Webology (ISSN: 1735-188X)
Volume 18, Number 6, 2021

Online Makes pan And Energy Optimization Mechanism With Dynamic Task Arrival And Parallel Processing In Cloud Computing Environment

Chaya T D[1] , Dr. Mohamed Rafi[2]

[1]Research Scholar, Computer Science and Engineering, University BDT College of Engg, Davangere Karnataka-577004

[2]Professor, Dept of Studies in Computer Science & Engineering, University BDT College of Engg, Davangere Karnataka-577004

Abstract

Cloud computing has become the main source for executing scientific experiments. It is an effective technique for distributing and processing tasks on virtual machines. Scientific workflows are complex and demand efficient utilization of cloud resources. Scheduling of scientific workflows is considered as NP-complete. The problem is constrained by some parameters such as Quality of Service (QoS), dependencies between tasks and users' deadlines, etc. There exists a strong literature on scheduling scientific workflows in cloud environments. Solutions include standard schedulers, evolutionary optimization techniques, etc. In this research work we design and develop an extension of makes pan optimization where task arrival is unknown and parallel processing. Hence to achieve that we have design and developed mechanism named OMEO (Online makes pan and energy optimization) mechanism; OMEO is designed with parallel processing and dynamic arrival of task. ; In OMEO we identify the problem of makes pan and processing time and establish the relation among them. Further an algorithm is designed which can handle the unknown processing time; followed by that we design and develop a mechanism for the dynamic arrival of task i.e. where the task arrival time is unknown. Further we evaluate OMEO by considering the scientific workflow considering the metrics, TET (Task Execution Time) by comparing with the existing model. Moreover comparative analysis shows that our model achieve better results than any other algorithm.

1 Introduction

In recent years, cloud computing has become a hot research topic, and it is widely used in telecommunications, manufacturing, education and scientific research [1]. For example, storage clouds provide secure data storage, backup and recording services, which provide great convenience for users. Educational clouds [2]-[4] can virtualize various types of hardware education resources and then transmit them to the internet system, providing a convenient information platform for education departments, teachers and students. In cloud computing, resources such as hardware, software and platforms are provided as services with the ``pay-as-you-go'' model. Users need to pay for only the services or resources they need without having to purchase hardware infrastructure. The current studies focus on virtualization, resource management, cloud security, green computing, task scheduling, and so forth. As cloud computing services rapidly grow, how to effectively schedule tasks to computational resources (virtual machines) according to goals has become increasingly important. The goals of task scheduling mainly include reducing task completion time and energy consumption and improving resource utilization and load balancing ability [5][6]. With the dramatic increase in the number of cloud users, reducing task completion time is helpful for improving user experience. Improving load balancing ability contributes to fully utilizing virtual machines to prevent execution efficiency from decreasing due to the overload of resources or waste caused by excessive idle resources [7]. However, the above two objectives are mutually constrained. For instance, to reduce task completion time, it is easy to centrally schedule the tasks on the resources with strong computing power, which will cause a load imbalance problem. Therefore, it is challenging to design and optimize the task scheduling algorithm to balance the two goals of reducing completion time and improving load balancing ability.

The concept of workflow has its roots in commercial enterprises as a business process modeling tool. These business workflows aim to automate and optimize the processes of an organization, seen as an ordered sequence of activities, and are a mature research area4 lead by the workflow management coalition* (Wf MC), founded in 1993. This notion of workflow has extended to the scientific community in which scientific workflows are used support large-scale, complex scientific processes; they are designed to conduct experiments and prove scientific hypotheses by managing, analyzing, simulating, and visualizing scientific data.5 Therefore, even though both business and scientific workflows share the same basic concept, both have specific requirements and hence need separate consideration.

A workflow is defined by a set of computational tasks with dependencies between them. In scientific applications, it is common for the dependencies to represent a data flow from one task to another; the output data generated by one task becomes the input data for the next one. These applications can be CPU, memory, or I/O intensive (or a combination of these), depending on the nature of the problem they are designed to solve. In a CPU intensive workflow most tasks spend most of their time performing computations. In a memory-bound workflow most tasks require high physical memory usage. The I/O intensive workflows are composed of tasks that require and produce large amounts of data and hence spend most of their time performing I/Ooperations.6

Scientific workflows are managed by different institutions or individuals in different fields meaning they have different requirements for the software needed by tasks to run. These characteristics make them great candidates to leverage the capabilities offered by cloud computing. Scientists can configure VM images to suit the software needs of a specific workflow, and with the help of scheduling algorithms and workflow management systems, they can efficiently run their applications on a range of cloud resources to obtain results in a reasonable amount of time[8]. In this way, by providing a simple, cost-effective way of running scientific applications that are accessible to everyone, cloud computing is revolutionizing the way e-science is done. Many scientific areas have embraced workflows as mean to express complex computational problems that can be efficiently processed in distributed environments. For example, the Montage workflow7 is an astronomy application characterized by being I/O intensive that is used to create custom mosaics of the sky on the basis of a set of input images. It enables astronomers to generate a composite image of a region of the sky that is too large to be produced by astronomical cameras or that has been measured with different wavelengths and instruments. During the workflow execution, the geometry of the output image is calculated from that of the input images. Afterwards, the input data is re-projected so that they have the same spatial scale and rotation. This is followed by a standardization of the background of all images. Moreover Workflow comes with some objective to minimize that are discussed below;

- **Makes pan:** Most of the surveyed algorithms are concerned with the time it takes to run the workflow, or makes pan. As with cost, it is included as part of the scheduling objectives by either trying to minimize its value, or by defining a time limit, or deadline, for the execution of the workflow.
- **Workload maximization.** Algorithms developed to schedule ensembles generally aim to maximize the amount of work done, that is, the number of workflows executed. This objective is always paired with constraints such as budget or deadline, and hence, strategies in this category aim at executing as many workflows as possible with the given money or within the specified time frame.
- **VM utilization maximization.** Most algorithms are indirectly addressing this objective by being cost-aware. Idle time slots in leased VMs are deemed as a waste of money as they were paid for but not used, and as a result, algorithms try to avoid them in their schedules. However, it is not uncommon for this unused time slots to arise from a workflow execution, mainly because of the dependencies between tasks and performance requirements. Some algorithms are directly concerned with minimizing these idle time slots and maximizing the utilization of resources, which has benefits for users in terms of cost, and for providers in terms of energy consumption, profit, and more efficient usage of resources.
- **Energy consumption minimization.** Individuals, organizations, and governments worldwide have developed an increased concern to reduce carbon footprints to lessen the impact on the environment. Although not unique to cloud computing, this concern has also attracted attention in this field. A few algorithms that are aware of the energy consumed by

the workflow execution have been recently developed. They consider a combination of contradicting scheduling goals as they try to find a trade-off between energy consumption, performance, and cost. Furthermore, virtualization and the lack of control and knowledge of the physical infrastructure limit their capabilities and introduce further complexity into the problem [9]-[12]

Further in this research work, we focus on the makes pan and energy minimization with constraint as it is one of the important parameter.

1.1 Motivation and contribution of research work

The goals of task scheduling mainly include reducing task completion time and energy consumption and improving resource utilization and load balancing ability. With the dramatic increase in the number of cloud users, reducing task completion time is helpful for improving user experience. Improving load balancing ability contributes to fully utilizing virtual machines to prevent execution efficiency from decreasing due to the overload of resources or waste caused by excessive idle resources. However, the above two objectives are mutually constrained. Several research work towards and succeeded, however they ignored issue such as if the processing time of task is unknown, dynamic task arrival and parallel processing. For instance, to reduce task completion time, it is easy to centrally schedule the tasks on the resources with strong computing power, which will cause a load imbalance problem. Therefore, it is challenging to design and optimize the task scheduling algorithm to balance the two goals of reducing completion time and improving load balancing ability. Hence motivated by these we develop and design mechanism to optimize futher Contribution of this research work is given through the below points;

- At first we study and analyze various problem related to the makes pan minimization and further several existing mechanism and their shortcomings are analyzed.
- In this research work we design and develop OMEO-mechanism for optimizing the energy and makes pan with various constraint which is ignored by the existing model; although several existing model have considered but they have focused on single constraint.
- In here we consider the issue of parallel processing, unknown processing time and task arrival time at once and further optimize the makes pan and energy with these constraint.
- At first we design and develop algorithm for solving general issue and extending for the parallel processing and task arrival.
- OMEO-mechanism is evaluated under the scientific workflow by comparing with the existing model and comparative analysis shows that our model achieve better results than the existing model.

This research work is carried out in various section; in first section background of cloud computing and the importance of workflow is discussed. Further the various objective of optimizing is discussed, and later motivation and contribution of this research wok is highlighted. In second section we discuss various existing technique of makes pan and energy minimization and their shortcoming. In third section, we have develop and designed OMEO- mechanism with various

constraint and helps in minimizing the energy and makes pan. Further in fourth section, we evaluate the OMEO-mechanism considering scientific workflow cybershake and its variant; further comparative analysis is carried out.

2 Literature Survey

Most related scheduling problems involving workflow tasks to cloud resources with geo-distributed data are NP-hard [13]. Though task scheduling considering energy-efficiency and data transmission time between tasks in a single data center has been widely studied, there are only a few papers dealing with the above two types of data transmission time in geo-distributed data centers. In the following, we give a brief review of the literature on workflow scheduling with transmission time and energy optimization with varying electricity prices. Generally, there are two types of data transmission time. The generated data transmission time refers to the time needed to transmit generated data from predecessor tasks to the current task. The original transmission time is the access time of data from geo-distributed data centers to execute the current task. Contention is crucial for task scheduling [14]. Contention awareness was accomplished by scheduling the communication which could be regarded as a special generated data transmission. The heuristic scheduling algorithm constructed by Lin et al. [15] considers the original data transmission time without taking into account the generated data transmission time. Mei et al. [16] employed a duplication strategy to design workflow scheduling algorithms in which the generated data transmission time was considered but the original data transmission time was not. Usually the generated data transmission time is considered in task communication models [17]-[20]. However, the original data transmission time between tasks and local data has seldom been taken into account. Optimizing energy consumption is an important topic in workflow scheduling with varying electricity prices in geographically distributed data centers. It is crucial to obtain a balance between energy consumption and electricity prices for total cost minimization. Energy consumption is often related to workflow applications or task migration [18]-[20]. [21] analyzed the power consumption of operations on network devices and computing resources, such as switching, transmission, data processing and data storage during task scheduling. They claimed that that the power consumption in switching and transmission accounts for a considerable amount. [22] presented an energy and cost aware algorithm for scheduling instance intensive IoT workflows with batch processing in clouds. Mahdevan et al. [23] investigated the power consumption of network equipment such as switches during task communication. The energy efficiency of switches could be improved by shutting down switch ports and selecting switches with low data rates. In terms of the location of data centers, Qureshi et al. [24] presented an algorithm to minimize the total electricity consumption cost. Considering the variability of energy prices, S [25] developed a distributed coordination algorithm to decrease the global energy cost using a dynamic speed scaling technique. [26] investigated a branch and bound algorithm to minimize the total electricity cost. By adopting a dynamic pricing mechanism, power loads could be balanced between utility companies and data centers. Power providers could fully sell their products while users could save energy costs with dynamic electricity prices. In addition, a

heuristic algorithm with low computational complexity was proposed to achieve close to optimal performance. [27] reviewed a unified energy portfolio optimization framework for data centers. By using onsite storage and deploying geographical workload distribution, data centers can utilize high-risk energy choices through offering ancillary services or participating in wholesale electricity markets. Most of the existing literature focuses on scheduling independent tasks rather than precedence constrained workflow applications. Since workflow applications are complex, it is common to decompose them into smaller fragments and to partition their deadlines into sub deadlines. Classic methods for this are the heterogeneous Earliset-Finish Time (HEFT) algorithm and the Partial Critical Path method (PCP) [17], [18] presented the minimum dependencies energy-efficient scheduling algorithm which was shown to outperform HEFT and PCP. HEROS is an energy efficient task assignment algorithm which allocates independent tasks to heterogeneous servers. Doyle et al. [28] designed the Stratus algorithm for independent task scheduling to allocate tasks to the nearest data center in order to optimize energy consumption. [29] proposed two approaches for multi-objective workflow scheduling in geo-distributed data centers. They minimized the total execution time and the total cost but only with the original data transmission time in mind. [30] proposed a bi-objective genetic algorithm (BOGA) to optimize both energy consumption and system reliability of workflows in the heterogeneous computing systems. [31] developed a prediction based dynamic multi-objective evolutionary algorithm for dynamic workflow scheduling problems where six objectives were investigated: makes pan, cost, energy, degree of imbalance, reliability and utilization. Garg et al. [29] presented a reliability and energy efficient workflow scheduling algorithm for jointly considering both resource and user constraints. Based on DVFS, [32] proposed a downward and upward energy consumption minimization method for the energy consumption in a single data center. Minimizing energy consumption only by using DVFS was demonstrated to be insufficient because the server frequency below a given threshold might lead to higher energy consumption [33]-[36].

3 Proposed Methodology

In this section, we design and develop OMEO (Online makes pan and energy optimization) mechanism to minimize the makes pan and energy for the workflow model; further considering the legacy of our previous research work where the processing time is negligible at first, two cases are design for optimization and we consider the parallel processing and dynamic task arrival together which was ignored by the other researcher. Moreover this is online algorithm as the input are scheduled one by one.

3.1 System model

Let's consider any hybrid cloud model where the device access to m identical parallel process denoted by $\mathbb{g} \in \mathbb{J} = \{1, \ldots\ldots, \mathbb{k}\}$. Moreover initially we consider that remote cloud as the single powerful processor referred as processor 0. Later we extend our work to multiple processor. Initially we assume that all task are available at time 0.

3.2 Initialization

Let's consider non-pre-emptible and independent tasks $\mathbb{l}$ that are available to given scheduler at time is zero; let's consider $\mathbb{R} = \{1, \dots, \mathbb{l}\}$ be task indices with processing time for task is unknown and denoted as $\mathbb{s}_{\mathbb{h}}$. Moreover main intention here is to optimize the makespan of scheduled task on the given processor; in here we consider the offloading cost and makespan together through weighted sum. Let $\mathbb{Q}$ be the set of possible schedule and $\mathbb{s}$ be the schedule, further $\mathbb{s}$ decides offloading of task on processor; let $\mathbb{R}_{\mathbb{g}}(\mathbb{r})$ be the task scheduling set on processor $\mathbb{g} \in \mathbb{K} \cup \{0\}$ under given schedule $\mathbb{q}$. consider $\mathbb{A}_{\mathbb{g}}(\mathbb{q})$ as the total time taken to complete the assigned to processor and it is formulated through the below equation.

$$\mathbb{A}_{\mathbb{g}}(\mathbb{q}) = \sum_{\mathbb{h} \in \mathbb{R}_{\mathbb{g}}(\mathbb{r})} \mathbb{s}_{\mathbb{h}}, \forall\, \mathbb{g} \in \mathbb{K} \quad (1)$$

$$\mathbb{A}_0(\mathbb{A}) = \sum_{\mathbb{h} \in \mathbb{R}_0(\mathbb{q})} \varepsilon \mathbb{s}_{\mathbb{h}} \quad (2)$$

Further as $\mathbb{s}_{\mathbb{h}}$ is unknown, cost is given as

$$\vartheta(\mathbb{q}) = \sum_{\mathbb{h} \in \mathbb{R}_0(\mathbb{q})} \widehat{\mathbb{x}_{\mathbb{h}}} \quad (3)$$

Moreover total cost of schedule $\mathbb{q}$ is given through below equation

$$\aleph(\mathbb{q}) \triangleq \max_{\mathbb{h} \in \mathcal{L} \cup \{0\}} \{\mathbb{A}_{\mathbb{g}}(\mathbb{q})\} + \mathbb{u}\vartheta(\mathbb{q}) \quad (4)$$

In the above equation $\mathbb{u}$ indicates weight parameter which allows to tune importance between cost and makespan, further cost minimization can be given as:

$$\min_{\mathbb{q} \in \mathbb{Q}} \aleph(\mathbb{q}) \quad (5)$$

3.3 Intermediate framework

In here we design an intermediate framework where processing time $\mathbb{s}_{\mathbb{h}}$ is unknown and costs are $\widehat{\mathbb{x}_{\mathbb{h}}}$, $\forall\ \mathbb{h}$; in order to develop this intermediate framework we consider $\mathbb{N}$ as a problem instance of $\mathbb{N}_{\text{sum}}$.Further $\mathbb{q}(\mathbb{N})$ is schedule of online algorithm and $\overline{\mathbb{q}}^*(\mathbb{N})$ is schedule of an optimal algorithm. Further the interactive framework is given as:

$$\max_{\forall \mathbb{N}} \frac{\aleph(\mathbb{q}(\mathbb{N}))}{\aleph(\overline{\mathbb{q}}^*(\mathbb{N}))} \leq \theta \quad (6)$$

In here θ is tight for algorithm such that it satisfies the below equation.

$$\aleph(\mathbb{q}(\mathbb{N})) = \theta\aleph(\overline{\mathbb{q}}^*(\mathbb{N})) \quad (7)$$

3.4 Problem definition

In this section we define the problem which is to reduce the makes pan on given $m + 1$ processor p_{max} as the cost of off-loading

$\underset{\mathbb{q} \in \mathbb{Q}}{\text{minimize}} \; \mathbb{A}_{\max}(\mathbb{q})$	(8)

In the above equation $\mathbb{A}_{\max}(\mathbb{q})$ is formulated as

$\mathbb{A}_{\max}(\mathbb{q}) \triangleq \max\{\max_{\mathbb{g} \in \mathbb{J} \cup \{0\}} \mathbb{A}_{\mathbb{g}}(\mathbb{q}), \mathbb{u}\mathbb{R}(\mathbb{q})\}$	(9)

Moreover $\mathbb{m}_{\max}$ and $\mathbb{A}^*_{\max}$ indicates objective and further optimal schedule is denoted throughs*; further if $\mathbb{u}$=0 then $\mathbb{m}_{\max}$ is minimal on $\mathbb{k} + 1$ processor.

Further we also observe that there has been issue of parallel processing, hence at first we consider in the single processor later section we extend it for parallel execution; similarly task arrival is also considered since in real time the task can arrive anytime.

3.4.1 Relation between $\mathbb{m}_{sum}$ and $\mathbb{m}_{max}$

Let $\mathbb{q}'$ be considered as computed schedule for solving the $\mathbb{m}_{\max}$, further inequalities are formulated as:

$\gamma(\mathbb{q}') = 2\theta\Upsilon(\bar{\mathbb{q}}*)$	(10)

Moreover in the above equation we observe that $\mathbb{m}_{\max}$and $\mathbb{m}_{sum}$requires similar solution, hence we develop mechanism for $\mathbb{m}_{\max}$. Moreover this is achieved through establishing the lower bound for $C^*_{\max}$

Let $\mathbb{A}_{\mathbb{g}}{}^*$ denotes completion time and $\mathbb{R}^*_{\mathbb{g}}$ indicates task scheduled on given processor i under optimal schedule $\mathbb{R}^*_{\mathbb{g}}$ .optimal equation is formulated as:

$\mathbb{A}^*_{\mathbb{g}} = \sum_{\mathbb{h} \in \mathbb{R}^*_{\mathbb{g}}} \mathbb{s}_{\mathbb{h}}, \forall \mathbb{g} \in \mathbb{J},$	(11)

$\mathbb{A}^*_0 = \sum_{\mathbb{h} \in \mathbb{R}^*_0} \varepsilon \mathbb{s}_{\mathbb{h}}$	(12)

$\mathbb{A}^*_{\max} \geq \sum_{\mathbb{h} \in \mathbb{R}^*_{\mathbb{g}}} \mathbb{s}_{\mathbb{h}}, \forall \mathbb{g} \in \mathbb{J},$	(12)

$$\mathbb{A}^{*}_{\max} \geq \sum_{\mathbb{h}\in\mathbb{R}^{*}_{\mathbb{g}}} \varepsilon \mathbb{s}_{\mathbb{h}} \quad (13)$$

Further substituting $\mathbb{A}^{*}_{\max}$ in equation (12) and (13), we achieve:

$$\left(\mathbb{I} + \frac{1}{\varepsilon}\right)\mathbb{A}^{*}_{\max} \geq \sum_{\mathbb{h}=1} \mathbb{s}_{\mathbb{h}} \quad (14)$$

Further, $\mathbb{A}^{*}_{\max}$ be the optimal objective, $\mathbb{q}'$ be the optimal schedule for scheduling task T with earlier assumption regarding the processing time. $\mathbb{A}^{*}_{\mathbb{g}}$ be the schedule length, $\mathbb{R}'_0 \subseteq \mathbb{R}'$ be the task offloaded then we have further equation

$$\frac{1}{\mu}\sum_{\mathbb{g}=1}^{\mathbb{k}} \mathbb{A}^{*}_{\mathbb{g}} + \sum_{\mathbb{h}\in\mathbb{R}'_0} \mathbb{x}_{\mathbb{h}} = \sum_{\mathbb{h}\in\mathbb{R}'} \mathbb{x}_{\mathbb{h}} \quad (15)$$

Further we use$\mathbb{A}^{*}_{\max} \geq \mathbb{A}^{*}_{\mathbb{g}}, \ \forall\, \mathbb{g} \in \mathbb{J}\mathbb{U}\{0\}$ and $\mathbb{A}^{*}_{\max} \geq \sum_{\mathbb{h}\in\mathbb{R}'_0} \mathbb{x}_{\mathbb{h}}$ and obtain

$$\left(1 + \frac{\mathbb{k}}{\mu}\right)\mathbb{A}^{*}_{\max} \geq \sum_{\mathbb{h}\in\mathbb{R}'} \mathbb{x}_{\mathbb{h}} \quad (16)$$

3.5 Case study for various values of processing time

In this case we study the various case of processing time; in previous research we considered ε value as zero whereas in here If ε is greater than zero then the above condition is equivalent to the below equation

$$\mathbb{A}^{(1)}_{\max}\left(\mathbb{q}^{\mathrm{alg}}\right) \leq \min\,(2\mu, \delta)\mathrm{C}^{*}_{\max} \quad (17)$$

Where

$$\delta = \left(1 + \frac{1}{\mathbb{k}\varepsilon} + \frac{\mathbb{k}-1}{\mathbb{k}}\max\left(1, \frac{1}{\varepsilon}\right)\right)$$

Through the equation 8, we get

$$\mathbb{A}^{*}_{\max} \geq \min(1, \varepsilon)\mathrm{u}_{\mathrm{j}} \geq \min\left((1, \varepsilon)\,\tau_{\mathrm{j}}, \forall \mathrm{j}\right)$$

And this results in $\mathbb{A}^{(1)}_{\max}\left(\mathbb{q}^{\mathrm{alg}}\right) \leq \mathbb{A}^{(1)}_{\max}(\mathbb{q}_1)$

3.6 OMEO algorithm

Further to improvise the algorithm discussed in previous research work where the processing time was negligible, here we consider the processing time greater than zero and also task arrival time is unknown. Although the proposed algorithm is an extension of previous algorithm. It has similar iteration as previous algorithm; further list is formed through forming the task in ascending order $\mathbb{x}_{\mathbb{h}}$and task from the start of list are given to the remote cloud and task from the end is given to the local processor. Further if the processing time reaches $\mu\mathbb{x}_{\mathbb{h}}$ then it is terminated and discarded. Furthermore in second iteration, all the terminated task are rescheduled in accordance with the first

iteration and it terminates the task on the processor exceeds $\mathbb{x}_{\mathbb{h}}$. Moreover the optimized algorithm is given in the below table1, this algorithm has three iteration, in each iteration proposed algorithm performs the task sorting which takes particular amount . $\mathbb{q}^{\mathrm{alg1}}$ indicates the schedule.

Step1:	$\ddot{\mathbb{R}}^{(\mathbb{j})} = \mathbb{R}$ and $\mathbb{j} = 1$
Step2:	While $\mathbb{j}$ <=3
Step3:	$\mathbb{e}_0 = \ddot{\mathbb{R}}^{(\mathbb{j})} + 1$ and $\mathbb{e}_1 = 1$
Step4:	Sorting $\ddot{\mathbb{R}}^{(\mathbb{j})}$ in the ascending order given through $\mathbb{x}_{\mathbb{h}}$
Step5:	Processing of task $\mathbb{e}_1$ on given processor

Step6:	If $\mathbb{j} = 2$ then
	Terminate task $\mathbb{e}_1$if the time execution exceeds $\mathbb{x}_{\mathbb{h}_0}$. and add in $\dddot{\mathbb{R}}^{(3)}$
	End if

Step7:	For $\mathbb{i}$= 1 to $\min\{\mathbb{k}, \lvert\ddot{\mathbb{R}}^{(\mathbb{j})}\}$ do $\mathbb{e}_0 = \mathbb{e}_0 - 1$
Step8:	Processing of task $\mathbb{e}_0$ on the given processor $\mathbb{i}$

Step9:	If $\mathbb{j} = 1$ then
	Terminate task $\mathbb{e}_0$ if execution time reaches $\mathbb{x}_{\mathbb{h}_0}/\varepsilon$ and add in $\dddot{\mathbb{R}}^{(3)}$
	End if
	End for

Step10:	While $\ddot{\mathbb{R}}^{(\mathbb{j})}$ is not equal to null do Wait till occurrence of next event $\mathbb{C}$. Further If $\mathbb{C}$ is equal to task that is completed or terminated on given processor $\breve{\mathbb{g}} \epsilon \mathbb{J}$ then Terminate the task $\breve{\mathbb{h}}$ scheduled on given processor
Step11:	$\ddot{\mathbb{R}}^{(\mathbb{j})}$is equal to $\ddot{\mathbb{R}}^{(\mathbb{j})} \backslash \{\ddot{\mathbb{h}}\}$ $\mathbb{e}_0 = \mathbb{e}_0 - 1$
Step12:	If $\mathbb{e}_0$ is greater than $\mathbb{e}_1$ then schedule on the given processor ï

Step13;	If $\mathbb{j} = 1$ then
	Terminate the task $\mathbb{e}_0$ if the execution time reaches time $\mu\mathbb{x}_{\mathbb{h}_0}$ and add in $\ddot{\mathbb{R}}^{(2)}$
	End if

Step13:	If $\mathbb{j} = 2$ then Terminate task $\mathbb{h}_0$ if the execution time reaches the $\mathbb{x}_{\mathbb{h}_0}/\epsilon$ and add in $\ddot{\mathbb{R}}^{(2)}$

	End if Else if $\mathbb{C}$ is equal to the task $\mathbb{h}_1$ finished on the given processor Then Terminate task $\mathbb{h}_1$ if it is schedule on the processor
Step14:	$\ddot{\mathbb{R}}^{(\mathbb{j})} = \ddot{\mathbb{R}}^{(\mathbb{j})} \backslash \{\dot{\mathbb{h}}\}$ $\mathbb{h}_1 = \mathbb{h}_1 + 1$
Step15:	If task $\mathbb{h}_1$ is not finished or terminated then schedule it on the given processor
Step16:	If $\mathrm{l} = 2$ then Terminate task $\mathbb{h}_1$ if the TAT(Task Execution Time) reaches the threshold of $\mathbb{x}_{\mathbb{h}_0}$ and add it in $\ddot{\ddot{\mathbb{R}}}^{(3)}$ End if statement End if statement End while loop $\mathbb{j} = \mathbb{j} + 1$ End while loop

In next section we extend this optimization of makes pan minimization by considering the constraint.

3.6.1 Dynamic task arrival and parallel processing

In this section, we extend developed work to be considered for the random task arrival and parallel processing. In the previous work, we considered that all that tasks are available and starts from null; however in practicality the task may arrive at any time and also their arrival time is not known. Hence considering the problem we denote it through $\mathbb{m}_{\mathrm{Sum}}^{\mathrm{DTA}}$ and $\mathbb{m}_{\mathrm{max}}^{\mathrm{DTA}}$. Further extension algorithm can be written as below:

Table 1

Step1:	Select any job $\mathbb{h}$ and run for execution on given machine $\mathbb{g}(\mathbb{h})$ Time taken is denoted as φ
Step2:	Let $\mathbb{o} = \varphi/\varepsilon\mathbb{l}$
Step3:	Design an algorithm for entire job which is not completed and schedule it with setting $\mathbb{o}_{\mathbb{h}} \leftarrow \mathbb{o}$
Step4:	If the jobs are not completed then set $\leftarrow 2\mathbb{o}$ and repeat step3

Moreover in the above algorithm, let $\mathbb{A}^{*}_{\max}$ be the optimal schedule length; here in first step the complete time φ taken by the job $\mathbb{h}$ on machine $\mathbb{g}(\mathbb{h})$ is comparatively less than the schedule length. Further in step 3, first iteration produces a schedule and construct the schedule through assigning the $\mathbb{o}$ jobs on particular machine which acts fast. Moreover in the worst case scenario entire jobs are given to one machine and this would have the length of $\mathbb{l}\mathbb{o} = \varphi/\varepsilon$

4 Performance Evaluation

4.1 System configuration and dataset details

Now a days, the request of CC (cloud computing) resources has highly emerged in real-time due to its vibrant uses, flexibility, cost effective and easily accessible at anywhere anytime through internet. Multimedia-signal-processing method is well-known technique that can be utilize in these CC-devices. Therefore, the performance of these computing devices must be superior due to the extensive demand of these computing devices in day-to-day life. However, high energy consumption in these computing devices can disturb their performance; further makes pan is an important constraints, hence to optimize these objectives, we have introduced a OMEO for heterogeneous computing devices which efficiently reduces energy consumption as well as provide superior performance. The run-time can be evaluated considering various jobs as 30, 50, 100, and 1000. Graphical representation of our outcomes is also presented considering execution-time, number of tasks and energy consumption. The run-time and total power consumed can be evaluated using different parameters in table 1 which is demonstrated in the following section. Our proposed model is tested on CyberShake scientific dataset [37] ; further details about the dataset is depicted in the table1. We have considered different sizes of scientific workflow experiments as 30, 50, 100 and 1000. Moreover proposed OMEO is evaluated considering 64-bit with operating system of windows 10 with 16 GB RAM and loaded with I5 processor; further the model 3.20 GHz CPU and the model is evaluated using the programming language using java and neon .3 editor and cloud sim simulator[38].

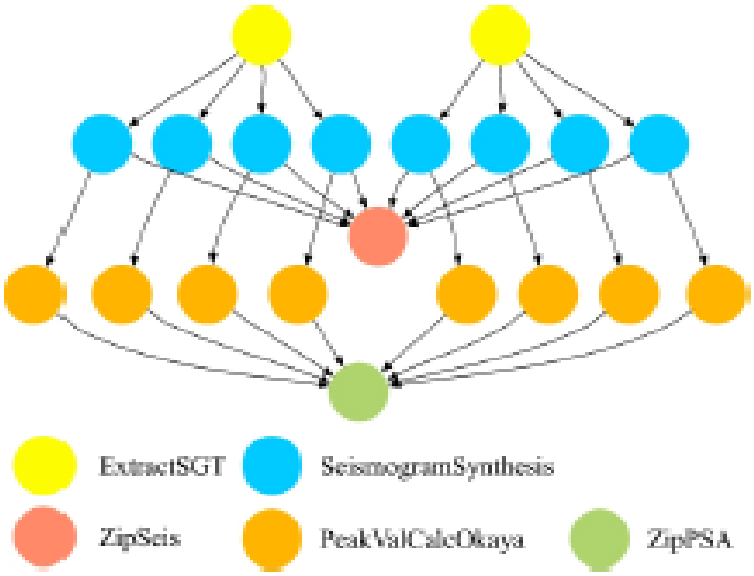

Figure 1cybershake_workflow

Table 2 dataset details

Dataset	Number of Nodes	W _levels	Total number of task	Tasks(Parallel)
cybershake	100	5	2083325	48

4.2 Comparison

Moreover is evaluated through varying the number of virtual machine as 25, 45 and 65, these varied results are compared with the existing model by considering the for eminent parameter i.e. total execution time, power sum, power average, average power and energy consumption. In here table 2 and table 3 presents the comparison of existing model [39]with the proposed - model by varying the virtual machine as 25, 45 and 65.

Further evaluation is carried out by considering four distinctive instance; first instance is created with virtual machine of 25 and cybershake_30 dax file, in second instance number of VM is increased by 45 on cybershake_50. Similarly increasing the number of VM as 65on cybershake_100, we create third instance. At last fourth instance is created by considering 25 VM on cybershake_1000.

4.2.1 Energy Consumption

Energy is one of the essential parameter in workflow scheduling; less the energy consumption better is the model. Moreover for instance discussed earlier has been consider for energy consumption; in case of first instance existing model consumes 3495.427 whereas consumes 416.85. for second instance, existing model consumes 8703.908 whereas consumes 1330.922;

similarly for third instance and fourth instance existing model consumes 19206.18 and consumes 1436.837. Table 3 shows the data comparison and figure 1 shows the graphical comparison.

Table 3 Enegy Consumption Comparison

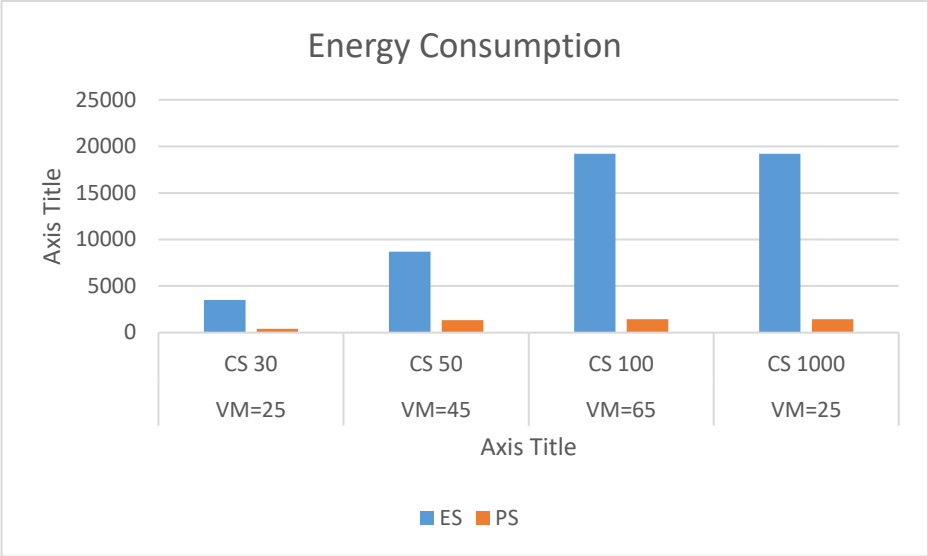

Figure 2 Energy Consumption comparison

Instance	First instance	Second instance	Third instance	Fourth instance
Number of VM and workflow variant	VM=25 CS 30	VM=45 CS 50	VM=65 CS 100	VM=25 CS 1000
ES	3495.427	8703.908	19206.18	19206.18
PS-OMEO	416.8566	1330.922	1436.837	1436.837
	88.07%	84.71%	92%	92%

4.2.2 Makes pan

Makes pan aka TET (Total Execution Time) is the time taken for execution of the task; moreover the makes pan needs to be minimized. In this section we compare TET which is given in table 4; in here for first instance, total execution time taken by the existing model is 6359.41 and takes 3030.25. Further in case of second instance and third instance existing model requires 14734.26 and 18609.25 sec whereas requires 2953.86 sec each. At last in case of fourth instance existing model requires 18609.25 and takes 2953.86.

Instance	First Instance	Second Instance	Third Instance	Fourth Instacne

Number Of VM and Workflow_variant	VM=25 CS 30	VM=45 CS 50	VM=65 CS 100	VM=25 CS 1000
ES	6359.41	14734.26	18609.25	18609.25
PS-OMEO	3030.25	2953.86	2953.86	2953.86
Improvement(in percentage)	52.35%	79.95	84.13	84.13

Table 4 Makespan Comparison

Figure 3 Makespan optimization

4.2.3 Average Power

Power is another parameter considered for comparison to prove the model efficiency, in this sub-section we compare the average power required for task. For first and second instance existing model requires 20.39242 and requires 15.9011. For third and fourth instance existing model requires 20.39242 and proposed model requires average power of 15.90111. Data comparison has been given in table 5 and the comparison graph is plotted in the figure 3.

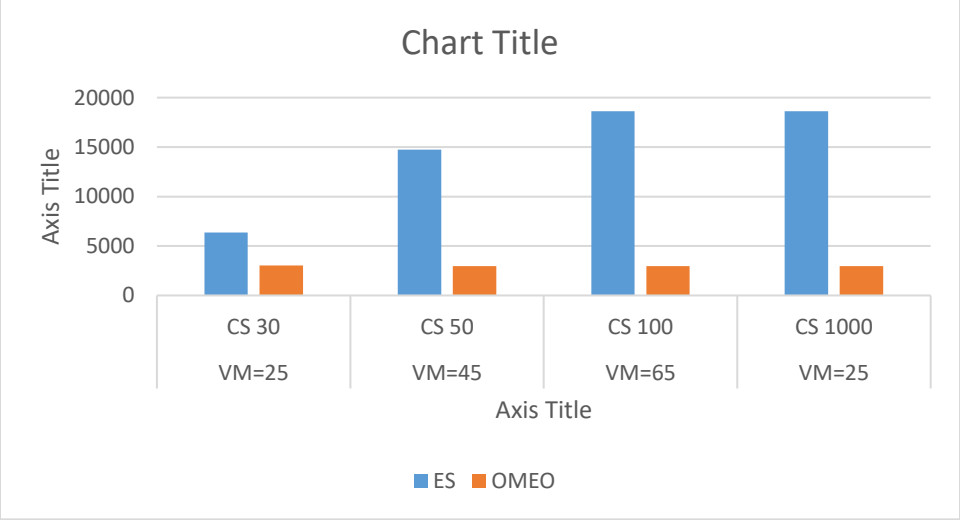

Table 5 Average Power

	First Instance	Second instance	Third Instance	Fourth Instance
	VM=25 CS 30	Vm=45 CS 50	VM=65 CS 100	Vm=25 CS 1000
ES	20.39242	20.39242	20.39242	20.39242

PS	15.9011	15.9011	15.9011	15.90111
Improvement	27.92	27.92	27.92	27.92

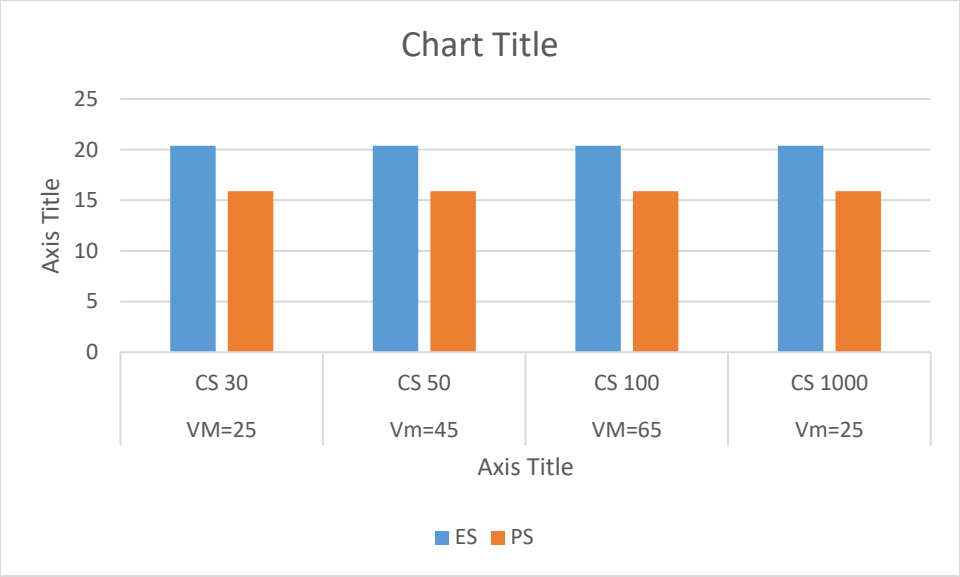

Figure 4Average Power comparison

4.2.4 Power Sum

At last we compare the Power sum Comparison with the existing model; here it is observed hat for first anzd second instance existing model requires 15723029 and 30769230 whereas proposed model requires 4955134 and 4696963. Similarly for third and fourth instance power sum required is 62040773 and 171478073.4

Table 6 Power Sum comparison

Instance	Instance 1	Instance2	Instance3	Instance4
Number of VM and workflow variant	VM=25 CS 30	VM=45 CS 50	Vm=65 CS 100	Vm=25 CS 1000
ES	15723029	30769230	62040773	171478073.4
PS-OMEO	4955134	4696963	4955134	7623610.319
Improvement	68.48	84.73	92.0177	55.54

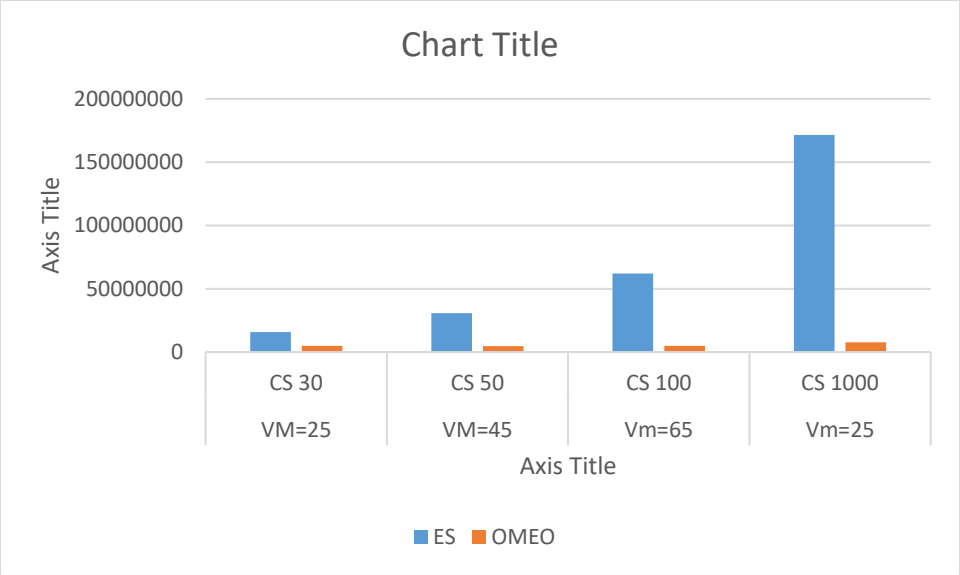

Figure 5 Power Sum Comparison

4.3 Comparative analysis

In this section, we analyze the comparison of existing methodology considering the various metric such as energy consumption, Average power, power sum and makes pan; at first we compare the energy consumption. Moreover through the comparative analysis it is observe that requires 88.07%, 84.17%, 92% and 92% less energy than the existing model on four distinctive instance respectively; further through the comparison of makes pan, we observe that requires 52.35%, 79.95%, 84.13% and 84.13% less execution time than the existing model on for instance respectively. Moreover average power required by is 27.92% less than the existing model in all four instance, further in case of power sum, requires 68.48%, 84.73%, 92.01% and 55.54% less than the existing model.

Conclusion

In this research work, we have designed and developed a mechanism named , is online algorithm for optimization of energy and makes pan; Moreover proposed approach considers the various considers which is ignored by the other researcher. These constraints includes dynamic arrival of task, unknown processing time and parallel processing; considers these as issue and optimizes through proposing two distinctive algorithm; one is for optimizing the parameter and other for considering the constraint. To evaluate the model several parameter such as energy consumption, average power, power sum and makes pan is considered on the scientific workflow; further for critical evaluation we considered four instance which varies the number of virtual machine and DAX file of cyber shake. Moreover through the comparative analysis, it is observed that simply outperforms the existing model with performing up to 90% better in terms of energy consumption. Further OMOE minimizes 75% of makes pan compared to the existing model and requires 27.92%

less average power than the existing model; at last in case of power sum it is 75% more efficient. Although performs marginally better than existing model However, in practice it would be too expensive and time-consuming to collect such data at run-time and further cost is one of the major issue. Hence in the future we would be focusing on increasing the reliability of the model through optimizing the cost.

Reference

1. Y. Xia, M. Zhou, X. Luo, S. Pang, and Q. Zhu, ``Stochastic modeling and performance analysis of migration-enabled and error-prone clouds," IEEE Trans. Ind. Informat., vol. 11, no. 2, pp. 495_504, Apr. 2015.
2. Y. Xia, M. Zhou, X. Luo, S. Pang, and Q. Zhu, ``A stochastic approach to analysis of energy-aware DVS-enabled cloud datacenters," IEEE Trans. Syst., Man, Cybern., Syst., vol. 45, no. 1, pp. 73_83, Jan. 2015.
3. Y. Yin, Y. Xu, W. Xu, M. Gao, L. Yu, and Y. Pei, ``Collaborative service selection via ensemble learning in mixed mobile network environments," Entropy, vol. 19, no. 7, p. 358, 2017.
4. J. Yu, Z. Kuang, B. Zhang, W. Zhang, D. Lin, and J. Fan, ``Leveraging content sensitiveness and user trustworthiness to recommend _ne-grained privacy settings for social image sharing," IEEE Trans. Inf. Forensics Security, vol. 13, no. 5, pp. 1317_1332, May 2018.
5. Y. Yin, F. Yu, Y. Xu, L. Yu, and J. Mu, ``Network location-aware service recommendation with random walk in cyber-physical systems," Sensors, vol. 17, no. 9, p. 2059, 2017.
6. J. Yu, B. Zhang, Z. Kuang, D. Lin, and J. Fan, ``iprivacy: Image privacy protection by identifying sensitive objects via deep multi-task learning," IEEE Trans. Inf. Forensics Security, vol. 12, no. 5, pp. 1005_1016, 2017.
7. A. Choudhary, I. Gupta, V. Singh, and P. K. Jana, ``A GSA based hybrid algorithm for bi-objective work_ow scheduling in cloud computing," Future Gener. Comput. Syst., vol. 83, pp. 14_26, 2018.
8. Q. Peng, M. Zhou, Q. He, Y. Xia, C. Wu, and S. Deng, ``Multi-objective optimization for location prediction of mobile devices in sensor-based applications," IEEE Access, vol. 6, pp. 77123_77132, 2018.
9. W. Li, Y. Xia, M. Zhou, X. Sun, and Q. Zhu, ``Fluctuation-aware and predictive work_ow scheduling in cost-effective infrastructure-as-a-service clouds," IEEE Access, vol. 6, pp. 61488_61502, 2018.
10. R. Xu et al., ``Asuf_cient and necessary temporal violation handling point selection strategy in cloud work_ow," Future Gener. Comput. Syst., vol. 86, pp. 464_479, 2018.
11. S. Deng, L. Huang, J. Taheri, and A. Y. Zomaya, ``Computation of_oading for service work_ow in mobile cloud computing," IEEE Trans. Parallel Distrib. Syst., vol. 26, pp. 3317_3329, Dec. 2015.

12. J. Yu, D. Tao, M.Wang, and Y. Rui, ``Learning to rank using user clicks and visual features for image retrieval," IEEE Trans. Cybern., vol. 45, no. 4, pp. 767_779, 2015.
13. B. Qureshi, "Profile-based power-aware workflow scheduling framework for energy-efficient data centers," Future Generation Computer Systems, vol. 94, pp. 453–467, 2019.
14. O. Sinnen and L. A. Sousa, "Communication contention in task scheduling," IEEE Transactions on Parallel and Distributed Systems, vol. 16, no. 6, pp. 503–515, 2005.
15. W. Lin, C. Liang, J. Z. Wang, and R. Buyya, "Bandwidth-aware divisible task scheduling for cloud computing," Software: Practice and Experience, vol. 44, no. 2, pp. 163–174, 2014.
16. J. Mei, K. Li, and K. Li, "Energy-aware task scheduling in heterogeneous computing environments," Cluster Computing, vol. 17, no. 2, pp. 537–550, 2014.
17. S. Abrishami, M. Naghibzadeh, and D. H. Epema, "Cost-driven scheduling of grid workflows using partial critical paths," IEEE Transactions on Parallel and Distributed Systems, vol. 23, no. 8, pp. 1400–1414, 2012.
18. M. Z˙ otkiewicz, M. Guzek, D. Kliazovich, and P. Bouvry, "Minimum dependencies energy-efficient scheduling in data centers," IEEE Transactions on Parallel and Distributed Systems, vol. 27, no. 12, pp. 3561–3574, 2016.
19. G. Xie, J. Jiang, Y. Liu, R. Li, and K. Li, "Minimizing energy consumption of real-time parallel applications using downward and upward approaches on heterogeneous systems," IEEE Transactions on Industrial Informatics, vol. 13, no. 3, pp. 1068–1078,2017.
20. X. Qu, P. Xiao, and L. Huang, "Improving the energy efficiency and performance of data-intensive workflows in virtualized clouds," Journal of Supercomputing, vol. 74, no. 7, pp. 2935–2955, 2018.
21. X. Xu, W. Dou, X. Zhang, and J. Chen, "Enreal: An energy-aware resource allocation method for scientific workflow executions in cloud environment," IEEE Transactions on Cloud Computing, vol. 4, no. 2, pp. 166–179, 2016.
22. K. Ye, Z. Wu, C. Wang, B. B. Zhou, W. Si, X. Jiang, and A. Y. Zomaya, "Profiling-based workload consolidation and migration in virtualized data centers," IEEE Transactions on Parallel and Distributed Systems, vol. 26, no. 3, pp. 878–890, 2015.
23. J. Baliga, R. W. Ayre, K. Hinton, and R. S. Tucker, "Green cloud computing: Balancing energy in processing, storage, and transport," Proceedings of the IEEE, vol. 99, no. 1, pp. 149–167, 2011.
24. Y. Wen, Z. Wang, Y. Zhang, J. Liu, B. Cao, J. Chen, "Energy and cost aware scheduling with batch processing for instance intensive IoT workflows in clouds," Future Generation Computer Systems, vol. 101, no. 1, pp. 39–50, 2019.
25. P. Mahadevan, S. Banerjee, and P. Sharma, "Energy proportionality of an enterprise network," in Proceedings of the first ACM SIGCOMM workshop on Green networking. ACM, 2010, pp. 53–60.

26. A. Qureshi, R. Weber, H. Balakrishnan, J. Guttag, and B. Maggs, “Cutting the electric bill for internet-scale systems,” in ACM SIGCOMM computer communication review, vol. 39, no. 4. ACM, 2009, pp. 123–134.
27. R. Stanojevic and R. Shorten, “Distributed dynamic speed scaling,” in INFOCOM, 2010 Proceedings IEEE. IEEE, 2010, pp.1–5.
28. J. Doyle, R. Shorten, and D. O’mahony, “Stratus: Load balancing the cloud for carbon emissions control,” IEEE Transactions on Cloud Computing, vol. 1, no. 1, pp. 116–128, 2013.
29. J. Liu, E. Pacitti, P. Valduriez, D. De Oliveira, and M. Mattoso, “Multi-objective scheduling of scientific workflows in multisite clouds,” Future Generation Computer Systems, vol. 63, pp. 76–95,2016.
30. L. Zhang, K. Li, C. Li, K. Li, “Bi-objective workflow scheduling of the energy consumption and reliability in heterogeneous computing systems,” Information Sciences, vol. 379, pp. 241–256,2017.
31. G. Ismayilov, H.R. Topcuoglu, “Neural network based multi-objective evolutionary algorithm for dynamic workflow scheduling in cloud computing,” Future Generation Computer Systems, vol. 102, pp. 307–322, 2020.
32. G. Xie, J. Jiang, Y. Liu, R. Li, and K. Li, “Minimizing energy consumption of real-time parallel applications using downward and upward approaches on heterogeneous systems,” IEEE Transactions on Industrial Informatics, vol. 13, no. 3, pp. 1068–1078, 2017.
33. M. Safari and R. Khorsand, “PL-DVFS: combining power-aware list-based scheduling algorithm with DVFS technique for realtime tasks in cloud computing,” Journal of Supercomputing, vol. 74, no. 10, pp. 5578–5600, 2018.
34. J. Jiang, Y. Lin, G. Xie, L. Fu, and J. Yang, “Time and energy optimization algorithms for the static scheduling of multiple workflows in heterogeneous computing system,” Journal of Grid Computing, vol. 15, no. 4, pp. 435–456, 2017.
35. B. Zhao, H. Aydin, and D. Zhu, “Shared recovery for energy efficiency and reliability enhancements in real-time applications with precedence constraints,” ACM Transactions on Design Automation of Electronic Systems, vol. 18, no. 2, p. 23, 2013.
36. Z. Tang, L. Qi, Z. Cheng, K. Li, S. Khan, and K. Li, “An energy-efficient task scheduling algorithm in dvfs-enabled cloud environment,” Journal of Grid Computing, vol. 14, no. 1, pp. 55–74, 2016.
37. https://pegasus.isi.edu/workflow_gallery/gallery/cybershake/index.php
38. W. Long, L. Yuqing and X. Qingxin, "Using Cloud Sim to Model and Simulate Cloud Computing Environment," 2013 Ninth International Conference on Computational Intelligence and Security, Leshan, 2013, pp. 323-328.
39. S. Pang, W. Li, H. He, Z. Shan and X. Wang, "An EDA-GA Hybrid Algorithm for Multi-Objective Task Scheduling in Cloud Computing," in IEEE Access, vol. 7, pp. 146379-146389, 2019, doi: 10.1109/ACCESS.2019.2946216.