000–3000
RAM	512 MB
Bandwidth	1000 Mbps
Number of cores	1
Voltage supply (Vmax)	220 V

energy consumption and system reliability are depicted in Figs. 6 and 7 respectively. In energy objectives, our proposed EERS consistently reduced energy concerning the workloads. It saved more energy as compared to the HEFT, EES, REEWS as the fact that the

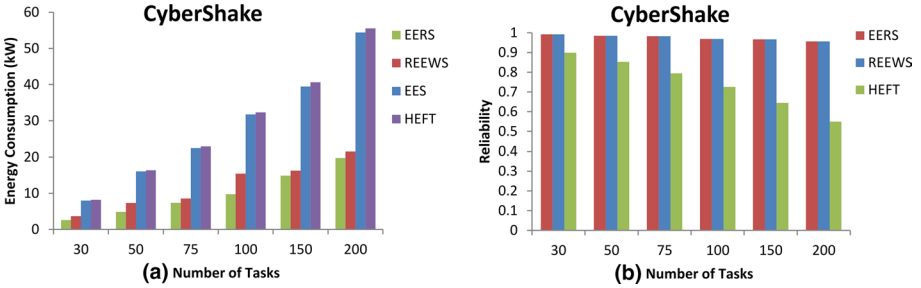

Fig. 6 **a** Energy consumption and **b** Reliability for various workloads on CyberShake Workflow

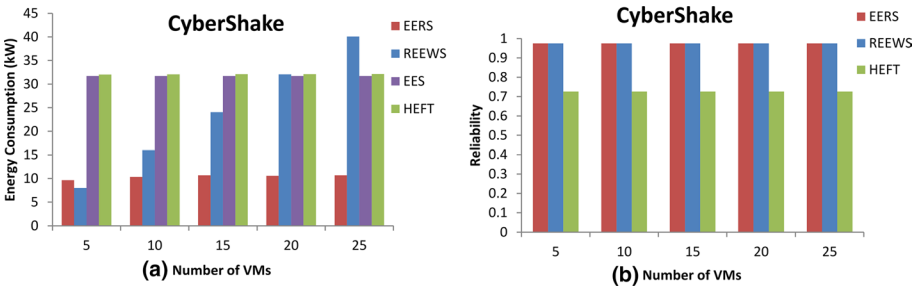

Fig. 7 **a** Energy consumption and **b** Reliability for diffrent number of VMs on CyberShake Workflow

EERS is efficient in allocating resources to the tasks in such a way to reduce energy consumption. Moreover, EERS efficient in clustering the tasks to reduce the communication costs and hence saves energy. Finally, the EERS used a task reclaiming approach to take the advantage of the DVFS techniques to reduce energy utilization by lowering the task's supply voltage and frequency. The REEWS algorithm is an efficient technique in maximizing the system reliability which outperformed the state-of-the-art techniques such as RHEFT and PALS in reliability objective. Our proposed EERS gives better reliability with HEFT and on par with REEWS but in every case, our proposed EERS approach consumed less energy compared with the other approaches.

5.2 Performance Evaluation with Different Number of VMs

Further, we evaluated the performance of the EERS approach for different numbers of VMs on both CyberShake and Montage workflows. We considered 5, 10, 15, 20, and 25 VMs on CyberShake and 15, 18, 20, 24, and 28 VMs on Montage. The simulation results for energy consumption and system reliability on CyberShake and Montage workflows are depicted in Figs. 8 and 9 respectively. Based on energy consumption and reliability the EERS algorithm efficient in selecting the number of VMs and as well as the type of VMs. Hence, in this case, our EERS algorithm saved more energy on both workflows compared to other approaches. With varying numbers of processors/VMs also our approach maintained good reliability this is because the EERS algorithm considers failure rate before selecting resources for mapping tasks to that resources.

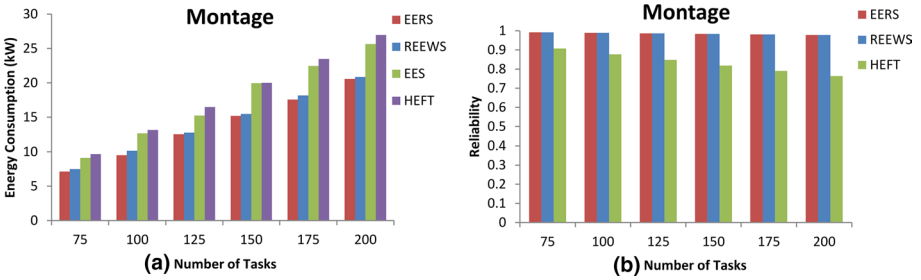

Fig. 8 **a** Energy consumption and **b** reliability for various workloads on Montage Workflow

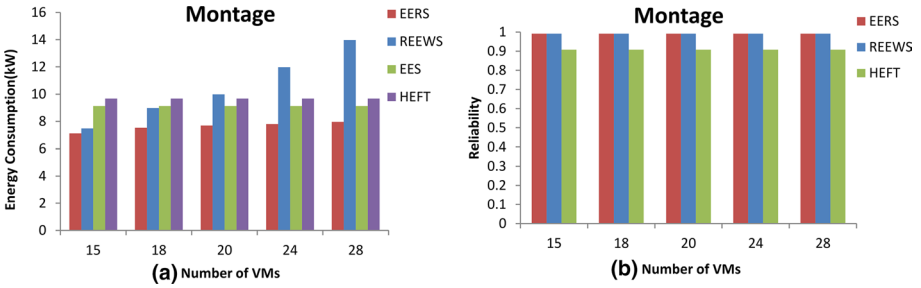

Fig. 9 **a** Energy consumption and **b** reliability for diffrent number of VMs on Montage Workflow

6 Conclusion

Recently, the need for energy conservation and maximizing system reliability has become important research. In this work, we considered scientific real-world workflows to schedule in the cloud environment. We present Energy Efficient and Reliability-Aware Scheduler (EERS) for scheduling workflow tasks in Cloud Computing to minimize energy consumption and maximize the application reliability while satisfying user-defined deadline constraints. Our proposed EERS approach comprise of five sub-algorithms (1) task rank calculation algorithm, (2) task clustering algorithm, (3) Sub-target time distribution algorithm (4) cluster-VM mapping algorithm, and (5) slack algorithm. We discussed our EERS performance in Sect. 5 by performed considerable simulation runs on the WorkflowSim toolkit and evaluated our algorithm on real-world scientific workflows CyberShake and montage for different numbers of virtual machines and different workloads. We compared the performance of our EERS algorithm with popular workflow scheduling techniques such as HEFT, EES, and REEWS. It was observed from the simulation experimental results that our proposed approach consumed less energy with maximizing the system reliability in all the cases. We can conclude that the time complexity of the proposed sub-algorithms is polynomial. Simulation experiments outcome reveal that our EERS approach surpasses other algorithms in both energy consumption and reliability.

As a subsequent work, we will examine the actual electricity costs for workflow tasks scheduling, and further, we will incorporate the monetary cost constraints and frequency-independent energy consumption.

Author Contributions An energy and reliability aware workflow scheduling algorithm, called EERS is proposed to reduce the application energy consumption while maximizing the system reliability. The proposed EERS approach consits of five sub-algorithms such as task rank calculation, task clustering Sub-target time distribution, cluster-VM mapping algorithm and slack algorithm to meet the stated objectives. The performance of proposed EERS approach evaluated through several numerical experiments. And the experimental results show that the proposed approach can achieve a significant amount of energy-saving, while maximizing the application reliability without regard to the diverse workflow structures.

Funding Not applicable.

Availability of Data and Material The data that support the findings of this study are openly available at https://codeload.github.com/pegasus-isi/WorkflowGenerator/zip/master/, and reference number [41].

Compliance with Ethical Standards

Conflict of interest We have no conflicts of interest associated with this publication.

Code Availability Custom code.

References

1. Microsoft-Azure. (2020). *What is cloud computing?*. https://azure.microsoft.com/en-in/overview/what-is-cloud-com-puting/.
2. Medara, R., Singh, R. S., Kumar, U. S., & Barfa, S. (2020). Energy efficient virtual machine consolidation using water wave optimization. In *2020 IEEE congress on evolutionary computation (CEC)*, pp. 1–7. IEEE.
3. Arroba, P., Moya, J. M., Ayala, J. L., & Buyya, R. (2017). Dynamic voltage and frequency scaling-aware dynamic consolidation of virtual machines for energy efficient cloud data centers. *Concurrency and Computation: Practice and Experience, 29*(10), e4067.
4. You, X., Li, Y., Zheng, M., Zhu, C., & Lifeng, Yu. (2017). A survey and taxonomy of energy efficiency relevant surveys in cloud-related environments. *IEEE Access, 5,* 14066–14078.
5. Rodriguez, M. A., & Buyya, R. (2017). A taxonomy and survey on scheduling algorithms for scientific workflows in IAAS cloud computing environments. *Concurrency and Computation: Practice and Experience, 29*(8), e4041.
6. Deelman, E., Gil, Y., & Zemankova, M. (2006). Nsf workshop on the challenges of scientific workflows, pp. 1–2.
7. Tang, X., Li, K., Li, R., & Veeravalli, B. (2010). Reliability-aware scheduling strategy for heterogeneous distributed computing systems. *Journal of Parallel and Distributed Computing, 70*(9), 941–952.
8. Zhou, G., Guo, B., Gao, X., Ning, W., & Yan, Y. (2014). Software analysis for transient faults: A review of recent methods. In J. S. Pan, V. Snasel, E. Corchado, A. Abraham, & S. L. Wang (Eds.), *Intelligent data analysis and its applications* (Vol. 2, pp. 575–581). Cham: Springer.
9. Zhou, X., Zhang, G., Sun, J., Zhou, J., Wei, T., & Shiyan, H. (2019). Minimizing cost and makespan for workflow scheduling in cloud using fuzzy dominance sort based heft. *Future Generation Computer Systems, 93,* 278–289.
10. Wu, H., Chen, X., Song, X., Zhang, C., & Guo, H. (2020). Scheduling large-scale scientific workflow on virtual machines with different numbers of vCPUs. *The Journal of Supercomputing, 77,* 1–32.
11. Topcuoglu, H., Hariri, S., & Min-you, W. (2002). Performance-effective and low-complexity task scheduling for heterogeneous computing. *IEEE Transactions on Parallel and Distributed Systems, 13*(3), 260–274.
12. Gupta, S., Agarwal, I., & Singh, R. S. (2019). Workflow scheduling using jaya algorithm in cloud. *Concurrency and Computation: Practice and Experience, 31*(17), e5251.
13. Tao, F., Feng, Y., Zhang, L., & Liao, T. W. (2014). CLPS-GA: A case library and pareto solution-based hybrid genetic algorithm for energy-aware cloud service scheduling. *Applied Soft Computing, 19,* 264–279.
14. Sellami, K., Ahmed-Nacer, M., Tiako, P. F., & Chelouah, R. (2013). Immune genetic algorithm for scheduling service workflows with QoS constraints in cloud computing. *South African Journal of Industrial Engineering, 24*(3), 68–82.

15. Rodriguez, M. A., & Buyya, R. (2014). Deadline based resource provisioning and scheduling algorithm for scientific workflows on clouds. *IEEE Transactions on Cloud Computing, 2*(2), 222–235.
16. Wang, S., Li, K., Mei, J., Xiao, G., & Li, K. (2017). A reliability-aware task scheduling algorithm based on replication on heterogeneous computing systems. *Journal of Grid Computing, 15*(1), 23–39.
17. Rehani, N., & Garg, R. (2017). Reliability-aware workflow scheduling using monte carlo failure estimation in cloud. In *Proceedings of international conference on communication and networks* (pp. 139–153). Springer, Berlin.
18. Dongarra, J. J., Jeannot, E., Saule, E., & Shi, Zhiao. (2007). Bi-objective scheduling algorithms for optimizing makespan and reliability on heterogeneous systems. In *Proceedings of the nineteenth annual ACM symposium on parallel algorithms and architectures* (pp. 280–288).
19. Mahato, D. R., & Singh, R. S. (2019). Load balanced scheduling and reliability modeling of grid transaction processing system using colored petri nets. *ISA Transactions, 84,* 225–236.
20. Mahato, D. P., & Singh, R. S. (2018). Maximizing availability for task scheduling in on-demand computing-based transaction processing system using ant colony optimization. *Concurrency and Computation: Practice and Experience, 30*(11), e4405.
21. Mahato, D. P., & Singh, R. S. (2017a). Balanced task allocation in the on-demand computing-based transaction processing system using social spider optimization. *Concurrency and Computation: Practice and Experience, 29*(18), e4214.
22. Mahato, D. P., & Singh, R. S. (2017b). Load balanced transaction scheduling using honey bee optimization considering performability in on-demand computing system. *Concurrency and Computation: Practice and Experience, 29*(21), e4253.
23. Chia-Ming, W., Chang, R.-S., & Chan, H.-Y. (2014). A green energy-efficient scheduling algorithm using the DVFS technique for cloud datacenters. *Future Generation Computer Systems, 37,* 141–147.
24. Stavrinides,G. L. & Karatza, H. D. (2018). Energy-aware scheduling of real-time workflow applications in clouds utilizing DVFS and approximate computations. In *2018 IEEE 6th international conference on future internet of things and cloud (FiCloud)*, (pp. 33–40). IEEE.
25. Stavrinides, G. L., & Karatza, H. D. (2019). An energy-efficient, QoS-aware and cost-effective scheduling approach for real-time workflow applications in cloud computing systems utilizing DVFS and approximate computations. *Future Generation Computer Systems, 96,* 216–226.
26. Safari, M., & Khorsand, R. (2018). Energy-aware scheduling algorithm for time-constrained workflow tasks in dvfs-enabled cloud environment. *Simulation Modelling Practice and Theory, 87,* 311–326.
27. Fernández-Cerero, D., Jakóbik, A., Fernández-Montes, A., & Kołodziej, J. (2019). Game-score: Game-based energy-aware cloud scheduler and simulator for computational clouds. *Simulation Modelling Practice and Theory, 93,* 3–20.
28. Li, C., Tang, J., Ma, T., Yang, X., & Luo, Y. (2020). Load balance based workflow job scheduling algorithm in distributed cloud. *Journal of Network and Computer Applications, 152,* 102518.
29. Abazari, F., Analoui, M., Takabi, H., & Song, F. (2019). Mows: Multi-objective workflow scheduling in cloud computing based on heuristic algorithm. *Simulation Modelling Practice and Theory, 93,* 119–132.
30. Garg, R., Mittal, M., et al. (2019). Reliability and energy efficient workflow scheduling in cloud environment. *Cluster Computing, 22*(4), 1283–1297.
31. Burd, T. D., & Brodersen, R. W. (1995). Energy efficient CMOS microprocessor design. In *Proceedings of the twenty-eighth annual Hawaii international conference on system sciences* (Vol. 1, pp. 288–297). IEEE.
32. Mishra, S., Singh, N. K., & Rousseau, V. (2015). *System on chip interfaces for low power design*. Burlington: Morgan Kaufmann.
33. Zhu, D., Melhem, R., & Mossé, D. (2004). The effects of energy management on reliability in real-time embedded systems. In *IEEE/ACM international conference on computer aided design, 2004. ICCAD-2004* (pp. 35–40). IEEE.
34. Zhang, L., Li, K., Yuming, X., Mei, J., Zhang, F., & Li, K. (2015). Maximizing reliability with energy conservation for parallel task scheduling in a heterogeneous cluster. *Inf. Sci., 319,* 113–131.
35. Cao, F., & Zhu, M. M. (2013). Energy-aware workflow job scheduling for green clouds. In *2013 IEEE international conference on green computing and communications and IEEE internet of things and IEEE cyber, physical and social computing* (pp 232–239). IEEE.
36. Buyya, R., Beloglazov, A., & Abawajy, J. (2010). Energy-efficient management of data center resources for cloud computing: A vision, architectural elements, and open challenges. In *Proceedings of the international conference on parallel and distributed processing techniques and applications, PDPTA 2010, Las Vegas, USA* (p. 2010).

37. Li, Z., Ge, J., Haiyang, H., Song, W., Hao, H., & Luo, B. (2015). Cost and energy aware scheduling algorithm for scientific workflows with deadline constraint in clouds. *IEEE Transactions on Services Computing, 11*(4), 713–726.
38. Yuming, X., Li, K., He, L., Zhang, L., & Li, K. (2014). A hybrid chemical reaction optimization scheme for task scheduling on heterogeneous computing systems. *IEEE Transactions on Parallel and Distributed Systems, 26*(12), 3208–3222.
39. Huang, Q., Su, S., Li, J., Xu, P., Shuang, K., & Huang, X. (2012). Enhanced energy-efficient scheduling for parallel applications in cloud. In *2012 12th IEEE/ACM international symposium on cluster, cloud and grid computing (ccgrid 2012)* (pp. 781–786). IEEE.
40. Chen, W., & Deelman, E. Workflowsim (2012). A toolkit for simulating scientific workflows in distributed environments. In *2012 IEEE 8th international conference on E-science* (pp. 1–8). IEEE.
41. Silva, R. F. D., Chen, W., Juve, G., Vahi, K., & Deelman, E. (2014). Community resources for enabling research in distributed scientific workflows. In *2014 IEEE 10th international conference on e-science* (vol. 1, pp. 177–184). IEEE.
42. Wang, L., Khan, S. U., Chen, D., KołOdziej, J., Ranjan, R., Xu, C.-Z., & Zomaya, A. (2013). Energy-aware parallel task scheduling in a cluster. *Future Generation Computer Systems, 29*(7), 1661–1670.

Rambabu Medara received his B.Tech. and M.Tech. degrees in Computer Science and Engineering from Andhra University in 2004 and JNT University in 2010, respectively. He is a Ph.D. candidate in the Department of Computer Science and Engineering at the Indian Institute of Technology BHU, Varanasi, Uttar Pradesh, India. His research interests include Cloud Computing, Scheduling, and resource management in the datacenter. He is a student member of the IEEE.

Ravi Shankar Singh received his Ph.D. degree in Computer Science and Engineering from the Indian Institute of Technology (BHU), Varanasi in the year 2010. He is currently working as an Associate Professor in the Department of Computer Science and Engineering, Indian Institute of Technology (BHU), Varanasi. He joined the Department of Computer Science and Engineering, Indian Institute of Technology (BHU), in 2004. He has held various positions at the Indian Institute of Technology, Varanasi. His mainstream research interests include design and analysis of algorithms, Parallel and distributed computing, cluster and grid computing, and High-Performance Computing. He is a senior member of IEEE and ACM.