Reliability aware green workflow scheduling using ε -fuzzy dominance in cloud

Rama Rani¹ · Ritu Garg¹

Received: 11 June 2021 / Accepted: 3 December 2021 / Published online: 20 December 2021
© The Author(s) 2021

Abstract

The enormous energy consumed by cloud data centers (CDCs) increases the carbon footprints, operational cost and decreases the system reliability, so it becomes a great challenge for CDCs providers. Dynamic voltage and frequency scaling (DVFS) is an efficient approach for energy efficiency, which reduces the operating frequency, and supply voltage of the processor during the task's execution. Recent research shows that scaling of the supply voltage and operating frequency has negative impact on the system's reliability as it increases transient fault rate of the resources. Thus, the system's reliability and the energy consumption are two prime concerns in a cloud computing environment that requires attention. Most workflow scheduling algorithms in literature do not consider energy and reliability simultaneously. In this paper, we proposed the ε -fuzzy dominance based reliable green workflow scheduling (FDRGS) algorithm, which optimizes the application's reliability and energy consumption simultaneously using the ε -fuzzy dominance mechanism. The simulation results obtained using fast Fourier transform (FFT) and gaussian elimination (GE) task graphs manifest that our scheduling algorithm is more efficient in optimizing energy consumption and lifetime system's reliability jointly than several widely used algorithms. The proposed algorithm will help scientists and engineers for further insight into future research in the area of cloud.

Keywords Workflow scheduling · Cloud computing · ε -fuzzy dominance · Energy consumption · DVFS · Reliability

Introduction

Cloud computing is a popular computing model that gives scalable and on-demand services to the users using the internet. In Cisco [1], by the year 2021, 94% of computing is done by the cloud computing systems. Due to the increased popularity of cloud services, different communities and enterprises prefer cloud environment to deploy scientific workflow applications appropriate computational resources to workflow tasks to accomplish the task's execution while satisfying the user-defined constraints. Although cloud computing has great benefits such as minimizing operational costs, it suffer from high-energy consumption problem. As the computational resources increases in cloud environment, energy consumption has become an essential concern [2]. For example,

a heterogeneous computing system, namely Tianhe-2 (world fastest supercomputer) situated at National Supercomputer Center in China (Guangzhou), and its power consumption was 17,808 kW in July 13, 2015 [3]. For economic development needs, the National Supercomputer Center concentrates on power efficacy rather than speed, since Supercomputers consume a lot of power, although they are beneficial to give a high-performance solution. Further, Moore's law [4] states that the transistor's count in integrated circuits is increased by two times in every 2 years. Therefore, this leads to a rise in the energy consumption in data centers. From the above investigations, how to obtain desirable green computing environment is an important issue in cloud computing.

Further, when the energy usage of the cloud system is increased, it creates negative impact on the system's reliability, causes carbon footprints and increases the operating cost significantly. Moreover, inefficient utilization of computing resources raises the energy consumption of workflow task's execution [5]. To tackle the problem of energy consumption, different hardware technologies [6] exist like powerful energy-efficient monitors, multiple CPU cores, low-power efficient microprocessors, and cache memory. Moreover,

✉ Rama Rani
rama_6170044@nitkkr.ac.in

Ritu Garg
ritu.59@nitkkr.ac.in

¹ Computer Engineering Department, National Institute of Technology, Kurukshetra, India

DVFS technique is widely used by researchers to lessen energy consumption by scaling down voltage/frequency during the execution [7]. However, research work [8], signifies that scaling the chip voltage dynamically increases the transient failure of resources, which affects the reliability of the application execution. Additionally, keeping all the applications executing on high-performance systems for a long duration will increase the temperature of the cloud computing systems, which further diminishes the reliability and availability. Thus, energy and reliability are conflicting factors.

In the recent past, high-performance resources attracted a number of researchers to work on workflow scheduling problem in cloud environment. Many authors developed the workflow scheduling algorithms, namely constrained earliest finish time algorithm (CEFT) [9], heterogeneous earliest finish time algorithm (HEFT) [10] and critical path on a processor (CPOP) algorithm [10] considering the single objective of makespan/execution time only. Moreover, considerable research efforts have been done in workflow scheduling with the aim to optimize the system's reliability and makespan. For example in paper [11], the authors proposed the *Reliable-HEFT* (RHEFT) algorithm to address the maximization of reliability along with makespan minimization. Authors in paper [12] proposed a reliability dynamic level scheduling (RDLS) algorithm concentrating on the reliability of the computing resources while minimizing the execution time. In paper [13], the authors proposed hierarchical reliability-driven scheduling (HRDS) heuristic to optimize the reliability issues in grid environment. Further, in paper [14, 15], authors considered reliability and energy efficiency parameters for workflow scheduling, but the same is applied in the area of embedded systems. In paper [16], authors developed the scheduling heuristics for workflow applications which uses DVFS mechanism to lessen energy consumption in a homogeneous cluster computing. However, optimizing reliability and energy efficiency jointly in a heterogeneous cloud environment is still to be explored more.

Since energy consumption and system's reliability conflicts each other, thus workflow scheduling in cloud computing is a multi-objective problem. In multi-objective problems, there exists no single solution, which is good for all the considered objectives. Thus, in multi-objective optimization, many latest researches [17, 18] are providing the Pareto optimal solution set rather than single best solution. Pareto optimality concept is to obtain multiple non-dominated solutions. In paper [19], authors used the well-known meta-heuristic i.e. PSO with non-domination sorting (NSPSO) in order to achieve the Pareto set of optimal solutions. Further, in paper [20], authors proposed the Discrete PSO using ϵ -fuzzy dominance mechanism that quantify the relative fitness of non-dominated solutions to achieve better trade-off front. However, generation of Pareto set with a meta-heuristic is very time-consuming. Further, it is not easy for the decision maker

to select one solution from the set of multiple solutions. Thus, in this research work, we are applying the heuristic technique in spite of meta-heuristic to find out the single best solution using ϵ -fuzzy dominance with very small computation time.

Our work in this paper contributes to reliability-aware energy-efficient scheduling on heterogeneous cloud environments. The above-mentioned works focus on optimization of either energy consumption or reliability but to the best of my knowledge very less research work focused on the joint optimization of energy consumption and reliability. However, we proposed an algorithm that focuses on maximizing the system's reliability and jointly minimizing energy consumption for workflow applications under precedence constraints. We designed a new heuristic scheduling algorithm named as ϵ -fuzzy dominance based reliable green workflow scheduling (FDRGS) for workflow applications. The algorithm works in two phases. Firstly, the rank is assigned to all tasks of the workflow and then those tasks that have the highest rank are assigned the highest priority. Secondly, the best processor for every task is selected by considering multiple objectives i.e. energy consumption and reliability of task's execution. ϵ -Fuzzy dominance sorting is applied to obtain best trade-off solutions based on the fuzzy dominance values. This procedure is helpful to find a single best-compromised solution by taking the relative fitness of the trade-off solutions with less overhead. Further, the accurate frequency level of the processor is chosen to lower the energy consumption and maximize the reliability of the application tasks using DVFS.

The primary contributions of the paper are as follows:

- In this paper, workflow scheduling application is considered to simultaneously optimize energy consumption and reliability for executing workflow applications in a cloud environment.
- We proposed ϵ -fuzzy dominance based reliable green workflow scheduling (FDRGS) algorithm in cloud environment. The proposed algorithm obtains the single best solution by taking into account ϵ -fuzzy dominance values (the relative fitness) of trade-off solutions that too with small computation overhead by pruning the multiple trade-off solutions.
- Further, to minimize the energy consumption, we have applied a DVFS technique that lowers the frequency/voltage of the processor to save energy. Moreover, fault rate of the computing system is affected by the operating frequency/voltage level. If the system is operated at minimum frequency; it does not mean that the occurrence of fault rate is less. Fault arrival rate depends on the operating frequency of computing nodes. Hence, our objective is to select the correct/appropriate operating frequency level to achieve maximum reliability and minimum energy consumption.

- Extensive simulation analysis is carried out to evaluate proposed FDRGS algorithm. We compared our algorithm with energy-conscious scheduling heuristic ECS [21], reliability HEFT (RHEFT) [11] and HEFT [10]. Further, statistical test has been obtained to validate the results. Results manifest that the proposed FDRGS algorithm performs better than other compared algorithms.

The remaining part of the paper is summarized as follows. The next section mentions related works and the following section shows the models used in current work, consisting of system model, scheduling model, energy and reliability model. The next section presents the proposed algorithm. The next section shows the simulation evaluation of the FDRGS approach. Last section provides the conclusion and future work.

Related work

Recently, numerous research works [11, 12, 22–29] in cloud environment have been proposed to achieve efficient performance, energy consumption and reliability. The authors in paper [11] proposed the task scheduling algorithm for heterogeneous systems to optimize makespan and reliability. Authors in paper [12] developed bi-objective scheduling algorithm considering reliability and execution time in heterogeneous systems. In paper [30], a scheduling algorithm has been proposed considering two objectives i.e. time and reliability simultaneously using weighted sum approach. In paper [31], an analytical resource allocation model is proposed to achieve system reliability considering application constraint and resource constraint. Application constraint takes care of task precedence and resource constraint is confined to storage and memory usage. In paper [32], authors proposed a reliable scheduling solution for DAG application considering timing constraints with the aim to maximize the reliability. In paper [33], authors proposed the reliability-driven architecture for task scheduling depending on the optimal reliable communication path. For assigning priority to the tasks, the authors considered the additional parameter of reliability overhead and proposed the scheduling technique with duplication for achieving high reliability for precedence-constrained tasks. Further, in paper [34], authors proposed a scheduling algorithm to optimize the makespan and reliability of tasks without taking care of energy consumption based on a genetic optimization technique. The authors introduced a reliability model and load balancing technique in [35] by using Colored Petri Nets (CPN). The authors in paper [36] proposed a heuristic that minimizes storage and network resource usage while guarantee service reliability. Above-mentioned research works focus on

the optimization of reliability along with other objectives of execution time in heterogeneous systems.

Along with the reliability of cloud computing resources, efficient consumption of energy has become a big challenge for the research community in the cloud environment, not only for reducing budget but also for increasing the reliability of the system. Cloud data centers use a lot of power and cause irreversible and serious damage to the global environment due to emission of carbon dioxide. Many scheduling algorithms [15, 37] have been proposed with the objectives of energy optimization at the resource level or cluster level [38]. The authors in [39] proposed an energy-efficient approach in a DVFS-enabled cluster to schedule workflow applications by utilizing slack time corresponding to non-critical jobs. However, in cloud systems, energy optimization along with reliability is still needed to be explored.

Further, there exist few energy efficient scheduling techniques for the sustainable computing along with the other important objectives of minimizing the execution cost, completion time etc. In heterogeneous systems, authors in paper [40] concentrate on the problem of minimizing the completion time while satisfying the energy constraint for DAG applications. However, authors in paper [41] focused on reducing energy consumption while satisfying deadline constraint as per service level agreement (SLA) by reclaiming slack time. The authors in paper [25], proposed two algorithms considering time and energy parameters for task scheduling, and then they combined these two algorithms and proposed a multi-resource scheduling algorithm to show the tradeoff between energy and performance. They considered the probability parameter α to control the cloud system's energy and performance. To show the better tradeoff, probability parameter values were changed dynamically according to the existing workload. In paper [22], authors proposed a task scheduling heuristic ECOTS by considering server power efficiency, resource requirements to optimize the energy consumption with minimal cost of performance in cloud environment. ECOTS provides power efficiency under different workloads. In paper [23], authors present an algorithm named as cost and energy efficient scheduling (CEAS) to reduce the workflow cost and minimize the energy consumption under deadline constraint. In paper [24] authors proposed an energy aware task scheduling (EATS) algorithms which minimizes energy consumption of the workflow application considering DVFS technique. EATS algorithm finishes task's execution within deadline to meet the QoS constraint. In above-mentioned algorithms, researchers considered one objective as the primary objectives while taking the other objective as the constraint to optimize the conflicting objectives and provide the unique solution.

On the other hand, there exist some scheduling algorithms considering multiple objectives simultaneously and provide the Pareto solutions set instead of the single best

solution. Authors in paper [42], proposed a multi-objective list scheduling algorithm (MOLS) for scientific workflows taking into account various objectives namely energy consumption, makespan, cost, and reliability. They maximize the distance from different constraints corresponding to each objective for dominating solutions. MOLS is used to find solution that dominates/converges to constraints i.e. defined by application user for multi-objective. MOLS algorithms find the dominant solution from the Pareto set of solutions. The authors in paper [43] proposed an extension of HEFT algorithm that is multi-objective HEFT (MOHEFT) which provides multiple solutions considering makespan and energy efficiency and provides Pareto solution set.

From the above-mentioned works, it is observed that most of the researchers considered either energy consumption or reliability with the execution performance of the application. However, considering both parameters reliability and energy consumption simultaneously opens new opportunities for researchers in cloud systems. Hence, in this paper, we proposed an energy efficient scheduling algorithm namely ε -Fuzzy dominance based reliable green workflow scheduling (FDRGS) that uses ε -fuzzy dominance procedure to find the best solution, which ensures the maximization of the reliability and minimization of energy consumption simultaneously. The proposed algorithm is very advantageous to explore the tradeoff between two conflicting objectives.

Mathematical formulation

This section introduces the cloud computing model, workflow model, task scheduling model, reliability model and the energy model considered. Table 1 provides some important notations used in this research work.

Cloud computing model

Cloud infrastructure consists of a number of physical machines (s) that are DVFS enabled and support maximum L frequency levels ($f_1, f_2, \dots, f_L$) and the respective voltages. All the physical machines (PM) are further virtualized into the different virtual machines (VMs) by using hypervisor.

Figure 1 shows the cloud computing framework for the proposed FDRGS algorithm. A *cloud user* sends the request to the *Cloud Manager* by considering workflow application graphs as input. After a request is approved, the Cloud Manager forwards workflow application to *Scheduler*. The main purpose of *Scheduler* is to generate the best scheduling solution for the user's application. *Reliability predictor* monitors a resource constantly and evaluates its state by analyzing its log. On the basis of history log information, *Reliability Predictor* predicts the failure characteristics of each physical machine in order to provide reliable resource allocation

Table 1 Some important notations used throughout the paper

Symbol	Definition
G	Directed acyclic graph (DAG)
T	Task set in the DAG ($T = t_1, t_2, \dots, t_n$)
E	Edge set in the application graph ($e_{jk} \in E, 1 \leq j, k \leq n, j \neq k$)
$w(t_j)$	Computation cost of task t_j
$CT(t_j, t_k)$	Communication time of the edges between task t_j and t_k
PM	Physical machines/processors ($PM_r, 1 \leq r \leq s$)
CPI	Cycles/instruction for particular PM
$f_{r,l}$	Operating frequency level of processor ($PM_r, 1 \leq l \leq L$)
$EST(t_j)$	Earliest start time of task t_j
$EFT(t_j)$	Earliest finish time of task t_j
$pred(t_j)$	Predecessor set of task t_j
$succ(t_j)$	Successor set of task t_j
t_{entry}	Entry task
t_{exit}	Exit task
$\xi(t_j, f_{r,l})$	Energy consumption for the particular task t_j executing on r th selected physical machine operating at l th frequency ($f_{r,l}$)
$RF(t_j, f_{r,l})$	Reliability exponential factor

Fig. 1 Cloud computing framework for FDRGS algorithm

for the execution of tasks. Further, *DVFS Manager* facilitates the scheduler with the saving in energy consumption characteristics of the resources and allows the process to execute at selected voltage level. Finally, *Scheduler* generates the efficient schedule and passes it to the *Cloud Manager* that is responsible for allocating tasks to different physical machines as per scheduling sequence.

Workflow model

In this paper, deterministic workflow model is considered that is represented as directed acyclic graph (DAG). In our work, the application consists of a bag of tasks with depen-

dependency/precedence constraints. A workflow application is represented by DAG ($G = \{T, ET, E, CT\}$) where

- T signifies the vertex set, represents n tasks in graph G . Each task $t_j \in T$ represents the task that can execute on available computing resources. $\text{pred}(t_j)$ is the immediate predecessor of task t_j and $\text{succ}(t_j)$ is the immediate successor of task t_j . Entry task (t_{entry}) is a task for which $\text{pred}(t_j) = \emptyset$ and exit task (t_{exit}) is a task for which $\text{succ}(t_j) = \emptyset$. The weight $w(t_j)$ allocated to task t_j denotes the computation cost i.e. number of instructions to be executed corresponding to particular task (in MI).
- $ET(t_j)$ denotes the execution time of task t_j on any physical machine. Because of heterogeneity, task has different execution time on each physical machine.
- E represents edges in the application graph where $e_{j,k} = (t_j, t_k) \in E$ represents the dependency edge from task t_j to task t_k . The weight $w(e_{j,k})$ assigned to edge $(e_{j,k})$ signifies the data transferred (in MB) from task t_j to t_k .
- Communication time/cost corresponding to edge $e_{j,k}$ is represented by $CT(t_j, t_k)$ and it is dependent on the amount of data transfer from task t_j to t_k when executed on different PMs. Communication time ($CT(t_j, t_k) = 0$) when two tasks are assigned on the same physical machine.

Workflow scheduling model

We consider the problem of workflow scheduling in cloud environment with the aim to reduce the energy consumption and maximize the reliability for the workflow application. We firstly define the parameters like: communication time ($CT(t_j, t_k)$), earliest start time ($EST(t_j)$) and earliest finish time ($EFT(t_j)$). Communication time ($CT(t_j, t_k)$) between two precedence constraint tasks is determined by communicated data from task t_j to t_k represented by $w(e_{j,k})$ and the communication bandwidth ($BW(PM_r, PM_s)$) among the chosen computational resources where $BW(PM_r, PM_s)$ represents the bandwidth (in Mbps) between physical machines PM_r and PM_s . So,

$$CT(t_j, t_k) = \begin{cases} \frac{w(e_{j,k})}{BW(PM_r, PM_s)}, & PM_r \neq PM_s \\ 0, & PM_r = PM_s \end{cases} \quad (1)$$

Execution time $ET(t_j)$ of task t_j on any PM at maximum frequency level ($f_{r,L}$) is as:

$$ET(t_j) = \frac{w(t_j) \cdot CPI}{f_{r,L}}, \quad (2)$$

where $f_{r,L}$ denotes maximum frequency level of CPU during the task's execution.

Estimated start time ($EST(t_j)$) of a task is defined as the time when the task starts its execution. Due to heterogeneity

of PM's, execution time is different for every PM. $EST(t_j)$ is equal to communication time $CT(t_p, t_j)$ from the predecessor task to current task and finish time of their predecessors on the assigned physical machines as follows:

$$EST(t_j) = \begin{cases} 0 & t_j = t_{\text{entry}} \\ \max_{t_p \in \text{pred}(t_j)} \{EST(t_p) + ET(t_p) + CT(t_p, t_j)\} & \text{otherwise} \end{cases} \quad (3)$$

Estimated finish time ($EFT(t_j)$) of a task t_j is the time at which task finish its execution on any PM. It is the summation of estimated start time ($EST(t_j)$) and execution time ($ET(t_j)$) and is given as:

$$EFT(t_j) = EST(t_j) + ET(t_j). \quad (4)$$

Power model

The power model provides mathematical modeling for calculating the energy consumption of a cloud computing environment. Power consumption is primarily occurred due to memory, CPU and networks. Out of these, CPU consumes the significant part of energy i.e. of two types: static and dynamic. Thus, the system power [44] is calculated as follows:

$$P = P^s + k(P^{\text{ind}} + P^d), \quad (5)$$

where P^s represents static power consumption and (P^{ind}, P^d) represents the frequency independent and frequency dependent dynamic power consumption. P^{ind} is independent of system operational frequency and its value is going to be decreased by keeping the system power in standby state. The k represents the system state indicating whether active power consumption has occurred or not. If $k = 1$, the system is in active mode and $k = 0$ signifies that the system is in snooze

Static power is consumed during the idle state of the system. This power keeps the memory in doze mode and clock in ready state. This keeps the power consumption low down by turning off the systems. The major contributing factor of power consumption is dynamic power consumption. Therefore, we mainly concentrate on the minimization of dynamic power using technique such as DVFS in which the voltage and frequency of the processor is dynamically scaled at run time to reduce dynamic power.

Further, dynamic power consumption (P^d) is calculated as follows:

$$P^d = C^{\text{eff}} v_{dd}^2 f_l, \quad (6)$$

where C^{eff} represents effective loading capacitance. v_{dd} represents the operating voltage and $v_{dd} \propto f_l$. When the workflow application executes using lower frequency levels, then supply voltage is linearly reduced. Thus, P^d is:

$$P^d = C^{\text{eff}} f_l^3. \quad (7)$$

We know, Energy is calculated by the product of power and time and can be calculated by using $\xi = P \times T$. Finally, the energy consumption ($\xi(t_j, f_{r,l})$) for the particular task t_j executing on r^{th} selected physical machine operating at frequency (f_l) is calculated as follows:

$$\begin{aligned} \xi(t_j, f_{r,l}) &= (P^{\text{ind}_j} + C^{\text{eff}} f_{r,l}^3) \frac{\text{ET}(t_j)}{f_{r,l}} \\ &= P^{\text{ind}_j} \frac{\text{ET}(t_j)}{f_{r,l}} + C^{\text{eff}} \text{ET}(t_j) f_{r,l}^2. \end{aligned} \quad (8)$$

In order to reduce the energy consumption, energy efficient frequency f_{ee} [15] is to be considered which can be calculated by performing differentiation of Eq. (8) with respect of f and is calculated as:

$$f_{ee} = \sqrt[3]{P^{\text{ind}}/2C^{\text{eff}}}. \quad (9)$$

Finally, the total energy consumption (ξ_{tot}) of cloud system due to workflow application tasks is calculated by summing up the energy consumption of individual tasks as follows:

$$\xi_{\text{tot}} = \sum_{j=1}^n \xi(t_j, f_{r,l}) \quad \text{Objective 1.} \quad (10)$$

Reliability model

At the time of application execution, different types of faults occur such as permanent and transient because of many reasons, like hardware malfunction, interference occur due to external devices, software bugs and devices exposed to extreme temperatures. It is very difficult to avoid them. However, occurrence of transient faults is high during the task's execution and can be taken care. Thus, in the current study, we concentrate on the transient faults only. Poisson distribution is advantageous to model the transient fault with arrival rate λ [8, 45]. The arrival rate of the transient fault is dependent upon the corresponding voltage (v_{dd}) and operating frequency ($f_{r,l}$). When the value of $f_{r,l}$ decreases then the fault rate is exponentially increased and it is important for the energy conservation purpose. Considering the exponential model for voltage scaling [15], the fault of the processor increases as the supply voltage decreases. At minimum frequency (f_{min}), the fault rate is max i.e. $\lambda_{\text{max}} = \lambda_0 10^d$ where positive constant d signifies fault rate dependency. When the system performs their execution at frequency f_l , the fault rate relation is:

$$\lambda(f_{r,l}) = \lambda_0 10^{\frac{d(1-f_{r,l})}{1-f_{\text{min}}}}, \quad (11)$$

where λ_0 is the initial value of the average fault rate at f_L .

Further, the system reliability is probability of executing the task completely without failure. Poisson distribution is used by transient fault [15, 46], then reliability of a task t_j with execution time $\text{ET}(t_j)$, is given as:

$$\text{Reliability}(t_j, f_{r,l}) = e^{-\lambda(f_{r,l}) \frac{\text{ET}(t_j)}{f_{r,l}}}. \quad (12)$$

Finally, total system's reliability which is the probability of executing n number of tasks corresponding to directed acyclic graph G without failure and is as follows:

$$\text{Reliability}_{\text{system}}(G) = \prod_{j=1}^n \text{Reliability}_{t_j}(f_{r,l}). \quad (13)$$

In order to optimize (maximize) the reliability, we have to minimize the exponential factor (as per (Eq. 12)) represented as reliability exponential factor ($\text{RF}(t_j, f_{r,l})$) which is given by

$$\text{RF}(t_j, f_{r,l}) = \lambda(f_{r,l}) \frac{\text{ET}(t_j)}{f_{r,l}}. \quad (14)$$

Thus, reliability maximization is formulated by minimizing the reliability exponential factor, which is as follows:

$$\begin{aligned} &\text{minimize}(\text{Reliability exponential factor}(S)) \\ &= \sum_{j=1}^n \text{RF}(t_j, f_{r,l}) \quad \text{Objective 2.} \end{aligned} \quad (15)$$

Problem statement

In this paper, we addressed the problem of efficient task scheduling corresponding to workflow application onto the dynamic set of cloud computing resources with the aim to jointly optimize multiple conflicting objectives such as lessen the energy consumption of the system and enhancing the system's reliability.

Proposed algorithm

This section presents the proposed ε -fuzzy dominance based reliable green workflow scheduling (FDRGS) algorithm in cloud environment. Our problem of workflow scheduling is multi-objective (MOP) where we are jointly optimizing two conflicting objectives of energy consumption and reliability. Our algorithm works in two phases. Firstly, the rank is assigned to all tasks of the workflow application, and then the tasks which are having the highest rank are assigned the highest priority. Secondly, selection of the most appropriate

processor for every task is done. Fuzzy dominance sorting mechanism is used to find the best trade-off solution based on the fuzzy dominance values. The proposed algorithm obtains the single best compromised solution by taking the relative fitness of the trade-off solutions with less overhead. Further, DVFS approach is applied to optimize the energy consumption along with reliability of the task by choosing accurate frequency level of the processor.

Fuzzy dominance sort

Normally, in MOP, there does not exist a single solution i.e. best in terms of all the conflicting objectives. Thus, Pareto optimality concept is used to obtain the multiple solutions in multi-objective problems. In Pareto-optimal set of solutions, every non-dominating solution is considered to be good because the concept of non-dominance does not determine the point by which one solution is dominating the other solution. As a result, a new metric called ϵ -fuzzy dominance [47] is introduced which quantify the relative fitness of solutions that provides the more crispy definition of dominance. Thus, it leads to fast convergence towards the optimal set of solutions.

Let the MOP problem contains M number of objectives functions $g_k(s)$ where $k = 1, 2, \dots, M$ and let $\sigma \subset K^\partial$ where ∂ is the dimensions and it is the solution set that consists of all possible solution space.

Some important definitions of Fuzzy dominance can be stated as follows:

- **Fuzzy k -dominance to a solution:** there exist a increasing function $\mu_{\text{dom}}^k(\cdot)$ and its values lies between 0 and 1. Solution $x \in \sigma$ is k -dominated solution to $y \in \sigma$ iff $g_k(x) < g_k(y)$ occurs. If there exist this relation ($x \succ_k^F y$) then the degree of fuzzy dominance is represented as

$$\mu_{\text{dom}}^k(g_k(y) - g_k(x)) = \mu_{\text{dom}}^k(x \succ_k^F y). \quad (16)$$

The membership function μ_{dom}^k is used to calculate a fuzzy k -dominance solution and its value is zero for the negative arguments [47]. By using ϵ -Fuzzy k -dominance, non-zero values are allowed. In order to obtain non-zero values, a trapezoidal membership function is used. In other words, $\mu_{\text{dom}}^k(g_k(y) - g_k(x))$ is zero if $g_k(y) \geq g_k(x)$. Assume Δ_k signifies the maximum value of $g_k(y) - g_k(x)$ and membership value $\mu_{\text{dom}}^k(g_k(y) - g_k(x))$ is 1 if $g_k(y) - g_k(x) \geq -\epsilon$ indicates that $\mu_{\text{dom}}^k(g_k(y) - g_k(x))$ reaches to its maximum value. The membership function $\mu_{\text{dom}}^k(g_k(y) - g_k(x))$ is defined as follows:

$$\mu_{\text{dom}}^k(g_k(y) - g_k(x)) = \begin{cases} 0 & g_k(y) - g_k(x) \leq -\epsilon \\ \frac{g_k(y) - g_k(x) + \epsilon}{\Delta_k} & -\epsilon < (g_k(y) - g_k(x)) < \Delta_k - \epsilon \\ 1 & g_k(y) - g_k(x) \geq \Delta_k - \epsilon \end{cases} \quad (17)$$

- **ϵ -Fuzzy dominance to a solution:** The solution $x \in \sigma$ is fuzzy dominance solution iff for every $k \in (1, 2, \dots, M)$, $x \succ_k^F y$ holds. Fuzzy intersection operator and t-norm operator is used to calculate the degree of fuzzy dominance and it is stated as follows:

$$\mu_{\text{dom}}(x \succ^F y) = \bigcap_{k=1}^M \mu_{\text{dom}}^k(x \succ_k^F y). \quad (18)$$

- **ϵ -Fuzzy dominance in the solution space:** There exist a set (solution) S and the solution $y \in S$ is fuzzy dominated if given solution set is dominated by another solution set $x \in S$. Therefore, union operation with t-co norm is used to calculate the value of fuzzy dominance.

$$\mu_{\text{dom}}(S \succ^F y) = \bigcup_{x \in S} \mu_{\text{dom}}(x \succ^F y). \quad (19)$$

After calculating the degree of fuzzy dominance, every solution is having a single value that signifies by how much one solution dominates another solution in the solution set. If the value of fuzzy dominance is less, it indicates that a better solution is achieved. Hence, after each iteration, this procedure is helpful to find out the best solution according to fuzzy dominance value.

Task priority

Before assigning the task to the specified processors, we determined the order of tasks in order to satisfy dependency constraints among tasks. The concept of rank is used to generate the topological ordering of tasks; the upward rank values are recursively computed as follows:

$$\text{rank}(t_j) = \overline{w(t_j)} + \max_{t_s \in \text{succ}(t_j)} (\overline{\text{CT}(t_j, t_s)} + \text{rank}(t_s)), \quad (20)$$

where $\text{succ}(t_j)$ represents the immediate children of task t_j . The rank value for t_{exit} is as:

$$\text{rank}(t_{\text{exit}}) = \overline{w(t_{\text{exit}})} \quad (21)$$

The task that is having a greater value of rank has a higher priority, and then the task scheduling sequence is generated as per the decreasing order of the rank values.

The proposed FDRGS algorithm is depicted in Algorithm 1. Firstly, average execution time $\overline{ET}(t_j)$ is calculated in step 2. Step 5 calculates the average communication time $\overline{CT}(t_j, t_k)$ for all tasks. the upward rank of all tasks is calculated in Step 8, starts from the t_{exit} and sorts the task in decreasing order of rank value. Step 10 calculates the EST and EFT for all the tasks in DAG G. Step 13 calculates the

all the solutions, fuzzy dominance values are calculated (as per Algorithm 2) and if values of fuzzy dominance are same which indicates that both solutions are equally good, then perimeter assignment (Algorithm 3) is used (step 30) to find the uniform spread of solutions. Finally, we select the best K solutions from the SS' to the SS in step 31 representing the best trade-off solution sequence.

Algorithm 2: Calculate fuzzy_dominance_sort (SS)

Input: Scheduling Solution Sequence SS

1. **for** $l=1$ to $len(SS)$
2. $\alpha \leftarrow 0$ //Initializes temporary variable for storing the value of $\mu_{dom}(SS \succ^F SS_L)$
3.   **for** $w=1$ to $len(SS)$
4.   **if** ($l==w$) then
5.   continue;
6.   **End if**
7. $\mu = 1$
8.   **if** $SS_l \succ SS_w$ then
9.   **for** $k=1$ to M // M objectives
10.   Calculate the value of $\mu_{dom}^k(g_k(SS_l) - g_k(SS_w))$ //according to eq (17)
11.   Calculate $\mu \leftarrow \mu \bigcap \mu_{dom}^k(g_k(SS_l) - g_k(SS_w))$ //according to eq (18)
12.   **End for**
13.   **End if**
14. $\alpha \leftarrow \alpha \cup \mu$ // according to eq (19).
15.   **End for**
16. $\mu_{dom}(SS \succ^F SS_l) \leftarrow \alpha$
17. $SS \leftarrow$ Sort the solutions in increasing order of fuzzy dominance in SS.
18. $SS \leftarrow$ Perimeter_Assignment (SS)   // Algorithm3
19. **End for**

energy efficient frequency for each physical machine PM_r . Step 14 calculates the reliability index for every task. In step 16, algorithm initializes a scheduling solution sequence SS to store all possible solutions and initializes it to empty. Step 21 firstly iterates for all the tasks sorted by their order of rank. The main inspiration is to lengthen each solution in SS by assigning the next task which is to execute on m possible resources. New solutions can be generated by iterating it over all resources in step 25 and the new intermediate schedules are added to the new set SS in step 26. After obtaining

Algorithm 2 calculates the fuzzy dominance values for the solutions represented by SS . Step 2, creates a temporary variable α which is used for storing fuzzy dominance values in a solution set (step13). Steps 3–12 calculates the value of fuzzy k -dominance using Eq. (17) which is then used to compute fuzzy dominance by a solution as per Eq. (18) and fuzzy dominance in a solution space is calculated as per Eq. (19).

If multiple solutions have the same fuzzy dominance values, then the perimeter assignment function (Algorithm 3) is used to find the spread of solutions that represents the perimeter value of the largest M dimensional hypercube in the whole solution space. Boundary solutions are assigned with infinity.

The solutions with high perimeter value are preferred as it maintains the diversity and indicate the sparsity region along the scheduling solution space. The value of perimeter $p(SS_l)$ is

$$p(SS_l) = \sum_{k=1}^M \frac{g_k(a) - g_k(b)}{(\max(g_k) - \min(g_k))}, \quad (22)$$

where a and b are the adjacent solution having the same value of fuzzy dominance. $g_k(a)$ and $g_k(b)$ are the k th objective solution. $\max(g_k)$ and $\min(g_k)$ are the largest and smallest value of k th objective function of solution sequence SS .

Algorithm3: Perimeter_Assignment(SS)

Input: Require solution sequence SS

1. Calculate $L = \text{len}(SS)$.
2. **for** $l=1$ to L
3. $p(SS_l) = 0$ //Initialize the perimeter value of all solutions to zero.
4. **End for**
5. **for** $k=1$ to 2 //for 2 objectives(i.e. energy consumption eq.(10), RF eq.(15))
6. $SS \leftarrow \text{Sort}(SS, g_k)$
7. $p(SS_1) = p(SS_L) = \infty$; // Initialize the boundary value of parameter to infinity
8. **for** $l=2 \dots L-1$ do
9. Calculate the perimeter value for rest of the scheduling sequence using eq.(22).
10. **End for**
11. **End for**

Algorithm 3 calculates the perimeter value of all solutions corresponding to M objectives. Step 3 initializes all solution values to zero. Algorithm sorts all the solutions based on objective value g_k in step 6. Set the initial and last perimeter value to infinity in step 7. After that in step 9 calculates the value of all the remaining solutions using Eq. 22.

Illustrative example

In this section, we consider an example to illustrate the proposed FDRGS algorithm and explain the process of generating and selecting scheduling sequences. We consider three tasks (t_1, t_2, t_3) run on two physical machines (PM_1 and PM_2). Firstly, for task t_1 , two solutions are generated and store it in a temporary solution set S . Generated Solutions are: $S_1 = \{(t_1, PM_1)\}$, $S_2 = \{(t_1, PM_2)\}$. These two solutions are sorted using fuzzy dominance sort (using Algorithm 2). When task t_1 is assigned and next task t_2 is ready, four possible solutions are generated and two solutions are selected from S and store in scheduling set SS . Generated solutions

are: $S_1 = \{(t_1, PM_1)(t_2, PM_1)\}$, $S_2 = \{(t_1, PM_1)(t_2, PM_2)\}$, $S_3 = \{(t_1, PM_2)(t_2, PM_1)\}$, $S_4 = \{(t_1, PM_2)(t_2, PM_2)\}$. Fuzzy dominance sort is applied and selected first K solution i.e. S_2 and S_3 . When task t_1 and t_2 are assigned, and t_3 is ready to execute then 8 possible solutions are generated and stored in set SS . Generated Solution are: $\{S_1 = \{(t_1, PM_1)(t_2, PM_1)(t_3, PM_1)\}$, $S_2 = \{(t_1, PM_1)(t_2, PM_1)(t_3, PM_2)\}$, $S_3 = \{(t_1, PM_2)(t_2, PM_1)(t_3, PM_1)\}$, $S_4 = \{(t_1, PM_2)(t_2, PM_1)(t_3, PM_2)\}$, $S_5 = \{(t_1, PM_1)(t_2, PM_2)(t_3, PM_2)\}$, $S_6 = \{(t_1, PM_1)(t_2, PM_2)(t_3, PM_1)\}$, $S_7 = \{(t_1, PM_2)(t_2, PM_1)(t_3, PM_1)\}$, $S_8 = \{(t_1, PM_2)(t_2, PM_2)(t_3, PM_1)\}$. Fuzzy dominance sort is applied, if two or more solutions have same

fuzzy dominance value then a tie is broken with the help of perimeter assignment function (Algorithm 3). Finally, two solutions S_5 and S_7 are selected i.e. $SS = \{S_5, S_7 = \{(t_1, PM_1)(t_2, PM_2)(t_3, PM_2)\}\{(t_1, PM_2)(t_2, PM_1)(t_3, PM_1)\}\}$. For the simplicity and clarity of the example, we omit the detail of fuzzy dominance sort and focused mainly on highlighting the process of generating the scheduling sequence by proposed FDRGS algorithm.

Simulation results and analysis

In order to evaluate the proposed FDRGS algorithm, we have conducted simulation experiments on CloudSim toolkit [48] considering different DAG applications like Gaussian elimination (GE) [49] and fast Fourier transform (FFT) [50] representing the real world workflow applications. We considered different matrix sizes to vary the number of tasks. We compared our proposed FDRGS algorithm with differ-

Table 2 Parameter values

Parameters	Corresponding values
Frequency level of PMs (f_l)	1.60–2.80 GHz
Computing capacity of resources(in MIPS)	1000–3000 MIPS
No of frequency levels supported	4 with $f_{\min} = 40\%$ of $f_{\max}$
Link bandwidth	50–100 (Mbps)
Failure rate	10^{-5} – 10^{-7} failures/s

ent state of art scheduling algorithms such as HEFT [10], Energy consciousness scheduling (ECS) [21] and reliability-aware HEFT (RHEFT) [11]. HEFT algorithm chooses the tasks according to their rank and assigns task to those processors having minimum EFT to minimize the makespan only. To minimize the energy consumption, ECS algorithm uses the concept of relative superiority (RS) to choose a processor for the ready task at suitable voltage/frequency. The processor having the largest RS value is chosen for the ready task. Reliability aware HEFT (RHEFT) aims to increase the system reliability considering the failure rate of the processors for mapping the tasks (Table 2). An extensive simulation was performed on real world task graphs by varying the number of available processors/physical machines, number of tasks and at different computation to communication ratio (CCR) values.

Performance metrics

The proposed FDRGS algorithm's performance is measured by considering following metrics:

- **Reliability:** in workflow application, reliability is the probability of task execution over the assigned processor successfully. System reliability is obtained by multiplying reliabilities of all tasks as in Eq. (15).
- **Energy consumption:** the complete energy consumption (ξ_{tot}) by processor during the task's execution is the summation of energy consumed by all tasks as in Eq. (10).
- **Runtime:** it is the length of time during which all the tasks execute and it is as shown in Table 5.

Workflow structure

Generally, a workflow application is modelled through DAG where the numbers of tasks are connected by precedence relation/dependencies. To perform the simulation of the workflow applications (DAGs), we used two sets of real-life problems i.e. GE and FFT task graphs. To evaluate the scheduling performance in cloud computing systems, Gaussian elimination and fast Fourier transform have been widely used which are briefly introduced as follows.

Fig. 2 GE graph with matrix size 5

Gaussian elimination (GE) task graph

Gaussian Elimination algorithm is used to find a solution of the linear equations by calculating the rank of a matrix and inverse of an invertible square matrix in mathematics. Gaussian elimination is a parallel application, which has precedence constraints. Assume that τ is the matrix size, which illustrates the GE graph. The total number of tasks are $\frac{\tau^2 + \tau - 2}{2}$ where $\tau \geq 2$. Figure 2 depicts a GE graph with matrix size 5. The organization of the Gaussian elimination application graph is not changed, so we do not require mentioning the shape parameter and number of tasks explicitly. However, we varied the size of a matrix and CCR values in GE. In order to consider different sizes of the workflow, different values of matrix size are considered. Table 3 shows the values for the GE task graphs.

Table 3 Values for GE task graphs

Parameter	Feasible values
CCR	0.2, 0.5, 1, 1.2
Processor	4, 8, 16, 32
Size	9, 20, 35, 65

Fig. 3 FFT application with $\tau = 4$ **Table 4** Parameter values for the FFT task graph

Parameter	Feasible values
CCR	0.2, 0.5, 1, 1.2
Processor	4, 8, 16, 32
Size	9, 20, 35, 65

Fast Fourier transform (FFT) task graph

FFT computation is used in different type of applications like mathematics, engineering and science. This algorithm calculates the discrete Fourier Transform and the inverse transform. Figure 3 shows FFT graph with the matrix size $\tau = 4$. Table 4 shows the parameter values considered for the FFT task graphs. The total number of tasks is calculated as:

$$|\text{Task}_{\text{tot}}| = (2 \times \tau - 1) + \tau \times \log_2 \tau. \quad (23)$$

Results and analysis

In this section, we evaluated the proposed FDRGS algorithm with workflows represented by GE task graphs.

Performance at different sizes of task graphs

At first, we considered the varying number of tasks 9–65 (9, 20, 35 and 65) by considering different matrix sizes in GE to estimate the performance of the FDRGS algorithm.

Fig. 4 Energy consumption with different number of tasks for Gaussian elimination graph **Fig. 5** Reliability with varying number of tasks for Gaussian elimination graph

The results obtained corresponding to energy consumption and reliability are represented in Figs. 4 and 5, respectively. In Fig. 4, results show that our proposed FDRGS algorithm consumes less energy as compared to other comparative scheduling algorithms. FDRGS algorithm provides savings in energy with respect to ECS by 15–27%, HEFT by 25–31%, and RHEFT by 39–45%. HEFT algorithm is the popular list based scheduling algorithm and it considers only the makespan factor. In RHEFT algorithm, reliability and makespan are the prime concerns and they schedule tasks on most reliable processors without paying attention to the energy consumption, that's why energy consumption is high in RHEFT algorithm. In ECS, the RS factor is used that contributes towards minimizing the makespan and energy consumption so overall gain in energy savings is less. Further, we observed that as the number of tasks increases, savings in energy is also increased due to the presence of dependency constraints where we can apply DVFS to reduce the energy consumption further.

From Fig. 5, we observed that our proposed algorithm FDRGS performs better than other algorithms in terms of reliability. In case of ECS, they have applied DVFS to minimize energy consumption without considering the fact that with voltage scaling, reliability of the resource

Fig. 6 Energy consumption with different number of processors for Gaussian elimination graph

Fig. 7 System's reliability with different number of processors for Gaussian elimination graph

Fig. 8 Energy consumption with different value of CCR for Gaussian elimination Graph

decreases exponentially. Thus, it provides very low reliability. However, in our proposed FDRGS, we are considering the appropriate frequency above the energy efficient frequency (f^{ee}) for the resources so that reliability is not compromised while preserving energy consumption. Reliability is minimum in ECS because they focused on makespan and energy consumption only.

Performance at different number of processors

Next, we considered different processors (such as 4, 8, 16 and 32) for workflow application with 65 tasks to estimate per-

Fig. 9 System's reliability with different value of CCR for Gaussian elimination Graph

formance of FDRGS algorithm in Figs. 6 and 7. It is depicted in Fig. 6 that the energy saving increases as the number of processor increases. In energy consumption objectives, our FDRGS algorithm reduce energy to ECS by 11–21%, RHEFT by 26–38% and HEFT by 14–25%. This is due to the number of processors increases because when the size of processor is going to increase; less number of tasks is allotted to each processor. Due to this, the chances of scaling the voltage appropriately are increased and thus, in the proposed FDRGS algorithm, energy consumption is minimized.

Next, reliability of the FDRGS algorithm is better as compared to other considered algorithms. Due to availability of large number of processors, probabilities of choosing good resources are increased.

Performance at different values of CCR

Figures 8 and 9 represent the FDRGS performance. In workflow applications, we vary the CCR value considering 65 tasks with 10 processors. In ECS algorithm, as the CCR value is going to increase, the energy savings decreases because reduction of communication energy is not considered. In our proposed algorithm FDRGS, for computation intensive graph ($CCR < 1$), the percentage of energy saving with regard to ECS from 20 to 12%, HEFT by 28–21% and RHEFT by 32–28%. Further, for communication intensive graph, energy saving is there with regard to ECS by 10%, HEFT by 20% and RHEFT by 26%.

Next, we conducted simulation experiments for the FFT application graph. For this, we vary the size of tasks as 9, 14, 27, 65 for the task execution. The simulation results are shown in Figs. 10 and 11 for the energy consumption and reliability by varying number of application tasks, computing processors and their CCR value. Results signifies that proposed FDRGS algorithm gives superior results in terms of energy consumption and reliability for real world problems i.e. FFT task graph.

Fig. 10 (i) Energy consumption for FFT task graphs at varied tasks, (ii) energy consumption for FFT task graphs at varied number of processor, (iii) energy consumption for FFT task graphs by varying CCR

Further, Table 5 shows the runtime of our proposed algorithm FDRGS and comparative algorithms ECS [21], HEFT [10] and reliability HEFT (RHEFT) [11] for executing different tasks on FFT and GE task graphs at number of resources (8) and CCR equals to 1. The results show that runtime of the FDRGS algorithm increases, as the number of tasks is increasing because large number of tasks is required to generate the optimal scheduling solution. Run time complexity of the proposed FDRGS algorithm varies with the size of the application i.e. number of tasks. However, our FDRGS algorithm takes less run time to other comparative algorithms and is very computational efficient. This happens due to the pruning-off the multiple trade-off solutions using fuzzy dominance sorting.

Statistical analysis

Statistical significance of the results corresponding to the proposed FDRGS algorithm is determined by the coefficient of variation (CoV). This is a statistical method which is used to determine the distribution of data about the mean value. It is data deviation as a proportion of its mean value, and is calculated as follows:

$$\text{CoV} = \frac{\text{SD}}{\text{Mean}} \times 100,$$

where SD represents standard deviation.

To guarantee statistical correctness, we run our proposed FDRGS algorithm to an average of 15–20 runs. CoV of average energy consumption and average reliability has been studied with regard to different number of tasks and num-

Fig. 11 (i) Reliability with varying number of tasks for FFT task graph, (ii) reliability for FFT at different number of processor, (iii) reliability at different CCR for FFT task graph

Table 5 Runtime of four algorithms (our proposed algorithm FDRGS and comparative algorithm ECS [21], HEFT [10] and reliability HEFT (RHEFT) [11])

Task graph	Runtime of four algorithms (s)				
	Number of tasks	FDRGS	ECS	RHEFT	HEFT
FFT	9	9.01	19.21	23.56	25.23
	20	22.20	29.89	31.12	34.14
	35	34.06	40.67	45.34	50.41
	65	60.15	67.17	73.23	80.23
GE	9	7.56	14.42	19.65	21.13
	20	16.76	22.12	27.24	31.10
	35	26.34	34.53	41.56	45.85
	65	52.15	58.77	69.10	76.21

bers of processors corresponding to Gaussian Elimination are shown in Tables 6, 7, 8 and 9. Smaller value of CoV for

proposed FDRGS algorithm signifies that it is more efficient and stable with respect to comparative algorithms.

A similar trend has been followed by fast Fourier transform workflow task graphs. CoV of average energy consumption and average reliability has been studied with respect to different number of tasks and number of processors is shown in Tables 10, 11, 12 and 13.

Conclusion

Due to increasing demand for computation, energy consumption in data centers is increasing. Thus, reduction in energy consumption is the primary concern for the data centers. In recent, the need for minimizing energy consumption and maximizing system reliability has become important research. In this paper, we are reducing the system's energy consumption by using DVFS technique along with maintaining the reliability of task execution. We proposed a ϵ -fuzzy dominance based reliable green workflow

Table 6 Statistical analysis for energy consumption for varied tasks for GE task graph

Tasks	FDRGS			ECS			RHEFT			HEFT		
	Mean	SD	CoV (%)	Mean	SD	CoV (%)	Mean	SD	CoV (%)	Mean	SD	CoV (%)
9	320	0.016	0.5	924	0.064	0.7	2766	0.248	0.9	4267	0.426	1
20	376	0.033	0.9	1153	0.126	1.1	3656	0.475	1.3	5867	0.792	1.35
35	400	0.052	1.3	1192	0.163	1.37	3969	0.595	1.5	6235	0.972	1.56
65	529	0.079	1.5	1598	0.252	1.58	4759	0.771	1.62	7789	1.300	1.67

Table 7 Statistical analysis for reliability for varying number of tasks for GE Task graph

Tasks	FDRGS			ECS			RHEFT			HEFT		
	Mean	SD	CoV (%)	Mean	SD	CoV (%)	Mean	SD	CoV (%)	Mean	SD	CoV (%)
9	0.828	0.00005	0.70	0.899	0.00007	0.85	0.899	0.00007	0.85	0.888	0.00008	1
20	0.662	0.00007	1.10	0.7	0.00008	1.2	0.675	0.00007	1.10	0.676	0.00008	1.20
35	0.688	0.00008	1.30	0.71	0.00009	1.34	0.686	0.00008	1.25	0.694	0.00009	1.31
65	0.791	0.00011	1.45	0.877	0.00012	1.40	0.872	0.00012	1.40	0.855	0.00012	1.45

Table 8 Statistical analysis for energy consumption for varying number of processor for GE task graph

Processor	FDRGS			ECS			RHEFT			HEFT		
	Mean	SD	CoV (%)	Mean	SD	CoV (%)	Mean	SD	CoV (%)	Mean	SD	CoV (%)
4	2565	0.3334	1.3	3168	0.4498	1.42	4054	0.608	1.5	4958	0.758	1.53
8	2874	0.4311	1.5	3768	0.5915	1.57	5064	0.815	1.61	6296	1.057	1.68
16	2989	0.4752	1.59	3856	0.6440	1.67	4820	0.814	1.69	6678	1.182	1.77
32	3476	0.5909	1.7	4439	0.7723	1.74	6245	1.099	1.76	7993	1.47	1.84

Table 9 Statistical analysis for reliability for varying number of processor for GE task graph

Processor	FDRGS			ECS			RHEFT			HEFT		
	Mean	SD	CoV (%)	Mean	SD	CoV (%)	Mean	SD	CoV (%)	Mean	SD	CoV (%)
4	0.867	0.00009	1.1	0.88	0.0078	1.19	0.89	0.0001	1.2	0.9	0.0001	1.25
8	0.6	0.00008	1.4	0.64	0.0042	1.43	0.66	0.00008	1.34	0.67	0.00009	1.43
16	0.68	0.00009	1.45	0.69	0.0048	1.5	0.70	0.0001	1.48	0.72	0.0001	1.51
32	0.827	0.00012	1.5	0.87	0.0074	1.55	0.86	0.0001	1.57	0.86	0.0001	1.6

Table 10 Statistical analysis for energy consumption for varying number of tasks for GE task graph

Tasks	FDRGS			ECS			RHEFT			HEFT		
	Mean	SD	CoV (%)	Mean	SD	CoV (%)	Mean	SD	CoV (%)	Mean	SD	CoV (%)
9	350	0.0189	0.54	954	0.0706	0.74	2986	0.283	0.95	4667	0.5601	1.20
20	425.6	0.0387	0.91	1253	0.1453	1.16%	3986	0.5262	1.32	6387	0.8879	1.39
35	467	0.0654	1.3	1343	0.1881	1.4	4197	0.6505	1.55	6782	1.0715	1.58
65	579	0.0897	1.55	1608	0.2557	1.59	4989	0.8183	1.64	8495	1.4357	1.69

Table 11 Statistical analysis for energy consumption for varying number of processor for FFT task graph

Processor	FDRGS			ECS			RHEFT			HEFT		
	Mean	SD	CoV (%)	Mean	SD	CoV (%)	Mean	SD	CoV (%)	Mean	SD	CoV (%)
4	3565	0.484	1.36	4168	0.5960	1.43	4567	0.7078	1.55	5124	0.8044	1.57
8	4024	0.5030	1.25	4989	0.7883	1.58	5564	0.9236	1.66	6456	1.0910	1.69
16	4134	0.6615	1.6	5234	0.8846	1.69	5786	0.9894	1.71	6790	1.2087	1.78
32	4836	0.8319	1.72	5739	1.01	1.76	6675	1.188	1.78	8393	1.5527	1.85

Table 12 Statistical analysis for reliability for varying number of tasks for FFT task graph

Tasks	FDRGS			ECS			RHEFT			HEFT		
	Mean	SD	CoV (%)	Mean	SD	CoV (%)	Mean	SD	CoV (%)	Mean	SD	CoV (%)
9	0.828	0.00006	0.75	0.899	0.00007	0.86	0.89	0.00007	0.8	0.88	0.00009	1.1
20	0.662	0.00007	1.2	0.65	0.00007	1.23	0.62	0.00006	1.12	0.67	0.00008	1.21
35	0.688	0.00009	1.32	0.68	0.00009	1.36	0.68	0.00008	1.28	0.69	0.00009	1.35
65	0.791	0.00011	1.48	0.877	0.00012	1.42	0.87	0.00012	1.42	0.85	0.00012	1.47

Table 13 Statistical analysis for reliability for varying number of processor for FFT task graph

Processor	FDRGS			ECS			RHEFT			HEFT		
	Mean	SD	CoV (%)	Mean	SD	CoV (%)	Mean	SD	CoV (%)	Mean	SD	CoV (%)
4	0.85	0.00009	1.1	0.89	0.0001	1.2	0.87	0.0001	1.28	0.91	0.00011	1.29%
8	0.61	0.00008	1.42	0.656	0.00009	1.48	0.67	0.00009	1.39	0.66	0.00009	1.47%
16	0.6	0.00008	1.48	0.69	0.0001	1.54	0.71	0.0001	1.57	0.697	0.0001	1.55%
32	0.84	0.00013	1.59	0.88	0.0001	1.59	0.86	0.00013	1.6	0.88	0.0001	1.64%

scheduling (FDRGS) algorithm, which optimizes the application's reliability and energy consumption jointly using the ϵ -fuzzy dominance mechanism. The proposed FDRGS algorithm address the energy consumption and reliability very well in case of occurrence of transient faults and is helpful to find single best-compromised solution by taking the relative fitness of the trade-off solutions with less computation time. The performance of the proposed FDRGS algorithm is evaluated against other algorithms like ECS, HEFT and RHEFT by considering the real-world application workflows such as GE and FFT graphs. We validated our results by a well known statistical test namely coefficient of variation. The simulation results clearly manifest that the proposed FDRGS algorithm outperforms other algorithms considered for comparison in terms of energy consumption and reliability. FDRGS algorithm provides saving in energy by 15–45%. The proposed algorithm will be helpful for researchers, engineers in establishing reliable and energy efficient cloud applications. In future, we will extend our algorithm by considering other important objectives like cost and makespan for workflow applications. The deployment of our work on a multi-cloud or real cloud environment will also be a future step.

Acknowledgements The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Declarations

Conflict of interest On behalf of all authors, the corresponding author states that there is no conflict of interest.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

References

- Cisco (2020) Cisco Annual Internet Report (2018–2023). Cisco, pp 1–41
- Etinski M, Corbalan J, Labarta J, Valero M (2012) Understanding the future of energy-performance trade-off via DVFS in HPC environments. *J Parallel Distrib Comput* 72:579–590. <https://doi.org/10.1016/j.jpdc.2012.01.006>
- n.d. (2015) <http://www.Top500.org/>
- Theis T, Wong P (2017) The end of Moore's law: a new beginning for information technology. *Comput Sci Eng* 19:41–50
- Wajid U, Cappiello C, Plebani P, Pernici B, Mehandjiev N, Vitali M et al (2016) On achieving energy efficiency and reducing CO₂ footprint in cloud computing. *IEEE Trans Cloud Comput* 4:138–151. <https://doi.org/10.1109/TCC.2015.2453988>
- Venkatachalam V, Franz M (2005) Power reduction techniques for microprocessor systems. *ACM Comput Surv* 37:195–237. <https://doi.org/10.1145/1108956.1108957>
- Beloglazov A, Abawajy J, Buyya R (2012) Energy-aware resource allocation heuristics for efficient management of data centers for Cloud computing. *Futur Gener Comput Syst* 28:755–768. <https://doi.org/10.1016/j.future.2011.04.017>
- Kartik S, Murthy CSR (1997) Task allocation algorithms for maximizing reliability of distributed. *Comput Syst* 46:719–724. <https://doi.org/10.1109/12.600888>
- Khan MA (2012) Scheduling for heterogeneous Systems using constrained critical paths. *Parallel Comput* 38:175–193. <https://doi.org/10.1016/j.parco.2012.01.001>
- Topcuoglu H, Hariri S, Society IC (2002) Performance-effective and low-complexity. *IEEE Trans Parallel Distrib Syst* 13:260–274
- Dongarra JJ, Jeannot E, Saule E, Shi Z (2007) Bi-objective scheduling algorithms for optimizing makespan and reliability on heterogeneous systems. In: *Annu ACM symp parallelism algorithms archit*, pp 280–288. <https://doi.org/10.1145/1248377.1248423>
- Atakan D, Özgüner F (2005) Biobjective scheduling algorithms for execution time-reliability trade-off in heterogeneous computing systems. *Comput J* 48:300–314
- Tang X, Li K, Qiu M, Sha EHM (2012) A hierarchical reliability-driven scheduling algorithm in grid systems. *J Parallel Distrib Comput* 72:525–535. <https://doi.org/10.1016/j.jpdc.2011.12.004>
- Zhang Y, Chakrabarty K (2003) Energy-aware adaptive checkpointing in embedded real-time systems. In: *Proc-Design, Autom Test Eur*, pp 918–23. <https://doi.org/10.1109/DATE.2003.1253723>
- Dakai Zhu, Rami Melhem DM. The effects of energy management on reliability in real-time embedded systems
- Wang L, Lu Y (2008) Efficient power management of heterogeneous soft real-time clusters. In: *Proc-Real-Time Syst Symp*, pp 323–32. <https://doi.org/10.1109/RTSS.2008.31>
- Verma A, Kaushal S (2017) A hybrid multi-objective particle swarm optimization for scientific workflow scheduling. *Parallel Comput* 62:1–19. <https://doi.org/10.1016/j.parco.2017.01.002>
- Garg H, Sharma SP (2013) Multi-objective reliability-redundancy allocation problem using particle swarm optimization. *Comput Ind Eng* 64:247–255. <https://doi.org/10.1016/j.cie.2012.09.015>
- Ritu Garg AS (2011) Multi-objective workflow grid scheduling based on discrete particle swarm optimization. In: *Int. Conf. Swarm, Evol. Memetic Algorithms*, pp 183–190
- Garg R, Singh AK (2014) Multi-objective workflow grid scheduling using ϵ -fuzzy dominance sort based discrete particle swarm optimization. *J Supercomput* 68:709–732. <https://doi.org/10.1007/s11227-013-1059-8>
- Lee YC, Zomaya AY (2009) Minimizing energy consumption for precedence-constrained applications using dynamic voltage scaling. In: *2009 9th IEEE/ACM Int Symp Clust Comput Grid, CCGRID*, pp 92–99. <https://doi.org/10.1109/CCGRID.2009.16>
- Lin W, Wang W, Wu W, Pang X, Liu B, Zhang Y (2018) A heuristic task scheduling algorithm based on server power efficiency model in cloud environments. *Sustain Comput Inform Syst* 20:56–65. <https://doi.org/10.1016/j.suscom.2017.10.007>
- Li Z, Ge J, Hu H, Song W, Hu H, Luo B (2015) Cost and energy aware scheduling algorithm for scientific workflows with deadline constraint in clouds. *IEEE Trans Serv Comput* 11:713–726. <https://doi.org/10.1109/tsc.2015.2466545>
- Garg R, Singh AK (2016) Energy-aware workflow scheduling in grid under QoS constraints. *Arab J Sci Eng* 41:495–511. <https://doi.org/10.1007/s13369-015-1705-y>
- Mao L, Li Y, Peng G, Xu X, Lin W (2018) A multi-resource task scheduling algorithm for energy-performance trade-offs in green clouds. *Sustain Comput Inform Syst* 19:233–241. <https://doi.org/10.1016/j.suscom.2018.05.003>
- Stavrinides GL, Karatza HD (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. *Futur Gener Comput Syst* 96:216–226. <https://doi.org/10.1016/j.future.2019.02.019>
- Maurya AK, Modi K, Kumar V, Naik NS, Tripathi AK (2019) Energy-aware scheduling using slack reclamation for cluster systems. *Cluster Comput*. <https://doi.org/10.1007/s10586-019-02965-7>
- Nzanywayingoma F, Yang Y (2019) Efficient resource management techniques in cloud computing environment: a review and discussion. *Int J Comput Appl* 41:165–182. <https://doi.org/10.1080/1206212X.2017.1416558>
- Sharma M, Garg R (2019) HIGA: harmony-inspired genetic algorithm for rack-aware energy-efficient task scheduling in cloud data centers. *Eng Sci Technol*. <https://doi.org/10.1016/j.jestech.2019.03.009>
- Boeres C, Sardiña IM, Drummond LMA (2011) An efficient weighted bi-objective scheduling algorithm for heterogeneous systems. *Parallel Comput* 37:349–364. <https://doi.org/10.1016/j.parco.2010.10.003>
- Faragardi HR, Shojaei R, Tabani H, Rajabi A (2013) An analytical model to evaluate reliability of cloud computing systems in the presence of QoS requirements. In: *2013 IEEE/ACIS 12th Int Conf Comput Inf Sci ICIS 2013—Proc 2013*, pp 315–21. <https://doi.org/10.1109/ICIS.2013.6607860>
- He Y, Shao Z, Xiao B, Zhuge Q, Sha E (2003) Reliability driven task scheduling for heterogeneous systems. *Proc IASTED Int Conf Parallel Distrib Comput Syst* 15:465–470
- Tang X, Li K, Li R, Veeravalli B (2010) Reliability-aware scheduling strategy for heterogeneous distributed computing systems. *J Parallel Distrib Comput* 70:941–952. <https://doi.org/10.1016/j.jpdc.2010.05.002>
- Wang X, Buyya R, Su J (2009) Reliability-oriented genetic algorithm for workflow applications using max-min strategy. In: *2009 9th IEEE/ACM Int Symp Clust Comput Grid, CCGRID 2009*, pp 108–15. <https://doi.org/10.1109/CCGRID.2009.14>
- Mahato DP, Singh RS (2019) Load balanced scheduling and reliability modeling of grid transaction processing system using colored Petrinets, p 84
- Zhou A, Wang S, Zheng Z, Hsu CH, Lyu MR, Yang F (2016) On cloud service reliability enhancement with optimal resource usage. *IEEE Trans Cloud Comput* 4:452–466. <https://doi.org/10.1109/TCC.2014.2369421>
- Kim KH, Buyya R, Kim J (2007) Power aware scheduling of bag-of-tasks applications with deadline constraints on DVS-enabled clusters. In: *Proc—Seventh IEEE Int Symp Clust Comput Grid, CCGRID 2007*, pp 541–548. <https://doi.org/10.1109/CCGRID.2007.85>
- Wang L, Khan SU, Chen D, Kołodziej J, Ranjan R, Xu CZ et al (2013) Energy-aware parallel task scheduling in a cluster.

- Futur Gener Comput Syst 29:1661–1670. <https://doi.org/10.1016/j.future.2013.02.010>
39. Wang L, Von Laszewski G, Dayal J, Wang F (2010) Towards energy aware scheduling for precedence constrained parallel tasks in a cluster with DVFS. In: CCGrid 2010—10th IEEE/ACM Int Conf Clust Cloud, Grid Comput, pp 368–77. <https://doi.org/10.1109/CCGRID.2010.19>
  40. Xiao X, Xie G, Xu C, Fan C, Li R, Li K (2018) Minimizing schedule length of energy consumption constrained parallel applications on heterogeneous distributed systems. J Comput Sci 26:344–353. <https://doi.org/10.1016/j.jocs.2017.05.002>
  41. Huang Q, Su S, Li J, Xu P, Shuang K, Huang X (2012) Enhanced energy-efficient scheduling for parallel applications in cloud. In: Proc—12th IEEE/ACM Int Symp Clust Cloud Grid Comput CCGrid 2012, pp 781–786. <https://doi.org/10.1109/CCGrid.2012.49>
  42. Fard HM, Prodan R, Barrionuevo JJD, Fahringer T (2012) A multi-objective approach for workflow scheduling in heterogeneous environments. In: Proc—12th IEEE/ACM Int Symp Clust Cloud Grid Comput CCGrid 2012, pp 300–309. <https://doi.org/10.1109/CCGrid.2012.114>
  43. Durillo JJ, Nae V, Prodan R (2014) Multi-objective energy-efficient workflow scheduling using list-based heuristics. Futur Gener Comput Syst 36:221–236. <https://doi.org/10.1016/j.future.2013.07.005>
  44. Zhang L, Li K, Xu Y, Mei J, Zhang F, Li K (2015) Maximizing reliability with energy conservation for parallel task scheduling in a heterogeneous cluster. Inf Sci (Ny) 319:113–131. <https://doi.org/10.1016/j.ins.2015.02.023>
  45. Sridharan R, Gupta N, Mahapatra R (2008) Feedback-controlled reliability-aware power management for real-time embedded systems. In: Proc—Des Autom Conf, pp 185–90. <https://doi.org/10.1109/DAC.2008.4555805>
  46. Garg R, Mittal M, Son LH (2019) Reliability and energy efficient workflow scheduling in cloud environment. Cluster Comput 22:1283–1297
  47. Koduru P, Dong Z, Das S, Welch SM, Roe JL, Charbit E (2008) A multiobjective evolutionary-simplex hybrid approach for the optimization of differential equation models of gene networks. IEEE Trans Evol Comput 12:572–590. <https://doi.org/10.1109/TEVC.2008.917202>
  48. N., Calheiros R, Ranjan R, Beloglazov A, De Rose CAF, Buyya R (2011) CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. Softw Pract Exp 39:23–50. <https://doi.org/10.1002/spe>
  49. Cosnard M, Marrakchi M, Robert Y (1988) Gaussian elimination task graph
  50. Duhamel P, Vetterli M (1990) Fast Fourier transforms: a tutorial review and a state of the art. Signal Process 19:259–299. [https://doi.org/10.1016/0165-1684\(90\)90158-U](https://doi.org/10.1016/0165-1684(90)90158-U)

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.