Journal of Network and Computer Applications 203 (2022) 103400

Contents lists available at ScienceDirect

Journal of Network and Computer Applications

journal homepage: www.elsevier.com/locate/jnca

Energy-efficient virtual-machine mapping algorithm (EViMA) for workflow tasks with deadlines in a cloud environment

J. Kok Konjaang *, John Murphy, Liam Murphy

School of Computer Science, University College Dublin, Dublin, Ireland

ARTICLE INFO

Keywords:
Energy consumption
Execution Cost
Execution makespan
Workflow scheduling

ABSTRACT

Processing large scientific applications generates a huge amount of data, which makes running experiments in the cloud computing environment very expensive and energy-consuming. To find an optimal solution to the workflow scheduling problem, several approaches have been presented for scheduling workflow on cloud resources. However, more efficient approaches are needed to improve cloud service delivery. In this paper, an energy-efficient virtual machine mapping algorithm (EViMA) is proposed to improve resource management in the cloud computing environment to achieve effective scheduling that reduces cloud data center energy consumption, execution makespan, and execution cost. This ensures that the requirements of cloud users are met, and improves the quality of services offered by cloud providers. Our proposed mechanism considers the heterogeneity of scheduling from both cloud users' and workflow applications' perspectives. Through simulation experiments on real workflow datasets, the proposed EViMA can provide better solutions for both cloud users and cloud providers by reducing energy consumption, execution makespan, and execution cost better than the state-of-the-art.

1. Introduction

Cloud computing has recently emerged as a unique paradigm for scientific discovery. The cloud is a multipurpose and high-performance internet-based computing that facilitates the interconnection of geographically dispersed data centers (Zeng et al., 2015) to enable scientists to model large-scale scientific applications in areas such as bioinformatics, astrophysics, earthquake science, health sciences, disaster modeling, and prediction (Hu et al., 2018). However, processing large scientific applications in the cloud computing environment generates a huge amount of data and hence is very expensive and energy-consuming. Since such complexity in processing these applications can affect the quality of service delivery, scientists have designed a step-by-step sequence (workflow) through which large-scale applications can be executed. A workflow is a sequence of scientific activities that are performed or should be performed to achieve a scientific goal (Konjaang and Xu, 2021). In reality, the components of these activities can be in any executable form (e.g., various datasets, load sets, programs, and report sets) (Alkhanak et al., 2016).

Reducing the energy consumption and execution makespan of scientific workflows while keeping the execution cost of workflow tasks within budget is an effective way to address the workflow scheduling problem. However, the heterogeneity of workflow scheduling makes it difficult for current heuristics to schedule a query in a way that reduces energy consumption, execution makespan, and execution cost. Heterogeneity in scheduling can be viewed from two perspectives: (i) from the workflow application perspective and (ii) from the cloud user perspective. Workflow application heterogeneity includes differences in workflow applications (Montage, CyberShake, SIPHT, Ligo Inspiral, and Epigenomics), differences in the application domain (bioinformatics, astronomy, astrophysics, etc.), and differences in task dependencies between these applications. On the other hand, user heterogeneity includes individual user-specific deadlines and Quality of Services (QoS). The diversity of QoS is due to the different needs of cloud users during workflow execution. Moreover, most of these QoS conflict with each other, and improving on one QoS may affect the others. For example, reducing energy consumption increases the execution cost of scientific workflow applications. This is because energy consumption and execution cost are conflicting goals that, if not managed well, can lead to client dissatisfaction. The problem is how algorithms can be developed to deal with this heterogeneity in such a way that the three most important, yet, conflicting workflow scheduling objectives, i.e., energy consumption, makespan, and execution cost, can be minimized simultaneously.

Many researchers have studied energy consumption in cloud data centers and presented various methods to effectively manage cloud resources to reduce energy consumption. For example, Kansal and Chana

* Corresponding author.
E-mail address: james.konjaang@ucdconnect.ie (J. Kok Konjaang).

https://doi.org/10.1016/j.jnca.2022.103400
Received 29 November 2021; Received in revised form 7 March 2022; Accepted 12 April 2022
Available online 29 April 2022

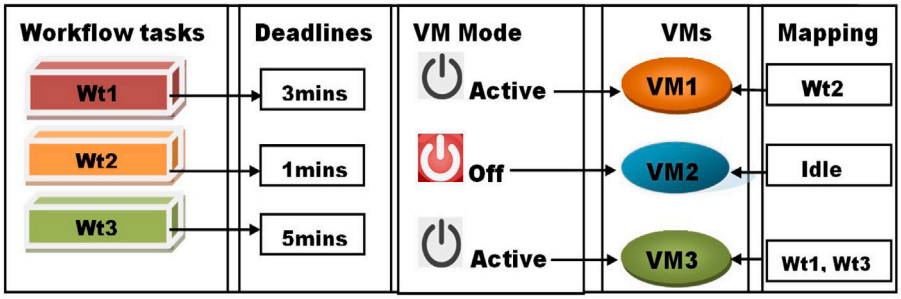

Fig. 1. Workflow Scheduling Example.

(2016) has presented an energy-aware VM migration algorithm that searches for the best unused VMs so that the workload of an overloaded VM(s) can be offloaded to the unused VM(s). Others have attempted to consolidate VMs by reducing the number of PMs to reduce energy consumption (Zhang et al., 2018). However, reducing the number of PMs results in a reduction in the provisioned sum of CPU procession power as the reduced number of PMs is distributed across a given number of VMs. This may affect workflow tasks with tighter deadlines. Unlike previous studies, this study in Fig. 1 presents a different method of managing cloud resources to reduce energy consumption. Suppose we have three workflow tasks (wt_1, wt_2 and wt_3) to be scheduled on three different VMs (VM_1, VM_2 and VM_3). These tasks have different deadlines and need to be distributed among the VMs based on their deadlines. The task wt_2 can be said to be a critical task because its deadline is too short. To prevent wt_2 from missing its deadline, we quickly assigned it to VM_1, as shown in Fig. 1. However, since the deadlines of wt_1 and wt_3 are far apart, they can be assigned to one VM (VM_3). In this scenario, VM_2 is idle and needs to be put on safe mode or turned off. Since VM_2 is off, its PM no longer consumes energy, resulting in a reduction in energy consumption.

This work has focused on developing a heuristic scheduling technique to solve the workflow task scheduling problem in a cloud computing environment. We investigated energy optimization methods in the cloud computing domain. We analyzed their impact and identified some research gaps in these methods and introduced a resource management and allocation model which coordinates with both the resource layer and workflow tasks layer to get the most appropriate tasks mapped onto the available VMs to improve the scheduling objectives. Furthermore, the model provides a mechanism that checks the status of all VMs (idle or use). This is because, in most of the existing algorithms, underutilized VMs are not shut down, which can increase the power consumption of a data center. Our model overcomes this drawback through the use of Idle "off" Busy "on" technique that identifies the idle VMs and turns them off automatically. Moreover, workflows execution normally contains some time gaps (idle time), which may increase the execution cost of workflow tasks. This paper has proposed a slack time harvesting algorithm that is effective in making use of the idle time. Finally, an energy-efficient VM Mapping Algorithm (EViMA) is proposed to support the scheduling model to achieve its aim of managing cloud resources to generate energy optimal schedules. The method is not only efficient in reducing energy consumption, but also can reduce execution cost and execution makespan while improving cloud resource management without affecting tasks deadlines. EViMA is supported by four sub-algorithms including Highly Critical Workflow Task Selection Algorithm (HiCTSA), Low Critical Workflow Task Selection Algorithm (LoCTSA), VM Power Regulating Algorithm (VM-PRA), and Slack Time Harvesting Method (STiHaM). The main contributions of the paper are summarized below:

1. A solution to the workflow scheduling heterogeneity problem that reduces the three most important, yet conflicting goals of workflow scheduling: Energy Consumption, Makespan, and Execution Cost.
2. Introduced VM allocation model in the proposed EViMA to regulate and assign the legitimate workflow tasks to the legitimate VMs to improve workflow scheduling.
3. A slack time harvesting method for data-intensive workflow optimization.
4. Highly critical workflow tasks selection algorithm (HiCTSA) and a low critical workflow task selection algorithm (LoCTSA) for task mapping.
5. A novel measure that identifies the status of VMs (idle or busy). The identified idle VMs are put on power-saving mode to save some energy.
6. Evaluate the variants of the proposed EViMA through empirical models for energy consumption, makespan and execution cost based on real scientific workflows.

The remainder of the paper is organized as follows. An overview of related work is given in Section 2. Section 3 presents the system model including the data center model and the formulation of the multi-objective optimization problem used for the workflow tasks scheduling problem. Section 4 describes the proposed energy-efficient VM mapping algorithm in detail. In Section 5, experimental evaluations are performed to demonstrate the validity of EViMA. In Section 6, we conclude the paper and provided an overview of a possible future work direction.

2. Related work

Improving energy efficiency is very essential for economic gains and environmental sustainability (Zhou et al., 2017). In recent past, research scholars have devoted significant time working on finding Pareto front for energy optimization problems, from heterogeneous multi-core processor scheduling (Salami et al., 2020; Chen and John, 2009; Wang et al., 2015) to the most recent workflow scheduling. These researchers have presented various methods, including meta-heuristics such as the Firefly algorithm (Barlaskar et al., 2016; Kansal and Chana, 2016), ACO (Malekloo and Kara, 2014), PSO-COGENT (Kumar and Sharma, 2018), GA (Ibrahim et al., 2018; Quang-Hung et al., 2013); Hybrid algorithms such as MPSO-FGA (Gabaldon et al., 2017), Hybrid (Babukarthik et al., 2012; Tao et al., 2014), ACO-PSO (Suseela and Jeyakrishnan, 2014); and heuristic methods such as EONS (Chen et al., 2016), MHRA (Juarez et al., 2018), EnReal (Xu et al., 2015), CEAS (Li et al., 2015), ETSA (Panda and Jana, 2019). The work in Ferdaus et al. (2014) presented meta-heuristic method to optimize energy consumption in cloud data center. The authors argued that VM instances have different capacities, some are large while others have relatively high capacity, which in their view, researchers should vary VM utilization and consolidation modeling. They presented different techniques for VM utilization and consolidation modeling, such as multidimensional vector packing, multidimensional resource utilization based on vector algebra, resource utilization and wastage, power consumption, and finally presented the AVVMC algorithm, a modified version of ACO to reduce power consumption. Also, an approach for energy-aware virtual machine migration technique based on Firefly algorithm was

considered by Kansal and Chana (2016). The method consists of four main parts: Source node selection, VMs selection, destination node selection and updated distance values. The proposed method reduces the energy consumption by finding the best idle VMs to share the workload of the overloaded VMs. Another interesting study using meta-heuristic techniques to solve the energy consumption problem was presented in Ibrahim et al. (2018). The researchers first formulated a task scheduling model as an integer linear programming problem. The model considered computational resources, storage resources, network resources and cooling system of a data center and finally presents a GA based method to minimize the energy consumption in the data center. The method is better in generating fair schedules with very low time complexity. Although meta-heuristic approaches are efficient for both exploration and exploitation and can give better results, but they are time consuming due to their many iteration steps (Arabnejad et al., 2018; Mboula et al., 2020). Moreover, none of these approaches consider the impact of workflow scheduling heterogeneity on cloud resources and how algorithms can be designed to manage this heterogeneity.

Various research works based on hybrid workflow scheduling algorithms have also been considered. The study in Azad and Navimipour (2017) combines cultural and Ant Colony Optimization algorithms to reduce both makespan and energy consumption. The proposed method uses ACO to select the best processor for the job and Cultural algorithm to find the best solution for the scheduling problem. The research in Gabaldon et al. (2017) recommended the use of evolution based techniques to address resource matching and scheduling problems for parallel applications. According to the authors, evolutionary based algorithms are efficient in finding optimal solutions for resource matching and scheduling of parallel applications. They presented MPSO-FGA, a combination of PSO and GA to improve both energy consumption and execution makespan. Lawanyashri et al. (2017) presented a hybrid multi-objective method based on Simulated Annealing, known as FOA-SA- LB to reduce execution makespan and energy consumption. In this approach, the first phase deployed FOA technique to allow free movement of the fly swarm in different directions for better exploration and exploitation. In the second phase, simulated annealing is integrated to update the current search positions so that FOA does not converge prematurely. FOA-SA- LB is efficient in balancing the workload, reducing makespan and reducing the energy consumption of data centers. However, the objectives of the aforementioned studies are different from our solution. The previous solutions aimed at optimizing makespan and energy consumption, while our solution aims at managing cloud resources to optimize three most important, yet conflicting workflow scheduling objectives, i.e., energy consumption, execution makespan and execution cost.

Other studies have also focused on using heuristic algorithms to address the workflow scheduling problem. In Garg et al. (2019), the authors used a state-of-the-art heuristic called Reliable and Energy Efficient Workflow Scheduling (REEWS) to reduce application lifetime reliability and energy consumption while ensuring that user-specified QoS constraints are not compromised. In Xu et al. (2015), energy-aware resource allocation (EnReal) was presented. The method uses a dynamic VM provisioning technique to deploy VMs to execute scientific workflow tasks in order to reduce energy consumption and effectively utilize all leased VMs. With the use of resource allocation table, EnReal is able to keep track of all resources, thereby distributing cloud resources evenly to optimize energy consumption.

In Fredy et al. Juarez et al. (2018), a multiheuristic resource allocation (MHRA) algorithm was proposed to provide both cloud service providers and end users with different options for workflow tasks execution. The study revealed several options that can be used to reduce energy consumption in a distributed platform such as cloud computing. These include: (i) applying energy efficient architecture or using Dynamic Voltage Frequency Scaling (DVFS), (ii) using energy efficient patterns in compilers to redevelop algorithms, and (iii) performing task scheduling with different scheduling strategies on the leased resources. Also, the researchers in Cao et al. (2014) used a multi-step heuristic workflow scheduling algorithm known as EARES-D to schedule a workflow DAG on the underlying time-dependent cloud infrastructure. Various objectives and constraints such as client QoS requirements, energy consumption, cost, CO_2 emission, cloud provider profit and resource utilization rate were considered. EARES-D uses DVFS technique to reduce CPU frequency and DNS approach to hibernate all unused servers to save energy. However, using DVFS and DNS techniques together means the algorithm will require more time to compute. This may lead to a delay in executing workflow tasks and hence a discrepancy in the results of the solutions.

Apart from this, other researchers have addressed the energy consumption issue in cloud data center through VM consolidation and migration policies Al-Dulaimy et al. (2018), Patel and Patel (2017), Hieu et al. (2015). In such cases, VM migration policies are deployed to determine whether a host is overloaded or underloaded. These techniques focused on ensuring that workload on cloud resources are shared equally to avoid overloading and underloading. This is done to provide efficient use of deployed cloud resources. The work in Al-Dulaimy et al. (2018) identified inefficient use of physical servers as the source of energy overconsumption in a cloud data center. They presented a number of ideas by which cloud datacenter could deploy to minimize energy consumption including, virtual machine management policy, virtual machine migration policy and virtual machine live migration policy with a focus on VM placement and consolidation strategy. The proposed algorithm is able to generate an optimal solution through the efficient use of cloud resources. The idea of workload consolidation was conceptualized and then a Host Utilization Aware (HUA) algorithm was presented to identify unused or underloaded hosts for workload redistribution in Patel and Patel (2017). HUA is able to evict more unused or underloaded hosts, resulting in energy savings. However, solving large scale bigdata problem like scientific workflow scheduling with VM consolidation and migration policy may result in unnecessary overhead and energy wastage.

The unique aspect of our work is that we focused on cloud resource management to improve the three most important, yet conflicting scheduling objectives – energy consumption, execution makespan and execution cost – while ensuring that all tasks meet their deadlines. We modeled a multi-objective optimization problem and incorporated a VM power regulation algorithm to identify inactive VMs. The identified inactive VMs are put on save mode. Also, a slack time harvesting method is used to obtain the slack times for each VM, which is then used to schedule other tasks. Table 1 gives a summary of the reviewed literature along with their advantages and limitations.

3. System models and multi-objective optimization problem formulation

The proposed solution aims to manage cloud resources to generate an effective schedule that reduces energy consumption, execution makespan and cost of workflow tasks. This section consists of the cloud data center model, the VM allocation model, and the formulation of the multi-objective optimization problem. Table 1 provides a list of the terms and definitions used in this paper.

3.1. Cloud datacenter model

Cloud data center is a network of computing resources that enables application and data sharing. In our proposed data center model, we assume that a data center has a set of M heterogeneous physical machines (PM) that are used to configure a set of k VMs for workflow tasks scheduling. Let $p = (p_1, p_2, p_3 \ldots\ldots p_n)$, denotes the different types of PMs available in a cloud data center. These PMs have different Processing capacity such as CPUs, memory, bandwidth, network I/O, and the size of the storage which are measured in million instructions per second (MIPS) (Medara and Singh, 2021). We further assume that

Table 1
Summary of literature review.

Author(s)	Name of algorithm	Highlights of the approach	QoS parameters	Advantages	Limitations
Ferdaus et al. (2014)	AVVMC	The research proposed adaptation and integration of the Ant Colony Optimization (ACO) to reduce energy consumption and resource wastage	Power consumption and resource wastage	Improvement in both energy consumption and resource wastage	limited search space. This could lead to limited consolidation decisions
Kansal and Chana (2016)	FFO-EVMM	An energy-aware virtual machine migration technique for cloud computing, based on the Firefly algorithm that uses load sharing technique to reduce energy consumption	Energy consumption	Efficient in sharing workload to under loaded VMs	Time consuming
Ibrahim et al. (2018)	Adaptive GA	The authors developed an In-teger Linear Programming (ILP) model with Adaptive Genetic Algorithm (GA to provide a near optimal scheduling that minimizes energy consumption	Energy consumption and Time	It generate fair schedules with very low time complexity	Stuck in local optimal solution
Azad and Navimipour (2017)	Hybrid Cultural and ACO	A hybrid approach combined Culture and ACO algorithm to optimize both makespan and energy consumption	Makespan and energy consumption	Faster than cultural and ACO algorithms	Limited search space
Gabaldon et al. (2017)	MPSO-FGA	MPSO-FGA, a combination of PSO and GA to generate near-optimal solutions that minimizes scheduling time and energy consumption	Energy consumption and makespan	Decreases energy and Makespan better than the compared algorithms	Less scalable
Lawanyashri et al. (2017)	FOA-SA- LB	FOA-SA- LB approach combined the FOA technique for better exploration and exploitation, and integrated SA to update the search positions so that FOA does not converge prematurely	Makespan and energy consumption	Faster convergence in comparison with other methods	Less efficient with large search space
Garg et al. (2019)	REEWS	The authors explored reliability issues and presented REEWS algorithm that uses priority calculation, clustering of tasks, distribution of target time and assigning the cluster to processing element with appropriate voltage/ frequency levels to reduces application lifetime reliability and energy consumption	Reliability and energy consumption	Improved system efficiency	Limited parameters
Patel and Patel (2017)	HUA	HUA is presented to evict more unused or underloaded hosts, resulting in energy savings	Energy consumption	Overcomes VM overloaded and underloaded problem	It requires more time to execute workflows
Xu et al. (2015)	EnReal	The approach uses a dynamic VM provisioning technique to deploy VMs with a resource allocation table to keep track of all resources leading to equitable resource distribution	Resource Utilization and energy consumption	improve resource utilization and energy consumption rate in datacenter	Workflow and user heterogeneity is ignored.
Juarez et al. (2018)	MHRA	An efficient energy and makespan reduction method, that provide both cloud service providers and end users with different options for workflow task execution to improve task scheduling performance	Energy consumption and makespan	It provide better scheduling options for both cloud users and service providers	Less efficient in scheduling deadline contains tasks
Cao et al. (2014)	EARES-D	The authors applied multiple-step resource provision and allocation algorithm with Dynamic Voltage and Frequency Scaling (DVFS) and DNS methods to reduce energy	Energy, completing time and resource utilization	Efficient in performing VM allocation that shares workload evenly.	It requires more time to compute
The proposed algorithm	EViMA	The authors proposed resource management and VM allocation model and incorporated Idle ”off” Busy ”on” techniques that identify the status of all VMs. A slack time harvesting method is introduce, and finally task selection method that ensures the right task is allocated to the right VM	Energy consumption, Makespan and Cost	Efficient in managing cloud resources and generates faster solutions in comparison with other approaches	Not considering instance acquisition and termination delay

PMs with higher MIPS consume more energy than PMs with less MIPS. VMs are deployed on PMs and each PM can host k VMs. Let k be a set of VM types (vm_1, vm_2, vm_3, …..vm_k) provided to users at an hourly based pricing model. We assume that the VMs are infinite and each VM type k is capable of providing an infinite number of VMs for workflow tasks execution (Qin et al., 2020).

3.2. VM allocation model

Reducing energy consumption in a cloud data center has recently become a major concern for many researchers. This is because the cloud environment has become a trusted platform that individuals, organizations, and the government rely on to conduct business. So, if the energy consumption in the cloud data center is not reduced, it will have a direct or indirect impact on all the stakeholders. In this study, a VM allocation model has been proposed (see Fig. 2), which aims at managing cloud resources to reduce energy consumption in the cloud data center. In workflow scheduling, a scheduler is developed to assign the right tasks to the right VMs to improve the scheduling objectives. However, the biggest challenge is how to manage the cloud resources effectively to achieve the set goals. This is because cloud users turn in a large number of requests that require resources to schedule. This becomes even more obvious if these requests (tasks) are to be scheduled within certain deadlines. This challenge can be well managed if the scheduler is implemented with a well-defined resource management plan. In the model, we assume that there are only three types of VMs (VM_{HEC}, VM_{MEC}, and VM_{LEC}) in the cloud data center, which are specifically defined herein:

Definition 1. The VMs are deployed based on their energy consumption level. There are three (3) physical machines (PMs) hosting five VMs (three different types). Each VM type k has an infinite number of instances to handle multiple workflow tasks. The VMs are defined by specifications as follows:

$$VM\,Type = \begin{cases} VM_{HEC}(HEC_1),(HEC_2),(HEC_3).....HECk \\ VM_{MEC}(MEC_1),(MEC_2),(MEC_3).....MECk \\ VM_{LEC}(LEC_1),(LEC_2),(LEC_3).....LECk \end{cases}$$

- Where VM_{HEC} is a list of virtual machines with high processing speed (VMs with high energy consumption).
- Where VM_{MEC} represents a set of VMs with medium processing speed (VMs with medium energy consumption).
- Where VM_{LEC} is a group of virtual machines with lower processing speed (VMs with lower energy consumption).

Definition 2. Each workflow contains several set of workflow tasks representing (Wt_1, Wt_2, Wt_3, Wt_4 . ....., Wt_n); Wt is a set of 'n' workflow tasks in a scientific workflow application. We assume there are three compositions of workflow tasks which are defined herein:

$$\textbf{Tasks Grouping} = \begin{cases} HCT(HCT_1),(HCT_2),(HCT_3).....HCTn \\ MCT(MCT_1),(MCT_2),(MCT_3).....MCTn \\ LCT(LCT_1),(LCT_2),(LCT_3).....LCTn \end{cases}$$

- Where HCT is a set of tasks in the tasks ready queue that is defined as highly critical tasks that are at risk of missing their deadlines if they are not migrated to a high-speed procession VM immediately.
- Where MCT represents a group of queued workflow tasks that are classified as medium critical. This group of tasks is less critical compared to HCT, but more critical than LCT. These tasks can be migrated to medium speed VMs for execution.
- Where LCT is a group of tasks in the ready queue defined as less critical tasks. This category of tasks does not need to be processed on high-speed VMs for scheduling. The LCTs can be allocated on lower power VMs (VMs with lower processing speed) because their expected finish time (EFT) is far from their deadline. The EFT of workflow $task_i$ on VM type k can be calculated using Equation 1. Where S(wti) is the size of workflow $task_i$, EC (VMk) is the execution capacity of VM type k (Haidri et al., 2017). This represents the expected schedule.

$$EFT_{wti,VMk} = \frac{S(wti)}{EC(VMk)} \Longleftrightarrow \begin{cases} Expected \\ Schedule \end{cases} \quad (1)$$

Definition 3. Every workflow task has a deadline. A deadline is a time-based scheduling constraint that requires a workflow task to be scheduled within a specified time period. These deadlines are set by users. A task can have a shorter/earliest deadline or a longer deadline. A shorter/earliest deadline task is a task that is close to its specified scheduling time. Such tasks are assigned the highest priorities. Tasks with long deadlines are tasks that do not require higher priority. This is because such tasks are not close to their predefined deadlines and therefore can be assigned to resources after the higher priority tasks (shorter deadline tasks) have been scheduled. Thus, each task must be executed before the specified deadline. If the execution of a workflow task T exceeds its deadline, it means that the deadline constraint set by the user has been violated.

As shown in Fig. 2, we have four workflow tasks that have arrived and are queued to be mapped to VMs. The arrived tasks are wt_1, wt_2, wt_3, and wt_4. We also have three types of VMs (VM_{HEC}, VM_{MEC}, and VM_{LEC}). These tasks have different deadlines, and we need to map them to VMs based on their deadlines. wt_1 belongs to the HCT group because its deadline is 2 min, which is too tight, and if it is not allocated immediately, it might miss its deadline. To prevent wt_1 from missing its deadline, we quickly mapped it to VM_{HEC}. The reason is that VM_{HEC} is a high-speed VM that can finish executing wt_1 before or on its deadline. The next task closer to its deadline is wt_4, and since VM_{HEC} is currently executing wt_1, we mapped wt_4 to VM_{MEC}. The projections show that by the time wt_1 will meet its deadline, wt_3 will have 3 min left to also meet its deadline (5min-2 min). Therefore, wt_3 is classified as an HCT at that time, and if there is no other HCT in the waiting queue at that time, then wt_3 is mapped to VM_{HEC} and wt_2 is mapped to VM_{MEC}. In this case, VM_{LEC} is idle. The idle VM is turned off or put into safe mode to save power.

3.3. Multi-objective optimization problem formulation:

The goal of workflow scheduling is to find the best optimal solution from various optimization methods that can give good results for the optimization problem. The optimization problem can be single objective or multi objective optimization problem (Sofia and GaneshKumar, 2018). Single-objective optimization involves optimizing only one scheduling objective, such as execution makespan or cost. Multi-objective optimization, on the other hand, focuses on optimizing two or more objectives. The problem of multi-objective optimization varies from user to user and must be managed accordingly. Some users may want a reduced execution makespan and execution cost while ensuring that QoS constraints are not violated, others may want to reduce energy consumption and task execution time while ensuring effective utilization of cloud resources. In this work, a multi-objective scheduling problem has been considered.

We consider a workflow application as DAG. It is modeled with data dependencies representing precedence constraints between tasks. The edge e(wt_i, wt_j) indicates that wt_i is a direct predecessor of wt_j and should finish execution before the workflow wt_j, which is a direct successor of wt_i (Li et al., 2015). These tasks represent requests from the users to be processed on cloud resources (VM_{HEC}, (VM_{MEC}, VM_{LEC}) so that the energy consumption, makespan and execution cost are reduced without violating the deadlines of the tasks. However, reducing energy consumption increases the execution cost of scientific workflow applications. This is because energy consumption and execution cost are conflicting metrics. Our goal is to reduce the energy consumption,

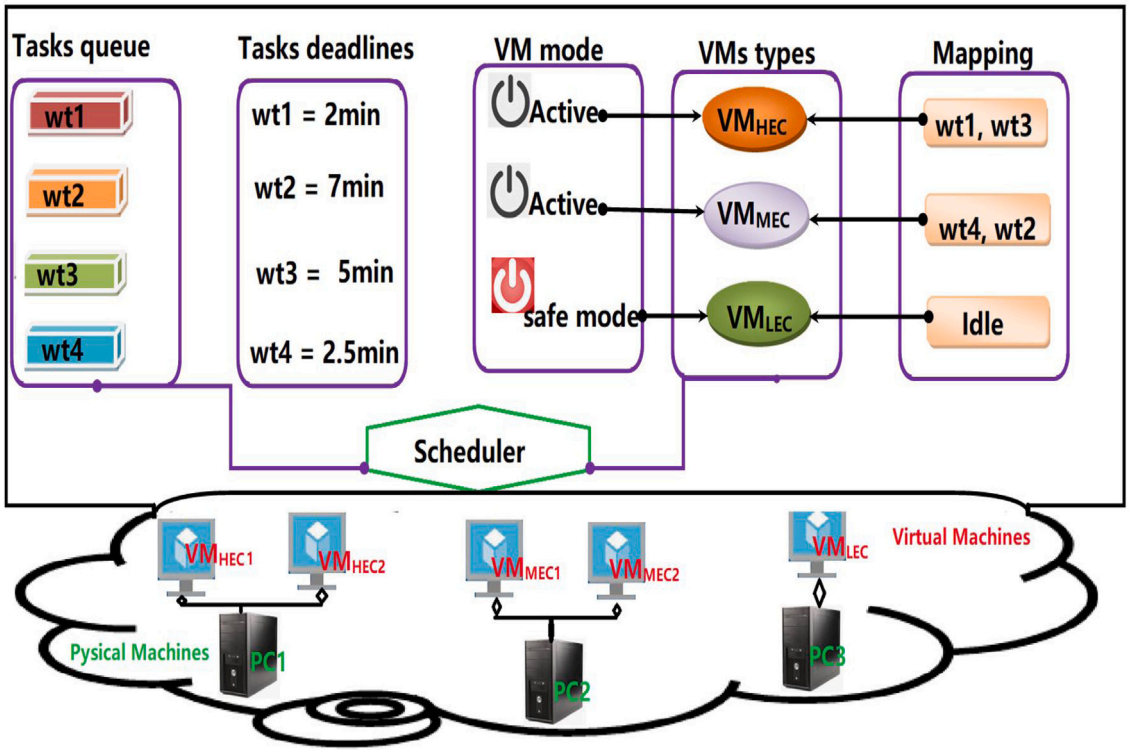

Fig. 2. VM Allocation Model.

makespan and execution cost of workflow tasks with dependency constraints. The degree of dependency of workflow task on wt_i can be calculated using Eq. (2) (Anwar and Deng, 2018).

$$dd_i = \left(\sum_{wt_p \epsilon\, pred_{(wt_i)}} ewt_{p,i} + \sum_{wt_c \epsilon\, Succ(wt_i)} ewt_{c,i} \right) \tag{2}$$

where $\sum_{wt_p \epsilon\ pred_{(wt_i)}} ewt_{p,i}$ is the summation of all the inward edges of workflow task wt_i and $\sum_{wt_c \epsilon Succ_{(wt_i)}} ewt_{c,i}$ is the summation of all the outward edges of workflow task wt_i.

Definition 4. Since the paper aims at improving resource management to reduce energy consumption in cloud data centers at lower cost without violating scheduling constraints, energy consumption can be derived from Eq. (3). Here, TE_E is the total energy consumed for executing workflows, E_{DT} is the energy consumed for transferring all data, and the sum of energy consumed for scheduling all tasks on all VM.

$$TE_E = ED_E + \sum_{Ek}^{En} \tag{3}$$

Definition 5. The metric for the execution cost is calculated using Eq. (4), where TC is the total cost, P_j is the processing time of job j, PC is the processing cost, and TrC is the cost of transferring the input files (Fin_j) and the output files ($Fout_j$).

$$TC = \left(P_j * PC + \left(\sum_{f \epsilon Finj} Size(f) + \sum_{f \epsilon Foutj} Size(f) \right) \right) * TrC \tag{4}$$

4. Proposed energy-efficient VM mapping algorithm (EViMA)

The existing workflow scheduling solutions for energy conservation in cloud data center faces scheduling efficiency problems and therefore, there is a need to introduce more robust scheduling solutions to improve data center energy management (Kaur and Mehta, 2017). In view of this objective, this paper presented a resource management approach for workflow tasks scheduling called EviMA. The proposed approach focuses on improving resource management in the cloud data center to obtain the best task schedules that reduces energy consumption, execution makespan and execution cost without violating deadline constraints. The EViMA algorithm is supported by four (4) other algorithms, namely Highly Critical Workflow Task Selection Algorithm (HiCTSA), Low Critical Task Selection Algorithm (LoCTSA), VM Power Regulating Algorithm (VM-PRA) and Slack Time Harvesting Algorithm (STiHA). The algorithm takes as input a set of workflow tasks represented by wt = (wt1, wt2, wt3, ...wtn) and a set of instances (VMs) represented by (VM_{HEC}, VM_{MEC}, and VM_{LEC}). The "n" workflow tasks can be any type of task in a queue that requires resources for scheduling. The algorithm starts by identifying the arrived tasks and computing their expected finishing times (EFT) to determine the task execution order. Then, a task priority queue is created in which the tasks are arranged in three execution orders based on their critical times (HCT, MCT, and LCT) as defined in Section 3. To determine the execution order of the tasks, their expected finishing times are calculated using Eq. (1). The tasks that are close to their deadline and need to be mapped to VMs at or before their deadline are given higher priority. To generate cost-efficient and energy-optimized schedules, a VM assignment order is also introduced (Line 5). In this work, we assumed that there are only three types of VMs (VM_{HEC}, VM_{MEC}, and VM_{LEC}). See Section 3 for a complete description of the VM types. Once both the tasks execution order and VM assignment order is established, we employ EViMA to explore through the various workflow tasks to find the overall critical tasks (HCTs), (tasks that are approaching their deadline) from the ready queue and immediately assign them to a high speed procession VM (VM_{HEC}). This is done to find high-quality solutions to the workflow scheduling problem. For example, if wt_i through the searching process is found to belong to the HCT queue, Algorithm 2 is called to assign it to a VM with high processing speed (VM_{HEC}), as described in line 8. However, if the task belongs to the LCT group, Algorithm 3 is called to process it, as described in line 9. On the other hand, if the expected finishing time of the task is less than

Algorithm 1: EViMA

Input : Workflow, set of VMs and set of VM types (VM_{HEC}, VM_{MEC}, VM_{LEC})

Output:

```
1  wt_ReadyPool = (wt1, wt2, tw3,.....wtn)
2  while wt_ReadyPool ≠ φ do
3      Compute EFT of each tasks
4      Group Tasks ⟹ HCT, MCT, LCT
5      Group VMs ⟹ VM_HEC, VM_MEC, VM_LEC
6      foreach wt in wt_ReadyPool do
7          if wt_i ∈ HCT and wt_j ∈ LCT then
8              apply algorithm 2 to execute HCT
9              apply algorithm 3 to execute LCT
10         else
11             if EFT of wt_i = Dl of wt_i then
12                 wt_i ↦ VM_MEC
13                 Update wt_ReadyPool
14             else
15                 if EFT of wt_i ≤ ST then
16                     apply algorithm 5
17                 else
18                     apply algorithm 4 to save mood the idle VM
19                     Update wt_ReadyPool
20                     While wt_ReadyPool ≠ φ
21                     Repeat step 6 to step 18
22                      if wt_ReadyPool = φ
23                 end
24             end
25         end
26     end
27 end
```

or equal to the harvested slack time of VM_k, Algorithm 5 is applied to schedule the task as in lines 15 and 16. If the expected end time is equal to the deadline, assign the task directly to VM_{MEC} as in lines 11 and 12. Otherwise, apply Algorithm 4 to shut down the VM when one of the VMs is idle to save power (line 18).

4.1. Highly critical task selection algorithm (HiCTSA)

Workflows are inherently complex. A single workflow application may contain multiple workflow tasks with dependencies and deadlines. This raises the question of which workflow tasks should be selected first and to which VM. To solve the task selection and assignment problem, we proposed a Highly Critical Task Selection Algorithm (HiCTSA), presented as Algorithm 2. HiCTSA works at both the VM and task levels to ensure that tasks are selected and assigned to resources for execution to avoid delays. We consider a set of workflow tasks WT = (wt1, tw2, tw3.....wtn) in a workflow application. These tasks have different deadlines, and therefore users expect them to be executed before their deadlines. Also, these tasks require some resources to start the scheduling process. Algorithm 2 starts by checking the criticality of the tasks as in line 1, and then compares the expected finishing time of each workflow task in the ready queue (line 2). If the expected finishing time of workflow task wt1 approaches its deadline, this task is marked as an HCT task (line 4). A mapping is then created for all HCT (line 5). However, if two or more tasks have the same deadline, another high speed VM is deployed to map those tasks (line 7 and 8). The goal of HiCTSA is to ensure that all workflow tasks meet their deadlines.

4.2. Low critical task selection algorithm (LoCTSA)

The cost associated with executing workflow tasks has become one of the biggest challenges in cloud data centers. This is because most scheduling techniques lack proper task selection and resource allocation mechanisms to assign the right task to the right resources for execution. Because of this, non-critical workflow tasks are assigned to high-speed resources, which drives up the execution cost of workflow tasks. We can reduce the execution cost by using the Low Critical Task Selection Algorithm (LoCTSA) proposed in Algorithm 3 in this paper. LoCTSA consists of two phases. The first phase is the task identification and grouping phase, in which less critical tasks are identified and queued for execution. In the first phase, the algorithm compares the expected finishing time of each workflow task with its deadline. If a task's EFT is greater than its deadline, the task is moved to a new queue called the LCT queue (lines 1, 2, and 3). This is done to avoid mapping less critical tasks to HEC VM. The second phase is the VM allocation phase. In the second phase, the quality of the solution is improved by ensuring that all less critical tasks are assigned to VMs with lower processing speed (VMs with lower cost and lower energy consumption) instead of assigning them to VMs with higher execution cost. This is done to reduce the cost of running workflow tasks.

4.3. VM power regulating algorithm (VM-PRA)

We introduced a new method called VM-PRA (see Algorithm 4) to check the status of all VMs used to execute workflow tasks. Sometimes,

Algorithm 2: HiCTSA

Input : • Workflow DAG, G=(W,T) where W= (wt_1 ,wt_2 ,wt_3 ,.....wt_n)
• A set of cloud resources with different CPU configuration and different pricing

Output:

```
Check tasks criticality in wt_ReadyPool
Compare EFT of wt_1 on VM_k to its Dl
if EFT of wt_1 > Dl of wt_1 then
    Call wt_1 HCT
    Map all HCT to VM_HEC
else
    if DLwt_i==DLwt_j
    deploy VM_HEC2
    If EFT of wt_i = Dl of wt_i, then
        Call the task MCT
        Update wt_ReadyPool
        while wt_ReadyPool ≠ ϕ do
            Repeat step 3 to 8
        end
        if wt_ReadyPool = ϕ
    else
end
```

Algorithm 3: LoCTSA

```
Compare EFT of wt_1 on VM_k to its Dl
if EFT of wt_1 < Dl of wt_1 then
    Call wt_1 LCT
    Map all LCT to VM_LEC
else
    if EFT of wt_i = Dl of wt_i, then
        move wt_i into MCT queue
        Update wt_ReadyPool
        while wt_ReadyPool ≠ ϕ do
            Repeat step 2 to 8
        end
        if wt_ReadyPool = ϕ
    else
end
```

underutilized VMs are not shut down, which can increase the power consumption of a data center. Our proposed VM-PRA overcomes this drawback by introducing Idle "off" Busy "on" techniques that identify the status of all VMs. The method starts by defining the status of all VMs that are used for workflow tasks execution. The VMs are divided into two stages, namely a busy stage and an idle stage. A VM is said to be busy when it is currently executing or processing a workflow task. Idle VMs, on the other hand, are VMs that have not yet been assigned any job or have completed the execution process and are waiting for more jobs to be executed. The algorithm checks, if there are some idle VMs, and if so, it switches all idle VMs to the memory state. For example, when VM_{HEC} and VM_{LEC} are running and VM_{MEC} is idle, the VM-PRA algorithm quickly switches VM_{MEC} into power saving mode to save some energy. In reality, VMs do not consume power, but when the VM is powered off or put into power saving mode, it reduces the power consumption of its PM, and hence a reduction in data center power consumption. For more information on how an inactive VM can be put into energy-saving mode, refer to Definition 3 of the VM deployment model.

4.4. Slack time harvesting method (STiHaM)

The Slack Time Harvesting Method is presented in Algorithm 5. Slack time is a time interval that occurs when a particular task completes its execution before its actual scheduled completion time. In other words, it is a time difference between a task's scheduled completion time and its actual completion time (Mezmaz et al., 2011). The

Algorithm 4: VM-PRA

Output:

$VM_{Pool} = (VM_{HEC}, VM_{MEC}, VM_{LEC}, ...VM_k)$
foreach *VM in the* VM_{Pool} **do**
  Check VMs status (busy or idle)
  if VM_{HEC} *is idle* **then**
    Put VM_{HEC} on save mood
  else
    if VM_{MEC} *and* VM_{LEC} *are idle* **then**
      Put VM_{MEC} and VM_{LEC} on save mood
      Update VM_{Pool}
      while $VM_{Pool} \neq \phi$ **do**
        Repeat step 2 to 8
      end
      if $VM_{Pool} = \phi$
    else
    end
end

Algorithm 5: STiHaM

Input : • Most workflow tasks have slack times
Output:

$Wt_{ReadPool} = (wt_1, wt_2, tw_3, ...wt_n)$
$VM_{Pool} = (VM_a, VM_b, VM_c, ...VM_K)$
foreach *VM in the* VM_{Pool} **do**
  Generate the EFT and ST of all tasks
  Harvest the slack time of each VM
  Store the slack times of each VM in ST_{Hub}
  Prompt the scheduler when EFT of $wt_{i,j}$ = ST of VM_k
  Allocate $wt_{i,j}$ to utilize the slack time
  Update VM_{Pool}
  while $VM_{Pool} \neq \phi$ **do**
    Repeat step 4 to 8
  end
  if $VM_{Pool} = \phi$
end

slack time is used to accumulate the slack times for each VM type in the VM pool as in line 5 of Algorithm 5. A slack time hub is attached to each VM to store the slack times of each VM (see line 6). If a slack time of VM_k is equal to the EFT of wt_i, the scheduler is asked to map wt_i to VM_k to use the slack time as in lines 7 and 8 of Algorithm 5. The slack time is used to further reduce the execution cost of the workflows. Since wt_i can be scheduled on VM_k using the obtained slack time, the cost of provisioning a new VM to schedule the tasks is saved.

5. Performance evaluation

In this section, we detailed how the proposed EViMA is tested through simulation runs. To evaluate the performance of the proposed work, extensive simulation experiments with workflowsim were conducted in this study. We compared the results of our algorithm with the state-of-the-art algorithms such as ERES (Garg et al., 2021) and EERS (Medara and Singh, 2021) for energy consumption. For execution makespan and cost, the proposed method is compared with JIT-C (Sahni and Vidyarthi, 2015) and HSLJF (Alworafi et al., 2019) algorithms. The evaluation was carried out to show that:

- EViMA is efficient in managing cloud resources.
- Our approach is capable of reducing energy consumption in cloud data centers
- The proposed method gives better results in makespan and execution cost than the state of the art works

5.1. Datasets used

The performance analysis of the proposed EViMA was performed using three different performance matrices (energy consumption, makespan and execution cost) over five (5) real scientific workflows from different scientific domains with different computation and communication characteristics as described in Juve et al. (2013), Bharathi

et al. (2008). The workflows used are Montage, LIGO Inspiral, Cybershake, Sipht, and Epigenomics, which include 25–1000 task nodes, as used in Alaei et al. (2021), Han et al. (2021). The Montage workflow application is an astronomical application used to create large mosaics of the sky based on a set of input images. Montage tasks are characterized by a data-intensive workflow application that requires a lot of time for data transfer. The LIGO inspiral workflow application is used in the field of astrophysics to detect gravitational waves for generating gravitational activity on Earth. LIGO requires a lot of memory as it is an CPU intensive workflow application (Mboula et al., 2020). CyberShake is used in earthquake research to characterize the earthquake hazard in a region. The CyberSnake workflow tasks are very complex and require more CPU and memory to process the data. The Sipht workflow is used in the field of bioinformatics science to computerize the sRNA coding genes. The Epigenomics workflow is used in bioinformatics to automate the sequence of events in the genome of human cells. It is a CPU intensive pipeline application that computes a large amount of data in parallel and requires higher systems with large memory for processing. These workflows have been widely used by many researchers in the field of workflow scheduling to evaluate and compare the performance of scheduling heuristics. Each of these workflow types has four different workloads. For example, the Cybershake workflow has four workloads, including 30, 50, 100, and 1000 (Yao et al., 2016).

5.2. Experimental setup

In this section, both the hardware and software configuration environments for the simulation have been described in detail. The hardware configuration is Intel(R) Core(TM) i5-8210Y, CPU @3.2 GHz, 8 GB memory. The software environments used are Windows 10, Eclipse 3.5 and WorkflowSim-1.0 toolkit. We chose to test the proposed approach in the WorkflowSim toolkit rather than in the real cloud because evaluating the performance of optimization algorithms in the real cloud environment is complex, time consuming and expensive (Dong et al., 2021; Chen and Deelman, 2012). WorkflowSim is an extension of cloudSim that allows workflow scheduling algorithm developers to simulate scheduling algorithms. The inputs to the simulation environment are as follows: (1) the average bandwidth between resources is 20 MBps as in Arabnejad et al. (2018), Mboula et al. (2020), which is the average bandwidth setting offered by Amazon Web Services (Palankar et al., 2008; Sahni and Vidyarthi, 2015), (2) the processing matrix for each VM is measured in Million Instruction Per Second (MIPS) as in Rodriguez and Buyya (2018), Singh et al. (2019), Adhikari and Amgoth (2019), (3) the task lengths are set in Million Instruction (MI) as in Singh et al. (2019). In this experiment, the job that is close to its deadline is selected first and submitted to a VM with high processing speed for execution. The sample structures of the workflows are shown in Fig. 3 and the characterization including the number of nodes, edges and average date size of each workflow is given in Table 2, as used in Juve et al. (2013), Zhou et al. (2019) and Zhu et al. (2015).

We assume that there are only three (3) types of workflow tasks, including HCT, MCT, and LCT, as described in Section 3 Definition 2. These tasks are given as follows:

- **Tasks HCT** = HCT(HCT_1), (HCT_2), (HCT_3).....HCT n
- **Tasks MCT** = MCT(MCT_1), (MCT_2), (MCT_3).....MCT n
- **Tasks LCT** = LCT(LCT_1), (LCT_2), (LCT_3).....LCT n

These tasks arrived at different intervals and with different deadlines. To execute these workflow tasks, virtual machines are required. We further assumed that there are only three (3) types of VMs in the cloud data center, each with an infinite number of instances. The computational capacities and the prices per hour for each VM type are set as the work in Li et al. (2015), Han et al. (2021), Wu et al. (2017) and are shown in Table 3. The instant capacity selection and pricing model are based on Amazon EC2. For more information about VM type, please refer to section three (3) Definition 1. In this research, we classified VMs into the following three groups:

Table 2
List of notations used.

Notations	Definitions
EViMA	Energy Efficient Virtual-Machine Mapping Algorithm
VMs	Virtual Machines
HiCTSA	Highly Critical Workflow Task Selection Algorithm
LoCTSA	Low Criticality Workflow Task Selection Algorithm
VM-PRA	VM Power Regulating Algorithm
STiHaM	Slack Time Harvesting Method
VM_{HEC}	VMs with high energy consumption
VM_{MEC}	VMs with Medium Energy Consumption)
VM_{LEC}	VMs with Lower Energy Consumption
HCT	Highly Critical Tasks
MCT	Medium Critical Tasks
LCT	Less Critical Tasks
S(wti)	size of Workflow $Task_i$
EC(VMk)	Execution Capacity of VM type k
EFT	Expected Finishing Time
TE_E	Total Energy Consumed on Executing Workflows
E_{DT}	Energy Consumed for the transmission of all data
ST	Slack Time
ST_{Hub}	Slack Time Hub

Table 3
Characteristics of the five Benchmarks Workflows.

Workflow application	Number of nodes	Number of edges	Average data size (MB)
Montage-25	25	95	3.43 MB
Montage-50	50	206	3.36 MB
Montage-100	100	433	3.23 MB
Montage-1000	1000	4485	3.21 MB
LIGO-30	30	95	9.00 MB
LIGO-50	50	160	9.16 MB
LIGO-100	100	319	8.93 MB
LIGO-1000	1000	3246	8.90 MB
CyberShake-30	30	112	747.48 MB
CyberShake-50	50	188	864.74 MB
CyberShake-100	100	380	849.60 MB
CyberShake-1000	1000	3988	102.29 BM
Sipht-30	30	91	7.73 MB
Sipht-60	60	198	6.95 MB
Sipht-100	100	335	6.27 MB
Sipht-1000	1000	3528	5.91 MB
Epigenomics-24	24	75	116.20 MB
Epigenomics-46	46	148	104.81 MB
Epigenomics-100	100	322	395.10 MB
Epigenomics-997	997	3228	388.59 MB

- **VM HEC** = VM_{HEC}(HEC_1), (HEC_2), (HEC_3).....HEC k
- **VM MEC** = VM_{MEC}(MEC_1), (MEC_2), (MEC_3).....MEC k
- **VM LEC** = VM_{LEC}(LEC_1), (LEC_2), (LEC_2).....LEC k

6. Results and discussions

In order to evaluate the performance of the solutions generated by the proposed algorithm and the benchmark algorithms, we conducted three experiments. The first experiment shows the performance of the three algorithms in terms of energy consumption, the second experiment evaluates the performance of the algorithms in terms of execution makespan and the third experiment evaluates the algorithms in terms of execution cost. The results for each of the performance metrics are presented below in Table 4.

6.1. Evaluation based on energy consumption

Energy consumption refers to the total amount of energy consumed in a data center to schedule a workflow task. We compared the results of our algorithm with two different sets of state-of-the-art-works such as ERES and EERS. We considered five workflows including Montage, CyberShake, Epigenomics, LIGO, and SIPHT with four different types

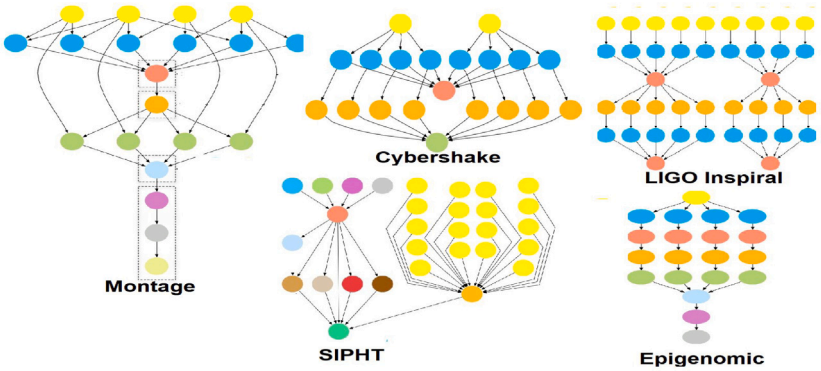

Fig. 3. Structure of Scientific Workflows used (Mehta et al., 2009).

Table 4
VM types and pricing model.

Virtual machine type		Capability (CU)	Price ($/h)
LEC	1	1.0	0.12
	2	1.5	0.195
MEC	3	2.0	0.28
	4	2.5	0.375
HEC	5	3.0	0.48
	6	3.5	0.595

of datasets such as small data, medium data, large data, and extra-large data. The results of the energy consumption metric are shown in Figs. 4, 5, 6, and 7. Eq. (3) is used to calculate the energy consumption of each VM in an active PM. In the small workflow tasks size (as shown in Fig. 4), it is obvious that the proposed EViMA consumes less energy compared to ERES and EERS algorithms. Moreover, EViMA can save more energy than ERES and EERS in all the five workflows when medium size workflow tasks are scheduled (see Fig. 5). The experimental results for large and extra-large workflow tasks for the proposed and existing algorithms are shown in Fig. 6 and Fig. 7, respectively. A careful look at the results shows an increasing trend in energy consumption for all the three algorithms when large and extra large workflow tasks are used. However, it can be observed that the EViMA algorithm consistently schedules each workflow task with lower energy consumption than the other two algorithms. In fact, each of the three algorithms performed differently on each of the five workflows and on each workflow task size. Among the algorithms, ERES performs worse than the other algorithms on almost all the four workloads, except for extra-large workflows tasks, where ERES performs better than EERS. This is because ERES is not able to manage the cloud resources during workflow scheduling in small, medium and large workflow tasks, which makes it difficult for ERES to effectively reduce the energy consumption in the cloud data center.

6.2. Evaluation based on execution makespan

Makespan is the finishing time of executed tasks on cloud resources. This is important because most cloud users want their tasks to be executed on cloud resources at a specific time to prevent tasks from missing their deadlines (Alworafi et al., 2019). The makespan incurred by the algorithms are diverse (see Figs. 8–11). It can be observed that EViMA produces shorter schedules in almost all the five workflows. However, EViMA fails to produce shorter schedules in the small-sized workflow tasks of Montage, Cybershake and in the medium-sized workflow tasks of Ligo. This is because the users of the small-sized workflow tasks of Montage, Cybershake, and the medium-sized workflow tasks of Ligo do not care about the time it takes to schedule their tasks. Their focus is on energy and execution costs. It is also observed that as the number of workflow tasks increases, the time taken by all the algorithms increases significantly to correspond to the number of tasks for all the workflows, as shown in Figs. 10 and 11. This is because as the number of workflow tasks increases, more cloud resources are required to execute more tasks. However, due to the limited number of cloud resources, some of the tasks need to be queued for some time so that others can finish execution, which increases the execution makespan. For example, if the number of workflow tasks for montage increases from 100 to 1000, the execution makespan of all algorithms increases by more than 500%. It is also observed that among all the workflows, the epigenomics workflow takes more time to generate a schedule. This is because the epigenomics workflow is a CPU-intensive application that requires more time to execute the workflow tasks. The overall performance improvement of the proposed algorithm for the makespan parameter is significant compared to the other algorithms. This is because the proposed method is able to manage VM deployment dynamically to avoid delays in task scheduling processes.

6.3. Evaluation based on execution cost

The execution cost of a workflow task is defined as the total payment that a cloud user makes to the cloud provider on a pay-as-you-go basis to get schedules generated on cloud resources (Madni et al., 2017). The execution cost of a workflow task is calculated by summing the cost of the processor, the communication cost between the processors, and the size of the workflow (data size) (Gupta et al., 2019). To better understand the performance of the proposed algorithm, we analyzed and compared the execution cost results for EViMA, JIT-C and HSLJF using five different scientific workflows for small, medium, large and extra-large workloads (see Figs. 12–15). The results in Fig. 12 show the execution cost achieved by each of the three algorithms when using small size workflow tasks (small workload). Compared to JIT-C and HSLJF, EViMA has well managed the distribution of all workflow tasks on cloud resources, making it a cost-effective scheduler. On the other hand, except for Ligo workflow tasks, JIT-C has performed better than HSLJF for the other workflow tasks. From the results in Fig. 12, we can conclude that our proposed method can satisfy cloud users at a lower cost when scheduling small workflow tasks.

We also compared the execution cost of the three algorithms in Fig. 13 with the medium-size workflows. It is evident from the results that, the proposed method produces minimum execution cost when medium-size workflows are scheduled. For each workflow type, the proposed EViMA achieves better results than JIT-C and HSLJF. Figs. 14 and 15 present several instances of results for the three algorithms for large and extra-large workflows respectively. As seen from Figs. 14 and 15, among the three algorithms, the proposed EViMA obtains good performance for the execution cost values. However, there is a significant increase in execution cost for all the three algorithms when Epigenomics workflow is used. This is because the epigenoics workflow

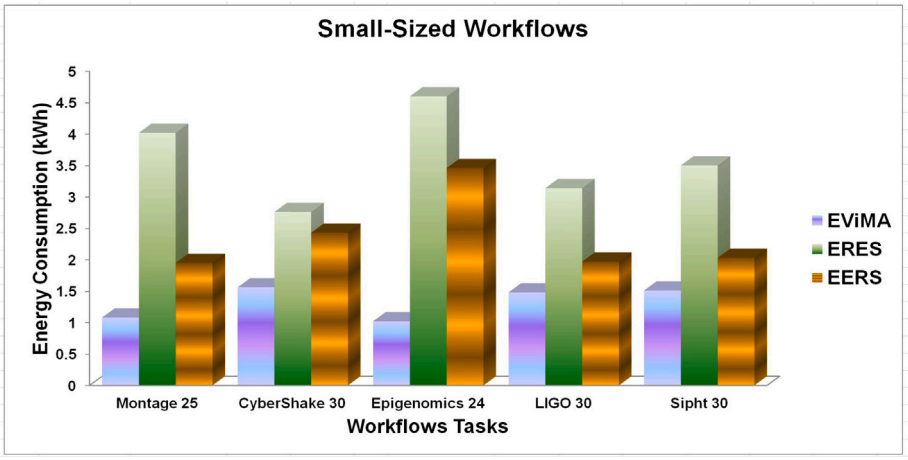

Fig. 4. Comparison of Energy Consumption in Small Sized Workflows.

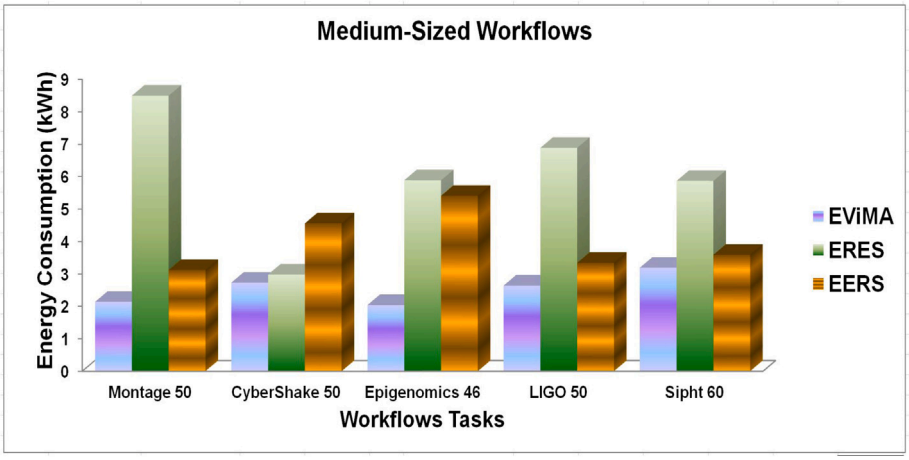

Fig. 5. Comparison of Energy Consumption in Medium Sized Workflows.

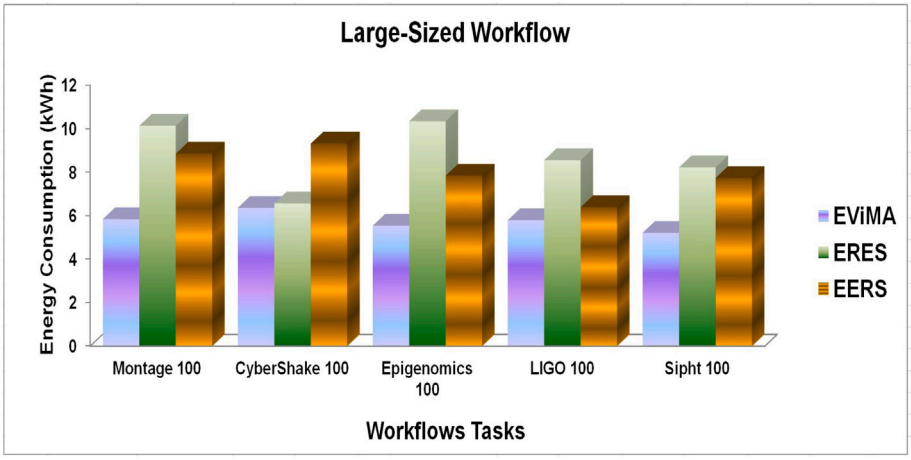

Fig. 6. Comparison of Energy Consumption in Large Sized Workflows.

is a CPU-intensive pipeline workflow that computes a large amount of data in parallel, and therefore will require VMs with higher memory to process. Notwithstanding, the execution cost of the proposed algorithm is still significantly lower than the other two algorithms. This is because the proposed algorithm employs a slack-time harvesting method, which allows it to use the harvested slack-time to schedule other tasks.

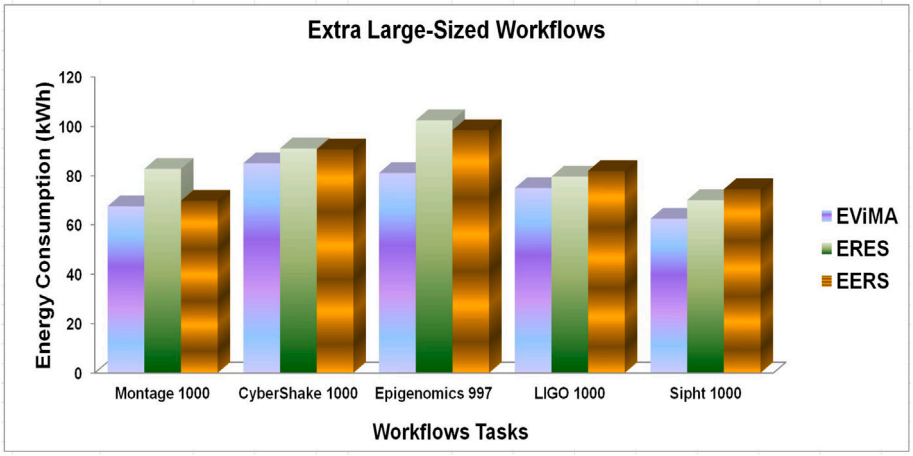

Fig. 7. Comparison of Energy Consumption in Extra-Large Sized Workflows.

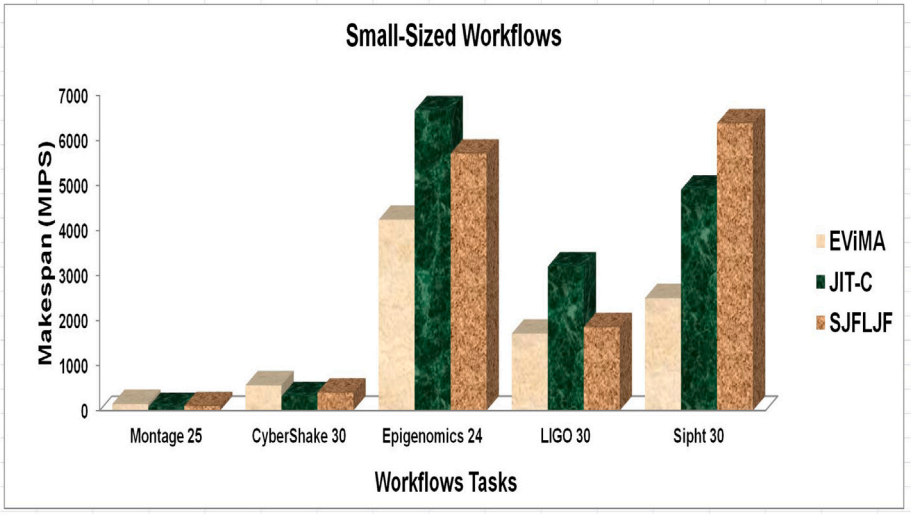

Fig. 8. Comparison of Makespan in Small Sized Workflows.

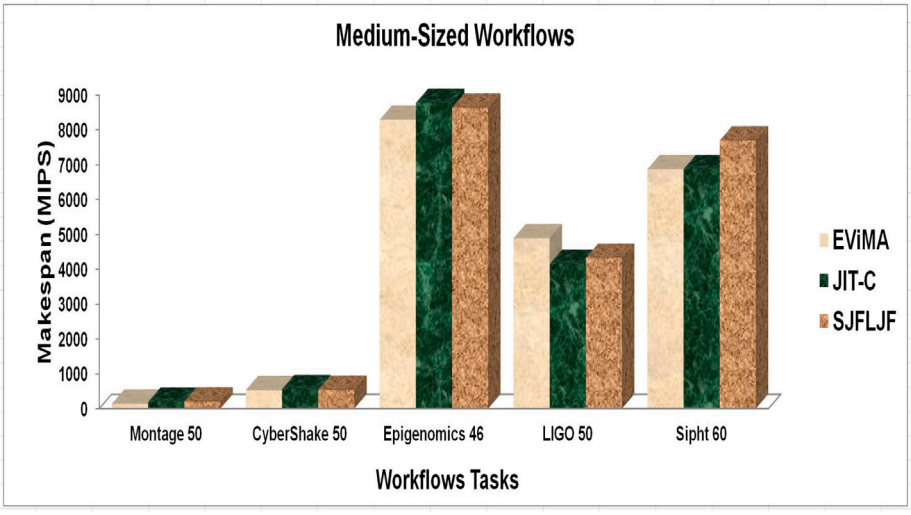

Fig. 9. Comparison of Makespan in Medium Sized Workflows.

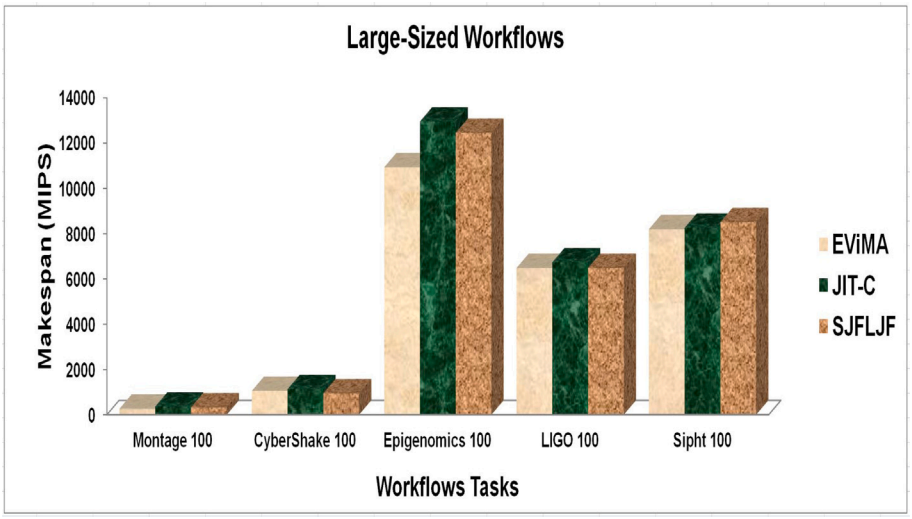

Fig. 10. Comparison of Makespan in Large Sized Workflows.

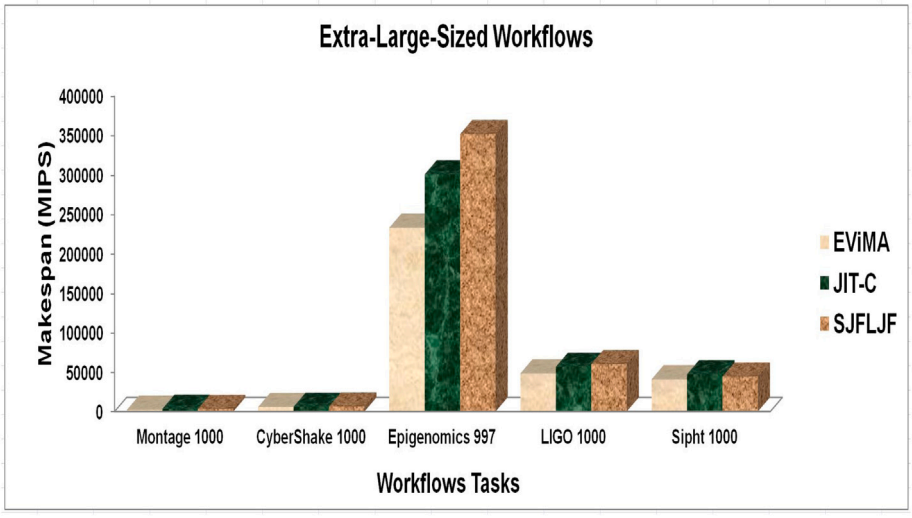

Fig. 11. Comparison of Makespan in Extra-Large Sized Workflows.

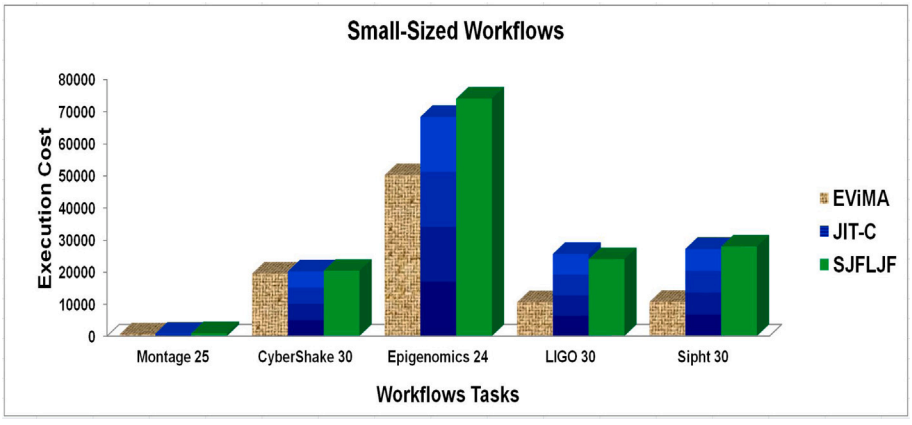

Fig. 12. Results of Execution Cost with Small Sized Workflows.

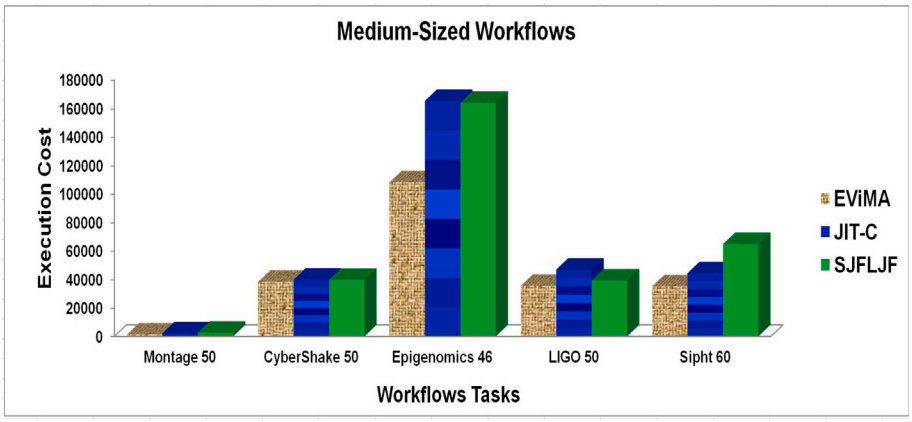

Fig. 13. Results of Execution Cost with Medium Sized Workflows.

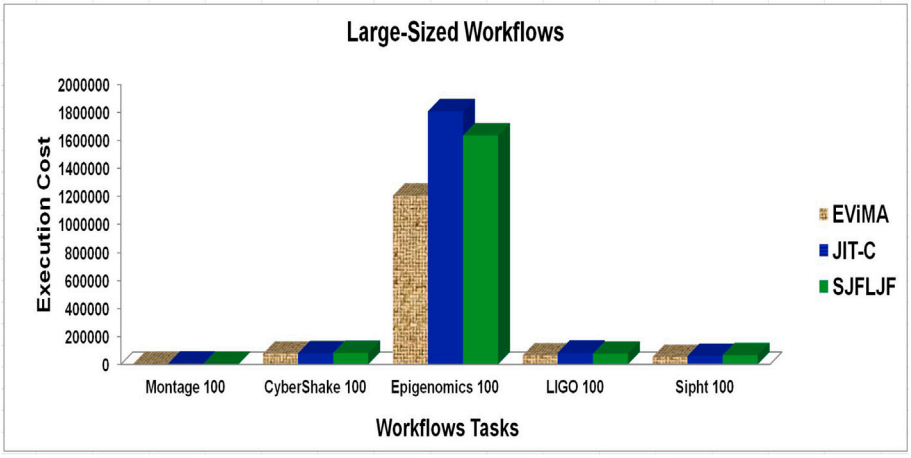

Fig. 14. Results of Execution Cost with Large Sized Workflows.

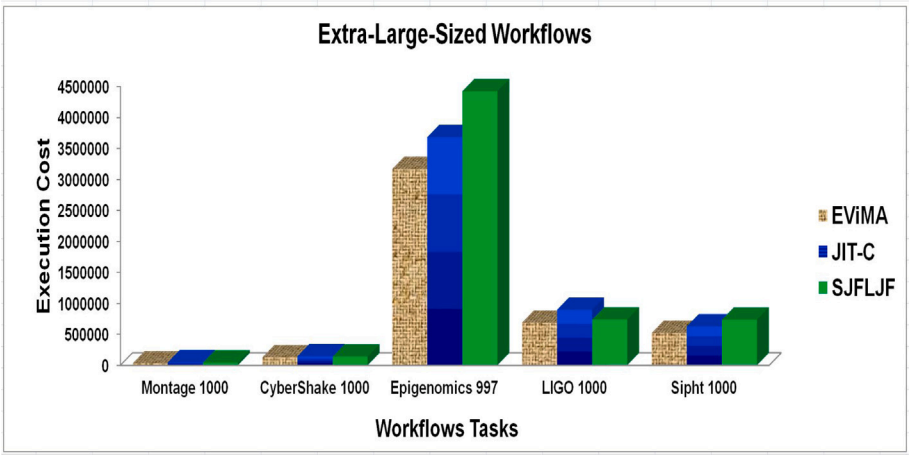

Fig. 15. Results of Execution Cost with Extra-Large Sized Workflows.

7. Conclusion and future work

Managing cloud resources to generate an effective schedule require improving the three most important, yet, conflicting scheduling objectives — energy consumption, execution makespan and execution cost. This paper exploit the existing workflow scheduling methodologies, particularly in energy consumption, makespan and execution cost, and presents a novel workflow scheduling algorithm that aimed at managing cloud resources to reduce energy consumption, makespan and execution cost. This will satisfy cloud user requirements and improve the quality of services offered by the cloud providers. The problem of the workflow high execution cost is dealt with by introducing a slack time harvesting method that allows other tasks to be scheduled using the harvested slack times. The algorithm also distributes workloads on cloud resources evenly through the use of VM-PRA, a resource management method that identifies idle VMs and turns them off to reduce energy consumption values in the cloud datacenter. The performance of the proposed EViMA was validated by conducting three experiments and comparing the results obtained with state-of-the-art-works. Five (5) real-world scientific workflows were used in different workload capacities — small, medium, large and extra-large workload tasks in the simulation process. The proposed EViMA drastically reduced the drawbacks in ERES, EERS, JIT-C and HSLJF algorithms and produced the best solution that manages the use of cloud resources better, to reduce energy consumption, makespan and execution cost.

This work is considering designing a hybrid scheduling scheme based on heuristic and meta-heuristic algorithms that orchestrate well with cloud resources to determine the most optimal VM configuration setting to be used. An optimal VM configuration depends on the chosen VM capacity, vis- à-vis the size of the workflow tasks. The size of the workflow tasks will determine the required number of VMs to speed up the execution time of workflows through parallelism. Furthermore, we intend to apply this work to microservices environment.

Declaration of competing interest

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.

Acknowledgment

This work was supported by the School of Computer Science, University College Dublin.

References

Adhikari, M., Amgoth, T., 2019. An intelligent water drops-based workflow scheduling for IaaS cloud. Appl. Soft Comput. 77, 547–566.

Al-Dulaimy, A., Itani, W., Zantout, R., Zekri, A., 2018. Type-aware virtual machine management for energy efficient cloud data centers. Sustain. Comput. Inform. Syst. 19, 185–203.

Alaei, M., Khorsand, R., Ramezanpour, M., 2021. An adaptive fault detector strategy for scientific workflow scheduling based on improved differential evolution algorithm in cloud. Appl. Soft Comput. 99, 106895.

Alkhanak, E.N., Lee, S.P., Rezaei, R., Parizi, R.M., 2016. Cost optimization approaches for scientific workflow scheduling in cloud and grid computing: A review, classifications, and open issues. J. Syst. Softw. 113, 1–26.

Alworafi, M.A., Dhari, A., El-Booz, S.A., Nasr, A.A., Arpitha, A., Mallappa, S., 2019. An enhanced task scheduling in cloud computing based on hybrid approach. In: Data Analytics and Learning. Springer, pp. 11–25.

Anwar, N., Deng, H., 2018. Elastic scheduling of scientific workflows under deadline constraints in cloud computing environments. Future Internet 10 (1), 5.

Arabnejad, V., Bubendorfer, K., Ng, B., 2018. Budget and deadline aware e-science workflow scheduling in clouds. IEEE Trans. Parallel Distrib. Syst. 30 (1), 29–44.

Azad, P., Navimipour, N.J., 2017. An energy-aware task scheduling in the cloud computing using a hybrid cultural and ant colony optimization algorithm. Int. J. Cloud Appl. Comput. (IJCAC) 7 (4), 20–40.

Babukarthik, R., Raju, R., Dhavachelvan, P., 2012. Energy-aware scheduling using hybrid algorithm for cloud computing. In: 2012 Third International Conference on Computing, Communication and Networking Technologies. ICCCNT'12, IEEE, pp. 1–6.

Barlaskar, E., Singh, Y.J., Issac, B., 2016. Energy-efficient virtual machine placement using enhanced firefly algorithm. Multiagent Grid Syst. 12 (3), 167–198.

Bharathi, S., Chervenak, A., Deelman, E., Mehta, G., Su, M.-H., Vahi, K., 2008. Characterization of scientific workflows. In: 2008 Third Workshop on Workflows in Support of Large-Scale Science. IEEE, pp. 1–10.

Cao, F., Zhu, M.M., Wu, C.Q., 2014. Energy-efficient resource management for scientific workflows in clouds. In: 2014 IEEE World Congress on Services. IEEE, pp. 402–409.

Chen, W., Deelman, E., 2012. Workflowsim: A toolkit for simulating scientific workflows in distributed environments. In: 2012 IEEE 8th International Conference on E-Science. IEEE, pp. 1–8.

Chen, J., John, L.K., 2009. Efficient program scheduling for heterogeneous multi-core processors. In: 2009 46th ACM/IEEE Design Automation Conference. IEEE, pp. 927–930.

Chen, H., Zhu, X., Qiu, D., Guo, H., Yang, L.T., Lu, P., 2016. EONS: minimizing energy consumption for executing real-time workflows in virtualized cloud data centers. In: 2016 45th International Conference on Parallel Processing Workshops. ICPPW, IEEE, pp. 385–392.

Dong, T., Xue, F., Xiao, C., Zhang, J., 2021. Workflow scheduling based on deep reinforcement learning in the cloud environment. J. Ambient Intell. Humaniz. Comput. 1–13.

Ferdaus, M.H., Murshed, M., Calheiros, R.N., Buyya, R., 2014. Virtual machine consolidation in cloud data centers using aco metaheuristic. In: European Conference on Parallel Processing. Springer, pp. 306–317.

Gabaldon, E., Vila, S., Guirado, F., Lerida, J.L., Planes, J., 2017. Energy efficient scheduling on heterogeneous federated clusters using a fuzzy multi-objective meta-heuristic. In: 2017 IEEE International Conference on Fuzzy Systems. FUZZ-IEEE, IEEE, pp. 1–6.

Garg, R., Mittal, M., et al., 2019. Reliability and energy efficient workflow scheduling in cloud environment. Cluster Comput. 22 (4), 1283–1297.

Garg, N., Singh, D., Goraya, M.S., 2021. Energy and resource efficient workflow scheduling in a virtualized cloud environment. Cluster Comput. 24 (2), 767–797.

Gupta, S., Agarwal, I., Singh, R.S., 2019. Workflow scheduling using jaya algorithm in cloud. Concurr. Comput.: Pract. Exper. 31 (17), e5251.

Haidri, R.A., Katti, C.P., Saxena, P.C., 2017. Cost effective deadline aware scheduling strategy for workflow applications on virtual machines in cloud computing. J. King Saud Univ.-Comput. Inform. Sci..

Han, P., Du, C., Chen, J., Ling, F., Du, X., 2021. Cost and makespan scheduling of workflows in clouds using list multiobjective optimization technique. J. Syst. Archit. 112, 101837.

Hieu, N.T., Di Francesco, M., Ylä-Jääski, A., 2015. Virtual machine consolidation with usage prediction for energy-efficient cloud data centers. In: 2015 IEEE 8th International Conference on Cloud Computing. IEEE, pp. 750–757.

Hu, H., Li, Z., Hu, H., Chen, J., Ge, J., Li, C., Chang, V., 2018. Multi-objective scheduling for scientific workflow in multicloud environment. J. Netw. Comput. Appl. 114, 108–122.

Ibrahim, H., Aburukba, R.O., El-Fakih, K., 2018. An integer linear programming model and adaptive genetic algorithm approach to minimize energy consumption of cloud computing data centers. Comput. Electr. Eng. 67, 551–565.

Juarez, F., Ejarque, J., Badia, R.M., 2018. Dynamic energy-aware scheduling for parallel task-based application in cloud computing. Future Gener. Comput. Syst. 78, 257–271.

Juve, G., Chervenak, A., Deelman, E., Bharathi, S., Mehta, G., Vahi, K., 2013. Characterizing and profiling scientific workflows. Future Gener. Comput. Syst. 29 (3), 682–692.

Kansal, N.J., Chana, I., 2016. Energy-aware virtual machine migration for cloud computing-a firefly optimization approach. J. Grid Comput. 14 (2), 327–345.

Kaur, P., Mehta, S., 2017. Resource provisioning and work flow scheduling in clouds using augmented Shuffled Frog Leaping Algorithm. J. Parallel Distrib. Comput. 101, 41–50.

Konjaang, J.K., Xu, L., 2021. Meta-heuristic approaches for effective scheduling in infrastructure as a service cloud: A systematic review. J. Netw. Syst. Manage. 29 (2), 1–57.

Kumar, M., Sharma, S., 2018. PSO-COGENT: Cost and energy efficient scheduling in cloud environment with deadline constraint. Sustain. Comput. Inform. Syst. 19, 147–164.

Lawanyashri, M., Balusamy, B., Subha, S., 2017. Energy-aware hybrid fruitfly optimization for load balancing in cloud environments for EHR applications. Inform. Med. Unlocked 8, 42–50.

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 (4), 713–726.

Madni, S.H.H., Latiff, M.S.A., Abdullahi, M., Abdulhamid, S.M., Usman, M.J., 2017. Performance comparison of heuristic algorithms for task scheduling in iaas cloud computing environment. PLoS One 12 (5).

Malekloo, M., Kara, N., 2014. Multi-objective ACO virtual machine placement in cloud computing environments. In: 2014 IEEE Globecom Workshops. GC Wkshps, IEEE, pp. 112–116.

Mboula, J.E.N., Kamla, V.C., Djamegni, C.T., 2020. Cost-time trade-off efficient workflow scheduling in cloud. Simul. Model. Pract. Theory 102107.
Medara, R., Singh, R.S., 2021. Energy efficient and reliability aware workflow task scheduling in cloud environment. Wirel. Pers. Commun. 1–20.
Mehta, G., Juve, G., Chen, W., 2009. Workflow generator. URL: https://Confluence.Pegasus. Isi. Edu/Display/Pegasus/WorkflowGenerator (11.06. 2016).
Mezmaz, M., Melab, N., Kessaci, Y., Lee, Y.C., Talbi, E.-G., Zomaya, A.Y., Tuyttens, D., 2011. A parallel bi-objective hybrid metaheuristic for energy-aware scheduling for cloud computing systems. J. Parallel Distrib. Comput. 71 (11), 1497–1508.
Palankar, M.R., Iamnitchi, A., Ripeanu, M., Garfinkel, S., 2008. Amazon S3 for science grids: a viable solution? In: Proceedings of the 2008 International Workshop on Data-Aware Distributed Computing. pp. 55–64.
Panda, S.K., Jana, P.K., 2019. An energy-efficient task scheduling algorithm for heterogeneous cloud computing systems. Cluster Comput. 22 (2), 509–527.
Patel, N., Patel, H., 2017. Energy efficient strategy for placement of virtual machines selected from underloaded servers in compute cloud. J. King Saud Univ.-Comput. Inform. Sci..
Qin, Y., Wang, H., Yi, S., Li, X., Zhai, L., 2020. An energy-aware scheduling algorithm for budget-constrained scientific workflows based on multi-objective reinforcement learning. J. Supercomput. 76 (1), 455–480.
Quang-Hung, N., Nien, P.D., Nam, N.H., Tuong, N.H., Thoai, N., 2013. A genetic algorithm for power-aware virtual machine allocation in private cloud. In: Information and Communication Technology-EurAsia Conference. Springer, pp. 183–191.
Rodriguez, M.A., Buyya, R., 2018. Scheduling dynamic workloads in multi-tenant scientific workflow as a service platforms. Future Gener. Comput. Syst. 79, 739–750.
Sahni, J., Vidyarthi, D.P., 2015. A cost-effective deadline-constrained dynamic scheduling algorithm for scientific workflows in a cloud environment. IEEE Trans. Cloud Comput. 6 (1), 2–18.
Salami, B., Noori, H., Naghibzadeh, M., 2020. Fairness-aware energy efficient scheduling on heterogeneous multi-core processors. IEEE Trans. Comput. 70 (1), 72–82.
Singh, V., Gupta, I., Jana, P.K., 2019. An energy efficient algorithm for workflow scheduling in IAAS cloud. J. Grid Comput. 1–20.
Sofia, A.S., GaneshKumar, P., 2018. Multi-objective task scheduling to minimize energy consumption and makespan of cloud computing using NSGA-II. J. Netw. Syst. Manage. 26 (2), 463–485.
Suseela, B.B.J., Jeyakrishnan, V., 2014. A multi-objective hybrid ACO-PSO optimization algorithm for virtual machine placement in cloud computing. Int. J. Res. Eng. Technol. 3 (4), 474–476.
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. Appl. Soft Comput. 19, 264–279.
Wang, G., Li, W., Hei, X., 2015. Energy-aware real-time scheduling on heterogeneous multi-processor. In: 2015 49th Annual Conference on Information Sciences and Systems. CISS, IEEE, pp. 1–7.
Wu, Q., Ishikawa, F., Zhu, Q., Xia, Y., Wen, J., 2017. Deadline-constrained cost optimization approaches for workflow scheduling in clouds. IEEE Trans. Parallel Distrib. Syst. 28 (12), 3401–3412.
Xu, X., Dou, W., Zhang, X., Chen, J., 2015. EnReal: An energy-aware resource allocation method for scientific workflow executions in cloud environment. IEEE Trans. Cloud Comput. 4 (2), 166–179.
Yao, G., Ding, Y., Ren, L., Hao, K., Chen, L., 2016. An immune system-inspired rescheduling algorithm for workflow in cloud systems. Knowl.-Based Syst. 99, 39–50.
Zeng, L., Veeravalli, B., Zomaya, A.Y., 2015. An integrated task computation and data management scheduling strategy for workflow applications in cloud environments. J. Netw. Comput. Appl. 50, 39–48.
Zhang, Y., Cheng, X., Chen, L., Shen, H., 2018. Energy-efficient tasks scheduling heuristics with multi-constraints in virtualized clouds. J. Grid Comput. 16 (3), 459–475.
Zhou, Z., Hu, Z.-g., Yu, J.-y., Abawajy, J., Chowdhury, M., 2017. Energy-efficient virtual machine consolidation algorithm in cloud data centers. J. Central South Univ. 24 (10), 2331–2341.
Zhou, J., Wang, T., Cong, P., Lu, P., Wei, T., Chen, M., 2019. Cost and makespan-aware workflow scheduling in hybrid clouds. J. Syst. Archit. 100, 101631.
Zhu, Z., Zhang, G., Li, M., Liu, X., 2015. Evolutionary multi-objective workflow scheduling in cloud. IEEE Trans. Parallel Distrib. Syst. 27 (5), 1344–1357.

James Kok Konjaang is a Ph.D. student in the School of Computer Science, University College Dublin (UCD), Ireland. He received M.Sc. in Computer Science from Universiti Putra Malaysia (UPM), in 2016, and B.Sc. in Management with Computing from Regent University College of Science and Technology, Ghana in 2012. He is a lecturer at Bolgatanga Technical University (BTU), Ghana. His research interests include workflow scheduling, cloud computing and Grid Computing.

John Murphy is a Full Professor of Computer Science at University College Dublin (UCD), Ireland and the co-director of Performance Engineering Lab (PEL). His research interests span the telecommunications and software performance engineering areas and he is a Fellow of Engineers Ireland and the IET.

Liam Murphy is a Full Professor of Computer Science & Informatics at University College Dublin. His current research interests include wireless network performance and software performance engineering. Prof. Murphy is a Member of the IEEE (Communications, Broadcasting, and Computer societies) and a Fellow of the Irish Computer Society.