Sustainable Energy Technologies and Assessments 45 (2021) 101210

Contents lists available at ScienceDirect

Sustainable Energy Technologies and Assessments

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

Cost-based Energy Efficient Scheduling Technique for Dynamic Voltage and Frequency Scaling System in cloud computing

Muhammad Sohaib Ajmal[a], Zeshan Iqbal[a], Farrukh Zeeshan Khan[a], Muhammad Bilal[b,*], Raja Majid Mehmood[c]

[a] Computer Science Department, University of Engineering and Technology Taxila, Punjab, Pakistan
[b] Department of Computer Engineering, Hankuk University of Foreign Studies, Gyeonggi-do 17035, South Korea
[c] Information and Communication Technology Department, Xiamen University Malaysia, Sepang 43900, Malaysia

ARTICLE INFO

Keywords:
Cloud computing
Energy efficiency
Resource allocation
Scheduling algorithm
SLA violations
Virtual server scaling

ABSTRACT

Cloud computing is used as a backbone infrastructure to meet exponentially increasing computational and storage demands. This increase in service demands in smart cities will result in escalating energy consumption in cloud datacenters. Such a rise in energy consumption will result in upsurge of operational costs and emission of greenhouse gases. In this work, a green cloud computing algorithm named "Cost-based Energy Efficient Scheduling Technique for Dynamic Volage Frequency Scaling (DVFS) Systems (CEEST)" is proposed. The proposed algorithm reduces energy consumption without compromising the quality of service (QoS). The goal of this algorithm is optimization and management of servers in the datacenters by utilizing maximum resources of the servers and powering off the underutilized servers. CEEST utilizes the scaling of virtual machines to finish jobs in the deadlines to reduce violations of service level agreement (SLA). Simulation results prove that the proposed algorithm outperforms the existing algorithms in terms of execution time, energy consumption, resource utilization, and SLA violations. The proposed algorithm saves energy up to 30% in comparison to existing algorithms. The utilization of resources is also significantly increased by to 30%. In terms of SLA violations, the proposed algorithm reduced SLA violations up to 50%.

Introduction

Cloud computing is serving as backbone infrastructure of the future generation IT industry. The tremendous increase in service demands of data storage and computations has increased requirement of cloud computing. Easy access to the Internet, smart phones, social media, multimedia, and the Internet of Things (IoT) [1,2] have resulted in large-scale production of data. The processing and storage of this data is a complex task for smaller organizations. Cloud computing provides on-demand resources such as computational resources, memory and data storage. The essence of Cloud computing is that it removes technology barriers and allows organizations to focus on their core business without bothering about the management and availability of their IT infrastructure [3].

To meet the ever increasing service demands, datacenters are comprised of thousands of servers which causes high energy consumption and the emission of greenhouse gases like carbon dioxide [4]. Higher energy consumption enlarges the running cost [5] and results in reduction of profits. Albert Greenberg [6] discussed that a datacenter with 50,000 servers can consume over 100,000,000 kW/H of energy. Miyuru Dayarathna [7] mentioned that the energy cost of the datacenter doubles every 5 years. Power bills became a major concern for today's datacenters. A study by Van Heddeghem [8] shows that, energy consumption in datacenters in 2012 was 270 Tw/h [9]. Energy consumption has a Compound Annual Growth Rate of 4.4% in 2007–2012 duration. Because of these problems, the area of energy consumption attracted researchers worldwide and now, it became a most important area of research. Techniques for energy efficiency can be categorized into two types [10], a detailed hierarchy of these techniques is shown in Fig. 1.

Energy efficiency is achieved both at the hardware and software level. Hardware-level optimization saves energy by switching logic gates, which leads to an optimized architecture. At the software level, energy is saved by optimization of tasks and efficient utilization of resources. In the datacenter, an idle server consumes 70% of its energy as compared to its peak time [11]. The average CPU utilization of over 5000 servers was between 10% and 50%. Energy consumption can be

* Corresponding author.
E-mail addresses: m.bilal@ieee.org (M. Bilal), rajamajid@xmu.edu.my (R. Majid Mehmood).

https://doi.org/10.1016/j.seta.2021.101210
Received 23 August 2020; Received in revised form 9 January 2021; Accepted 21 March 2021
Available online 15 April 2021

lessened with efficient utilization of resources in the datacenter. Researchers have proposed many methods and approaches to reduce energy consumption in datacenters. One of the methods is the efficient utilization of available resources in the datacenter by using scheduling algorithms for well-organized scheduling of tasks to the servers [12]. The major purpose of the algorithms is to turn off or hibernate maximum servers to save energy.

Focus of these algorithms is not only efficient utilization of resources but they also focus on to minimize the completion time of the tasks. The second technique used to cope with energy consumption is Dynamic Voltage/Frequency Scaling (DVFS) [13]. The DVFS is typically used, where an application is not time critical [14]. In DVFS, processors are operated on different voltage and frequency combinations to save power. Voltage supply depends upon the requirements of servers and the job they are executing. The power consumption of integrated circuits is proportional to the voltage and frequency. When low voltage and frequency is provided to the system, energy is saved, but the execution time of the job is increased. DVFS technique has a drawback related to runtime performance. When frequency is low, performance of systems is decreased, resulted in the SLA [15] violations.

Provisioning of resources is challenging in terms of management due to variations in workload demands. Workload demands of various datacenters transform within the time of the single day [16]. To cope with the abrupt changes in workload, scaling is used. There are two types of scaling in datacenters [17], vertical scaling, and the horizontal scaling. In horizontal scaling, virtual machines are added and removed from the servers. As workload demands can change within seconds, horizontal scaling takes relatively more time in completion. On the other hand, vertical scaling adds and remove resources from existing virtual machines. Vertical scaling is achieved at run time and within a few seconds [17]. In this research work, above discussed issues are addressed and an algorithm "Cost-Based Energy Efficient Scheduling Technique for DVFS Systems (CEEST)" is proposed. The Proposed algorithm aims to save energy consumption, handle high workloads, and decrease SLA violations using DVFS technique.

Performance of the CEEST algorithm is evaluated through extensive simulations. Heterogeneous systems were used for simulation purposes, as they are capable of handling complex and diverse nature of tasks. Whereas homogeneous systems work for the similar nature of tasks. Real-world workflow PlanetLab was also used to ensure fair simulations. Main contributions of this research work are:

- Proposed task scheduling algorithm to reduce energy consumption by efficient utilization of resources.
- Execution of workflow is decreased due to efficient scheduling of tasks on resources.
- If workload changes abruptly, proposed algorithm can handle abrupt changes in algorithm by provisioning of resources.
- Proposed algorithm minimizes SLA violations by scheduling tasks on virtual machines, which can finish task execution in deadline constraints.
- The proposed algorithm saves energy consumption and increase resource utilization up to 30% and reduces SLA violations up to 50%.

The rest of the paper is organized as follows. In Section "Related Work", the proposed work is compared with related research efforts. Section "Modeling and Design of Datacenter" describes the system model, energy model, and cloud application. In Section "Cost-based Energy Efficient Scheduling technique for DVFS systems (CEEST)" proposed algorithm CEEST, and the system architecture is discussed. Experimental results and evaluation analyses are presented in Section "Simulation and evaluation of results". Section "Conclusion" concludes the whole paper.

Related work

Scheduling algorithms are divided into two types, heuristic based, and random guided search based. Heuristic based algorithms perform well for scheduling as compared to random guided search. Heuristic algorithms are further divided into various types like rule-based heuristic [18], list-based heuristic [19] and *meta*-heuristic algorithms [20]. Rule based heuristics algorithms, include Min-Min [21], Max-Min [22], Minimum completion time (CMT) [23] and Most efficient server first (MESF) [24]. Rule-based algorithms are simple, they do not perform well when workload is high.

List-based heuristics algorithms efficiently reduce execution time of workflows. These algorithms have two phases, first phase calculates priority of tasks, then second phase schedules tasks to appropriate processors. Haluk et al. [25] proposed a list scheduling algorithm for heterogeneous systems named Heterogeneous Earliest Finish Time (HEFT). This algorithm selects tasks by descending order of their upward rank and schedules them to processors which can execute in earliest time. Upward rank of task is maximum distance from current task to exit node. This algorithm effectively reduces execution time, but computational cost is high and do not perform well when workload is high. Shahzad

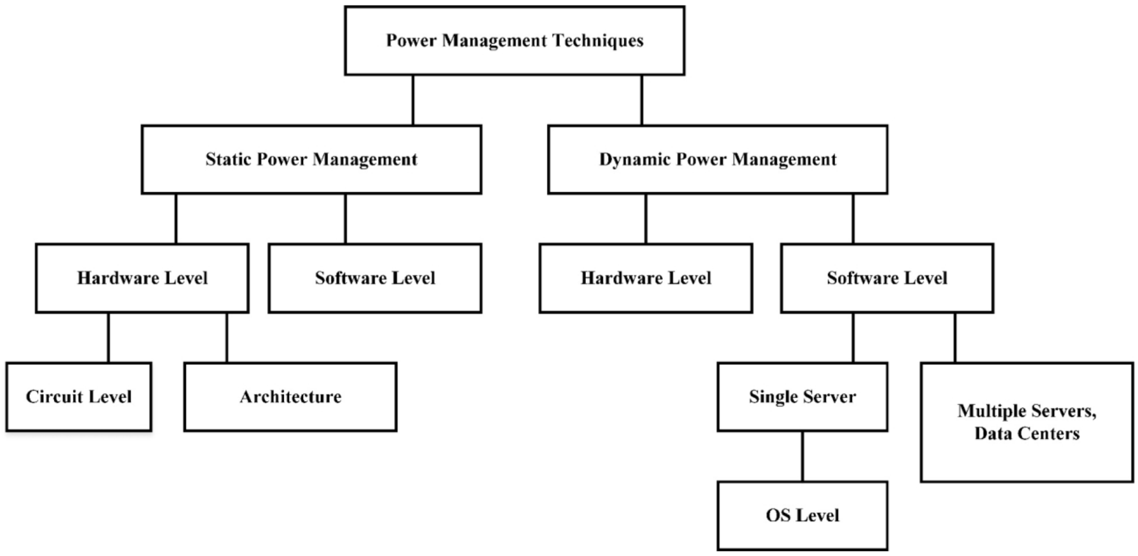

Fig. 1. Hierarchy of power management technique.

et al. [26] proposed a scheduling algorithm named “Parental Prioritization- Based Task Scheduling Algorithm in Heterogeneous systems.” In this algorithm, a parental priority queue (PPQ) is used for scheduling of tasks. This algorithm calculates priority of tasks by ascending order of their downward rank and cost. Then tasks are scheduled to processors according to order of their priority. This algorithm effectively reduces execution time, and cost of high workload but do not focus on SLA violations. Xiaozhong et al. [27] proposed a task scheduling algorithm based on priority list and task duplication in cloud computing environment. This algorithm constructs a task scheduling queue by calculating priority of tasks and reduces communication latency between tasks. Then it schedules tasks to virtual machines which can finish execution in minimum time. This algorithm performs well for load balancing but do not maintain QOS. List-based heuristics algorithms achieve great performance in minimization of execution time, but they do not focus on resource utilization.

Meta-Heuristic algorithms include bio inspired algorithms like Genetic Algorithms [28], Ant colony optimization [29,30], Particle swarm optimization [31] and Artificial Bee Colony [32]. Bappaditya et al. [33] proposed modified particle swarm optimization algorithm. It saves energy by minimizing average scheduling length of workflow but do not focus on QOS. Daun et al. [34] proposed adaptive incremental genetic algorithm for efficient task scheduling. This algorithm automatically adjusts genetic parameters and divides tasks in small groups. It achieves great performance in finding optimal solution. Reddy et al. [35] proposed energy efficient scheduling algorithm using hybrid of genetic and particle swarm optimization. This algorithm efficiently reduces energy consumption and have low convergence time. Meta-heuristic algorithms perform well for scheduling problem, but they need a lot computational power to find optimal solution.

Researchers also proposed algorithms that reduce power consumption by management of virtual machines and servers in datacenter. Qingjia Huang et al. [36] proposed an Enhanced Energy Efficient Algorithm (EES) for parallel applications in clouds. This algorithm saves power by slacking noncritical tasks while fulfilling SLA requirements. It assigns lower frequency to non-critical jobs and reassigns tasks to appropriate slots for lower energy consumption. This algorithm efficiently minimizes energy consumption, but SLA violations of non-critical tasks are increased. Anton Beloglazov et al. [37] proposed energy-efficient heuristics for efficient management of datacenters. This algorithm uses a utilization threshold named as “THR” to find time to migrate virtual machines. It sets upper and lower threshold of servers and maintains utilization of all virtual machines on a server within threshold. If utilization of a server falls below the lower threshold, then algorithm migrates virtual machines from that server to other servers and switches server to sleep mode to eliminate idle power consumption. If the utilization exceeds the upper threshold, then some virtual machines are migrated to avoid overloading. This method can improve the energy efficiency of the datacenter while maintaining QoS, but it sacrifices system performance. Mekala et al. [38] proposed an energy-efficient resource ranking and utilization factor-based virtual machine selection (ERVS) algorithm. It evaluates overloaded and underloaded servers and migrates to appropriate servers. This algorithm effectively reduces energy consumption, cost and SLA violations but do not cope with changing workload demands.

Chia-Ming Wu et al. [39] proposed a Green Energy Efficient Algorithm Using DVFS (GEE) that minimizes energy utilization by applying DVFS on processors. The algorithm takes jobs in the form of frequencies and the SLA level as input. A scheduling algorithm calculates the weights of servers and VMs to allocate a job. VM manager assigns resources to the job based on a scheduling algorithm decision and guarantees execution performance of job. The algorithm fulfills SLA when assigning resources to user. This algorithm efficiently reduces energy consumption, but execution time of jobs is slightly high and does not consider input workload. Zhuo Tang et al. [14] proposed DVFS enabled Energy Efficient Workflow task scheduling (DEWTS). DEWTS uses the slack time of inefficient processors and turns off inefficient processors. It calculates the scheduling order of all the tasks and gets the makespan and deadline of all the tasks using the HEFT algorithm. Then algorithm selects underutilized processors and moves tasks from underutilized processors to other processors and merges these processes. It saves energy by turning off inefficient processors, but it is unrealistic approach and increases time complexity. Youwei et al. [40] proposed an energy-efficient scheduling algorithm (EEVS). EEVS focuses on dynamic VM scheduling to save energy and complete jobs in deadline constraints. This algorithm used a metric called “optimal performance-power ratio” for scheduling VMs to physical servers (PM) and assigns VM to PM with a greater performance-aware ratio. The algorithm checks for optimization after a fixed interval. At each interval, it assigns jobs to virtual machines to complete the job in the deadline, while keeping energy consumption low. The frequency to operate each core is then determined by the total amount of resources required by virtual machines. They claimed to achieve 20% reduction in energy usage and 8% increase in processing capacity. But this algorithm does not focus on resource utilization. Monire Safari et al. [41] proposed an energy-efficient optimization model for time-constrained workflows in DVFS enabled processors. This algorithm reduces energy consumption and resource utilization and fulfills SLA. This algorithm finds a sequence of VMs for workflow according to deadline constraints. It ranks tasks according to deadlines and assigns resources to the tasks which can complete a job in a deadline. Mostly energy efficient algorithms are not good for problems with multiple criteria that are required simultaneously. This algorithm performs well for multi criteria like energy consumption, resource utilization, execution time and SLA but do not cope with changing workload. Hadeer et al. [42] proposed a smart energy and reliability aware scheduling algorithm for workflow execution in DVFS-enabled cloud environment. It splits the deadline of workflows across tasks then it allocates tasks to virtual machines without compromising deadlines. This algorithm efficiently reduces energy consumption and minimizes SLA violations but do not focus on utilization of resources. DVFS algorithms reduces energy consumption effectively but system performance is compromised.

Many researchers have focused on either reducing energy consumption or minimizing completion time. When energy consumption is decreased, the problem of SLA violations rises. DVFS algorithms minimizes energy consumption but system performance is compromised and cannot cope with changing workload demands. The proposed algorithm reduce energy consumption by optimizing the utilization of systems in the datacenter and also maintains QoS. The SLA violations are reduced by efficient task scheduling to complete job in deadline constraints and provisioning of resources according to workload demands. Table 1 summary of work discussed in this section.

Modeling and design of datacenter

This section describes datacenter design and input model.

Datacenter model

DVFS enabled datacenter with a large number of heterogeneous computing servers was considered for the evaluation of proposed algorithm. Computing servers have different processors, memory and bandwidth requirements.

Suppose there are n servers, then a set of servers S can be represented as:

$$S = \{s_1, s_2, s_3, ..., s_n\}$$

Where, $R \subseteq S$, such that $R = \{r_1, r_2, r_3, ..., r_m\}$, is a set of resources of server s_i. The execution time of a job depends on the resources of the server. For example, if a server has capacity of executing 500 million instructions per second (MIPS) and the job length is 5000 MIPS then the

Table 1
Summary of related work.

References	Scheduling Objective	Technique	Comparison Matrices
[14]	Task Scheduling	DVFS	Resource UtilizationTask SchedulingEnergy Saving Ratio
[25]	Task Scheduling	List-Based	Schedule Length Ratio (SLR) Speedup
[26]	Task Scheduling	List-Based	Execution TimeSLREfficiency
[27]	Task Scheduling	List-Based	SLR
[33]	Task Scheduling	Evolutionary	Execution Time
[34]	Task Scheduling	Evolutionary	Convergence TimeExecution Time
[35]	Energy Efficiency	Evolutionary	Energy ConsumptionConvergence Time
[36]	Energy Efficiency	Evolutionary	Energy Conservation
[37]	Energy Efficiency	List-Based	Energy ConsumptionSLA Violations
[38]	Resource Utilization	List-Based	Resource UtilizationEnergy ConsumptionNumber of Migrations
[39]	Energy Efficiency	DVFS	Execution TimeEnergy Consumption
[40]	Energy Efficiency	List-Based	Energy ConsumptionFailed VMs
[41]	Energy Efficiency	DVFS	Execution TimeSLA ViolationsEnergy Consumption
[42]	Energy Efficiency	DVFS	Energy Consumption

execution time of the job is shown in the Eq. (1).

$$T_{exec} = \frac{j_l}{S_m} \tag{1}$$

Where j_l is job length and s_m is MIPS of the server. In this example, the job will be finished in 10 s. In DVFS frequency is adjusted by changing the voltage level. If a server performs at the full frequency, then it will finish the job early, but more energy will be consumed. If half voltage is applied to the server then this job will be finished in 20 s. This will save power, but the execution time is increased.

If there are k virtual machines in the datacenter then, set of virtual machines V can be presented as:

$$V = \{v_1, v_2, v_3, ..., v_k\}$$

Each virtual machine has its own set of resources. For example, if a job length is 5000 MIPS and three virtual machines are available with the resources 800, 500 and 200 respectively. This job can be allocated to any three virtual machines. VM0 will finish this job in 7 s, VM1 will finish in 10 s, and in case of VM2 job will be completed in 25 s. The purpose of scheduling is to allocate this job to a virtual machine that consumes less energy and complete the job in deadline constraint to avoid SLA violations.

Each virtual machine and server have a weight W_v and W_s associated with it. Weights can be used to calculate the priorities of servers and virtual machines. The priority of the server and virtual machine can be calculated from the Eq. (2).

$$\frac{P\alpha 1}{W} \tag{2}$$

Where P and W are the priority and weights of a server and VM respectively.

SLA is an important contract between a user and a cloud provider. Every user has an SLA level. If N is the maximum level of the contract, then the SLA level of a user will be between

$$0 \leqslant L \leqslant N - 1$$

Where, L is a contract of the user. Each SLA level has different number of virtual machines and resources. If the SLA level is low, less resources will be allocated to the user. This will result in less energy consumption, but the execution time of the job will also increase. Table 2 shows an example of an SLA level and their respective available virtual machines. In many network applications, deadlines are very important. SLA violations occur when a task finishes its execution after its deadline is passed. It can be represented it by the Eq. (3).

$$T_{delay} = T_d - T_{exec} \tag{3}$$

Where, T_{delay}, T_{exec} and T_d are the delay, execution time and deadline time of the task t respectively.

DVFS enabled processors were used in this simulation evaluation. Voltage V is given to a processor P. Every processor runs at a different pair of voltage (V) and frequency (f) pairs. If less voltage is applied, the processor will run at a low frequency. This will cause low energy usage but an increase in the execution time of tasks. Scheduling Interval T_{int} is the time to check the state of servers in the datacenter after every fixed interval. If scheduling interval is too small, then servers will change states frequently and will load the servers.

Application model

Parallel tasks, also called jobs are used in this evaluation. Let T is a set of L tasks then T can be represented as:

$$T = \{t_0, t_1, t_2, ..., t_{n-1}, t_L\}$$

Tasks execute in parallel and child tasks are dependent on parent tasks. They will start execution when parent tasks will finish their execution. These tasks can be represented as a directed acyclic graph (DAG). Fig. 2 and Fig. 3 show example of DAG with 10 tasks. The first task is entry tasks, and the last task is the exit task. If the graph has two or more entry tasks and two or more exit tasks, then a new virtual entry task and a virtual exit task are introduced. Task 0 and 1 are parallel tasks. No other task will start its execution because child tasks start their execution only after the parent task. Task 2 and 3 depend on task 0. They start execution after completion of task 0. Tasks 4, 5 and 6 depend on task 1. They are parallel tasks, all of them can execute at the same time. Task 3 has one dependent task 7, while task 4 has two dependent tasks 8 and 9. Tasks 2, 7, 8, 9, 5 and 6 have no dependent tasks. After the execution of these tasks, the application will finish. DAG G can be represented as:

$$G = \{N, E\}$$

Where G represents DAG, N represents node or task t, and E represents edges of task t. Each edge between two tasks t_i and t_j has weight called communication delay c. This delay occurs when tasks t_i and t_j are scheduled on different processors. If both tasks are scheduled on the same processor, then the communication delay will be 0.

Energy model

In complementary metal–oxidesemiconductor (CMOS) circuits, power consumption comprises of dynamic power consumption and static power consumption. Dynamic energy consumption is the most important factor. DVFS technique reduces dynamic power consumption

Table 2
Example of SLA level and related number of VMs.

Level	Number of VMs
0	1–200
1	1–400
2	1–600
3	1–800
4	1–1000

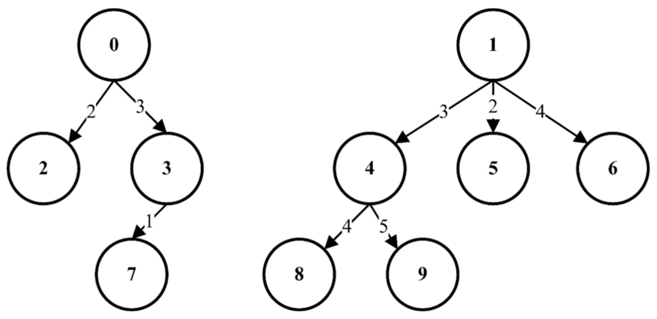

Fig. 2. Parallel tasks.

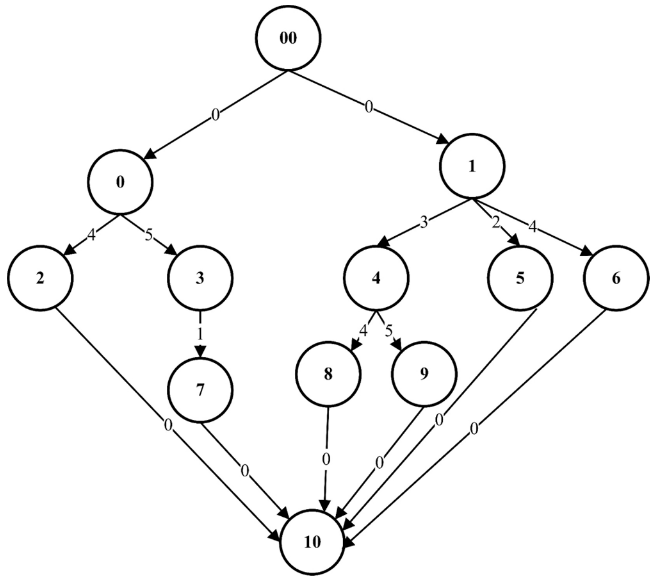

Fig. 3. A typical DAG.

by scaling up and down voltage and clock frequency. The relationship between $P_{dynamic}$, voltage, and frequency is shown in the Eq. (4) [14]

$$P_{dynamic} = K \times V_{j,s}^2 \times f_s \tag{4}$$

Where K is constant and depends on device capacity. $V_{j,s}$ is the voltage level s supplied to the processor j. f_s is the frequency of relevant $V_{j,s}$. Energy dynamic $E_{dynamic}$ can be represented by the Eq. (5)

$$E_{dynamic} = P_{dynamic} \times \Delta T \tag{5}$$

Where ΔT is the time period. The running time of a server can be divided into busy time and idle time. Energy consumption where a server is busy E_{busy} can be calculated by Eq. (6) [14].

$$E_{busy} = \sum_{i=1}^{n} k \times V_{i,P_{j,s}}^2 \times f_{i,p_{j,s}} \times t_{i,j}$$

$$E_{busy} = \sum_{i=1}^{n} P_{dynamic} \times t_{i,j} \tag{6}$$

Where, $t_{i,j}$ shows the execution time of task t_i on processor P_j. $V_{i,p_{j,s}}^2$ shows voltage s supplied to task t_i on processor p_j. $f_{j,s}$ shows clock frequency of processor p_j at the voltage levels s. Zero voltage cannot be applied when processors are idle. For idle time voltage should be at the lowest level. Energy used during idle time E_{idle} can be represented by the Eq. (7) [14].

$$E_{idle} = \sum_{j=1}^{p} k \times V_{j,\,lowest}^2 \times f_{j,lowest} \times t_{j,\,idle}$$

$$E_{idle} = \sum_{j=1}^{p} p_{j,\,idle} \times t_{j,\,idle} \tag{7}$$

Where, $t_{j,idle}$ represent the idle time of processor P_j. $F_{j,lowest}$ shows the lowest frequency of the processor and $V_{j,lowest}$ shows the lowest voltage level s of the processor. From Eqs. (6) and (7), total energy consumption can be calculated. Total energy consumption E_{total} can be represented by the Eq. (8).

$$E_{total} = E_{busy} + E_{idle} \tag{8}$$

Cost-based Energy Efficient Scheduling technique for DVFS systems (CEEST)

Energy efficiency in datacenters is a challenging job, whenever low energy mode of servers is utilized to save energy, result is degradation in performance in terms of job completion time and SLA violations. There is always a trade-off between energy saving and execution time of the task. A Cost-based Energy Efficient Scheduling Technique (CEEST) for DVFS systems is proposed, which considers the challenge of energy saving and SLA violations. CEEST schedules jobs to reduce energy consumption, handles high workloads, minimizes task completion time and minimizes SLA violations. The architecture of CEEST comprises of job submission, job manager, scheduling algorithm, scalability algorithm, server manager, DVFS controller and servers/VMs. Fig. 4 shows the complete proposed system architecture.

Jobs are submitted to system in the form of length, communication cost, and SLA level. Job manager gets a list of servers and VMs and their states from the server manager and gives it to a scheduling algorithm. After getting results from the scheduling algorithm, they are submitted to the server manager. The server manager then assigns required resources to the job. Scheduling algorithm receives a list of servers and VMs from the job manager and calculates appropriate resources for allocation to the job. If no virtual machine can complete the job in deadline, then it sends a request to the scalability algorithm.

If scheduling algorithm cannot find a virtual machine which can complete the job in deadline given, then it sends a request to scalability algorithm. The scalability algorithm chooses a virtual machine and scales it. If system remains unable to complete job in the deadline after scalability, then it will be SLA violation.

Server manager has a list of states of servers and virtual machines. It receives results of the scheduling algorithm from the job manager and allocates resources to the job. Server Manager also monitors the server's utilization after some interval and migrates virtual machines from underutilized servers to save power. DVFS controller intelligently supplies adequate voltage and frequency to the servers based on their utilization. Servers and VMs execute the jobs. The scheduling algorithm decides which resources should be given to a job.

There are three phases in CEEST algorithm. The first phase is VM mapping, the second phase is job scheduling, and the third phase is system optimization.

VM mapping

The aim of VM Mapping is to allocate an appropriate server to run VM on it. In the first step, weights of all servers are calculated. Weight W_s of server s_i from the Eq. (9).

$$Ws = C_p \times R_{used}; \quad R_{used} > 1 \tag{9}$$

Where, C_p is the unit power cost of the server. R_{used} are used resources of the server. If server's resources are not utilized by any virtual machine, then R_{used} will be 1 instead of 0. In second step servers will be

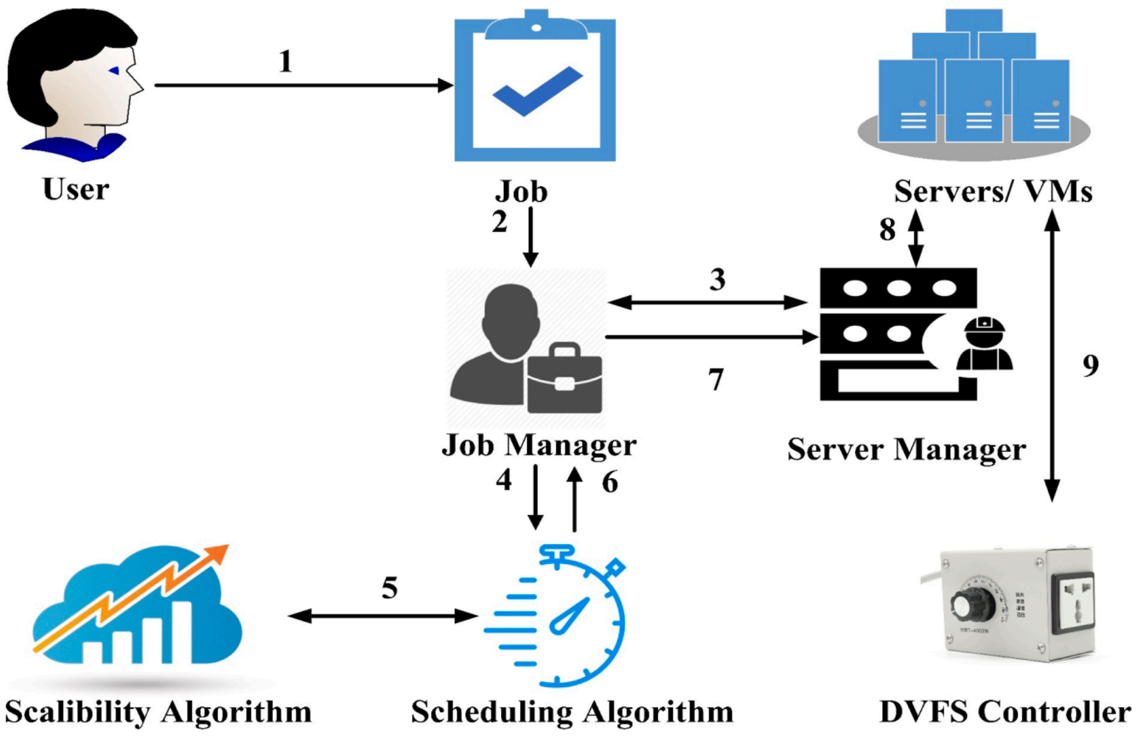

Fig. 4. Proposed system architecture.

sorted in ascending order of their weights. A virtual machine is assigned to the server if it has lowest weight and has required resources. Selected server s_{selec} can be found from the Eq. (10).

$$s_{select} = min(W_i)\ ; \quad s_{select} can\ run\ VM \tag{10}$$

If no server is capable of hosting a VM, then the job manager sends a request to the server manager. If there are any powered off servers, server manager starts the server and assigns VM to it. If there are no available servers, then the server manager finds a solution by migrating VMs from one server to the other. If any of the VMs can be migrated from the server, then it chooses the best destination server for this VM using Eq. (10). After creating space, it creates VM on that server. If again no server can host VM, then it is considered to be a failure. Algorithm 1 describes the VM mapping.

```
Algorithm 1: VM Mapping phase ()
Input: Server List, Vm
Output: select server to run VM
1: for each server in server list
2:   Calculate weight from Eq. (9)
3: end for
4: sort server list in ascending order of weights
5: for each server in server list
6:   choose server which can host VM
7: end for
8: if (no server can host VM)
9:   for each server in server list
10:    migrate VMs from a server to another server
11:  end for
12: end if
13: if (no server selected to host VM)
14:   return error
15: else
16:   return selected server
17: end if
```

Job scheduling

Job Scheduling algorithm finds an appropriate virtual machine for the current job. The first step is to find the weight of VMs. Weight W_v of virtual machine v_i is calculated from the Eq. (11).

$$Wv = C_p \times \sum_{i=1}^{n} T_{exec} \tag{11}$$

Where, Cp is the unit power cost of the server s_i on which virtual machine v_i is running. $\sum_{i=1}^{n} T_{exec}$ is sum of the execution time of tasks t on VM v. Eq. (11) ensures to choose virtual machine with lower power cost. Job is assigned to a virtual machine which is capable of starting execution of job as early as possible. T_{exec} of task t can be calculated by Eq. (12).

$$T_{exec} = (T_L / V_M) + C \tag{12}$$

Where, T_L, V_M and C are the task length, MIPS of VM and communication cost of task t respectively. In the second step, virtual machines are sorted in ascending order of their weights. In the third step, virtual machine is chosen which can execute the job in the required deadline. The virtual machine selected for execution of job is found using Eq. (13).

$$V_{select} = min(W_i)\ ; \ V_{select}\ can\ finish\ t\ in\ T_d \tag{13}$$

If no VM is capable of finishing the current job in required deadline, then the system vertical scales a VM to complete the job in the deadline. If again, even after scaling, job is not completed in deadline, then it is considered to be a failure, and this causes SLA violation. Algorithm 2 describes the job scheduling.

```
Algorithm 2: Job Scheduling Phase ()
Input: VM List, Job Length
Output: Select VM to schedule job
1: for each vm in vmlist
2:   calculate weight of vm from Eq. (11);
3: end for
4: sort vmlist in ascending order of weights;
5: for each vm in vmlist
6:   select VM which can finish job in deadline
7: end for
8: if (no VM selected)
9:   select a VM with highest resources and upscale it
10: end if
11: return selectedvm
```

System optimization

Purpose of third phase of algorithm is system optimization to achieve maximum performance with minimum energy consumption. In the first step, server manager inspects states of servers after a fixed scheduling interval. Algorithm categorizes underutilized servers as infected servers. In the second step, algorithm selects the targeted server for each virtual machine of the infected server. Target server is calculated from Eq. (10). In the third step, server manager migrates virtual machines from the infected servers to the targeted servers. If no server can host virtual machine of infected server, virtual machines are not migrated. After migration of virtual machines, infected servers are powered off. Algorithm 3 defines the system optimization phase.

```
Algorithm 3: System Optimization ()
Input: Server List, Time
1: for each server in server list
2:   if (server is underutilized)
3:     add server to underutilized list
4:   end if
5:
```

(*continued on next page*)

(*continued*)

```
end for
6: for each server in underutilized list
7:     if (vms can be hosted on another server)
8:         migrate all VMs
9:         power off servers
10:    end if
11: end for
```

Motivational example

To understand the proposed method, consider a simple DAG shown in Fig. 3. Consider datacenter has three physical servers and three virtual machines. Virtual machines have 300, 400 and 700 MIPS respectively. Table 4 and Table 5 show details of servers, virtual machines and tasks. In the first phase, virtual machines are assigned to servers according to Eq. (10). VM0 required MIPS are 400. Machine1 has lowest weight according to Eq. (10) but it cannot run VM0 because it has 300 MIPS.

Server0 can host VM0, therefore, it will host VM0. VM1 and VM2 are assigned to server1 and server2, respectively. Second phase is job scheduling. At start there are two tasks 0 and 1. According to Eq. (13) weight of VM0 is 10, VM1 is 5 and VM2 is 15. VM having less weight can also execute task0 in deadline time, which is 4 s, calculated using Eq. (12). Therefore, task0 is assigned to VM1. Now for task1 weight of VM0 is 10, VM1 is 40 and VM2 is 15. Here VM1 is chosen because it has less weight and has enough MIPS to complete the job in 8 s. At the time interval 4 task0 finishes its execution. Now task2 and task3 are scheduled. Table 6 shows assignments of all jobs.

Consider the case of task9, its deadline is 3 and no virtual machine is capable of finishing it in given deadline. VM2 has less weight, system will vertically scale up VM2 to 1500 MIPS. Now job is being finished in required deadline. Third phase is system optimization. Suppose server underutilization threshold is 50%. In this scenario server2 is underutilized, algorithm will select it as infected server. No other server can host VM2, therefore no migration will happen in this scenario. If another server can host VM2, then VM2 will be migrated from infected server and server2 server will be powered off.

Simulation and evaluation of results

The proposed algorithm Cost-based Energy Efficient Scheduling Technique (CEEST) for DVFS systems is simulated and results are compared for evaluation. Famous algorithms in literature; GEE [39], PPEFT [26], and Heft [25] are simulated and results are compared with CEEST.

Cloudsim Plus[43] based on Cloudsim [44], is used for simulating scenarios discussed in Section 3 and Section 4. Cloudsim is a framework for modeling and simulating cloud datacenters. Cloudsim Plus is an extension of Cloudsim to simulate realistic scenarios. A set of heterogeneous servers is used in the simulation. AMD-Athlon-64 processor is selected for processing, voltage and frequency pairs of processors are given in table 3. All servers have MIPS capacity ranging from 1000 to 10,000 MIPS. Directed acyclic graphs and real-world workflows are implemented to evaluate the proposed algorithm.

Performance of proposed algorithm is evaluated on the basis of execution time, energy consumption, utilization and SLA violations. Execution time the time required by the servers to finish execution of

Table 3
shows the voltage and relative frequency pairs of common processors.

Level	AMD Athlon −64		Intel Pentium M		AMD Opteron 2218	
	Voltage (V)	Speed (%)	Voltage (V)	Speed (%)	Voltage (V)	Speed (%)
0	1.5	100	1.484	100	1.30	100
1	1.4	90	1.463	85	1.25	88
2	1.3	80	1.308	70	1.20	78
3	1.2	70	1.180	57	1.15	67
4	1.1	60	0.956	43	1.10	56
5	1.0	50			1.05	44
6	0.9	40				

Table 4
Details of servers used in motivational example.

Server Id	MIPS	Cost
0	500	10
1	300	5
2	1500	15

Table 5
Details of tasks used in Motivational Example.

TaskID	Length	CommunicationCost	Deadline
0	1000	0	4
1	3000	0	10
2	1500	2	6
3	2500	3	7
4	2000	3	10
5	1000	2	4
6	3500	4	10
7	2000	1	6
8	1000	4	7
9	4000	5	3

workflow/task graph. Energy consumption is calculated by taking the average of energy consumed by the servers in the datacenter. Utilization is the ratio of MIPS allocated to total MIPS of the server at a time interval. SLA violation occurs if the job cannot complete its execution in given time constraints. If the algorithm remains unable to create a virtual machine in servers, then it will also be an SLA violation. Randomly generated graphs are used to evaluate the performance of the algorithms. Input parameters for DAG experiments are given in Table 7.

In the first simulation scenario, performance of above-mentioned algorithms is evaluated by incrementing number of tasks for scheduling with fixed number of servers and virtual machines. The number of servers is 20 and the number of VMs is 50. Fig. 5 compares the time which different algorithms are taking to complete the execution of the workflow assigned. The proposed algorithm CEEST outperforms all algorithms in terms of execution time and completes execution of workflow in lesser time. It is the intelligence of the algorithms to schedule jobs to the VMs which are capable of starting its execution as early as possible. Fig. 6 shows energy consumption in (kw/h) by each of the algorithm to execute the assigned tasks. Comparison of results prove that CEEST algorithm reduces energy consumption from 20% to 30%. CEEST performs better in energy consumption because if a server is underutilized, then the optimization process turns off the underutilized server which results in significant energy saving. Fig. 7 demonstrates the utilization of servers by each algorithm. CEEST algorithm utilizes the maximum resources of servers to complete the execution of workflow. CEEST algorithm turns off unused or underutilized servers and schedules virtual machines and tasks to other available servers, resulting in increase in the utilization of resources of servers. Fig. 8 compares the SLA violations of made by different algorithms during execution of tasks. The proposed algorithm reduces the number of SLA violation up to 50% because it makes sure to assign jobs to the virtual machines which are capable of executing it in the required deadline. The results prove that the CEEST algorithm improves overall performance of the system. With the increase in workflow length, CEEST algorithm performs outperforms other algorithms.

In the second scenario of the simulation, number of tasks are fixed while incrementing number of servers/hosts and VMs. The number of tasks is fixed to 1000, number of VMs is varying from 50 to 250 with an increment of 50. The number of tasks is fixed to 1000 and the number of hosts is 20 to 100 with an increment of 20 in every simulation step. Fig. 9 shows a comparison of the execution time of algorithms. With every increase in the number of hosts execution time of tasks declines. CEEST remains winner with completion of tasks execution in minimum time. The energy consumption of algorithms is represented in Fig. 10. Energy

Table 6
Assignment of all tasks of Motivational Example.

TimeInterval	TaskArrived	VMAssigned	ExecutionTime (s)	CommunicationCost	FinishedExecution
0	0	1	4	0	
	1	0	8	0	
4	2	1	5	0	0
	3	2	4	3	
8	4	1	7	3	1
	5	0	3	0	
	6	0	9	6	
9					4
11	7	2	3	1	3
					5
15					7
17					6
19	8	1	4	0	4
	9	2	8	5	
23					8
26					6
32					9

Table 7
Input Parameters for DAG Graph.

Name	Value	Description
Task Length	51000–60000 (MIPS)	Increment of 1000
Communication Cost	2–20	If cost is low application is computation intensive otherwise communication intensive
Parallelism Factor	3–10	If parallelism is high DAG length will be small, but more tasks will be parallel and vice versa
Task Deadline	30–600 (S)	
VM MIPS	100–1000	Increment of 100
Host MIPS	500–3000	Increment of 500
Host Power Off	N/A	3 Scheduling Intervals
Underutilization Limit	>50%	Server will be unfertilized if resource utilization is less than 50%.

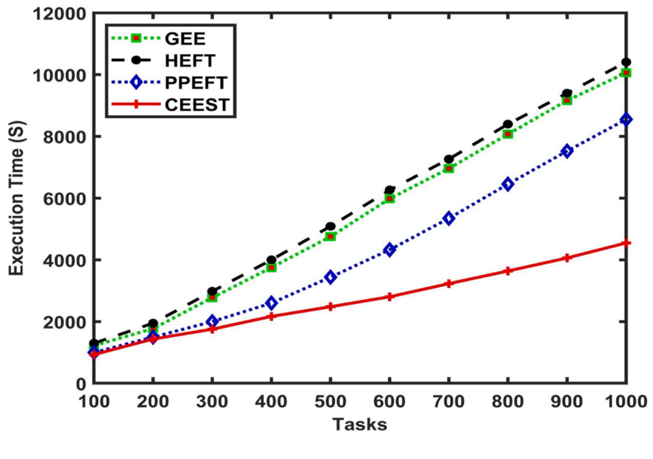

Fig. 5. Execution time for different workflow lengths.

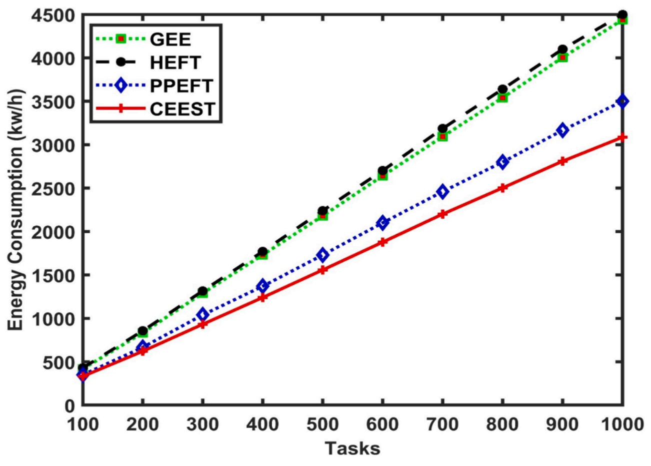

Fig. 6. Energy Consumption for different workflow lengths.

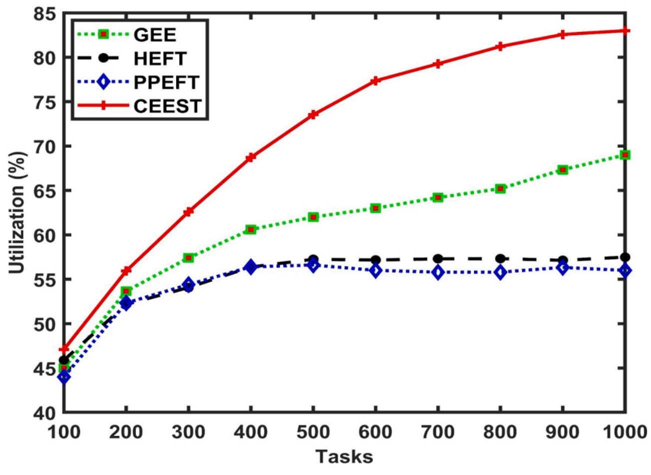

Fig. 7. Utilization of servers for different workflow lengths.

consumption is decreasing with an increase in the number of hosts because execution time is decreased. The proposed algorithm CEEST consumes minimum energy as compared to existing algorithms. Fig. 11 shows the percentage of utilization of resources by the algorithms. Here utilization of servers is decreasing due to an increase in number of hosts. With the increase in the number of hosts resources in the datacenter are increasing and workload remains the same which results in low utilization of resources. Fig. 12 depicts SLA violations committed by the algorithms. With the increase in the number of hosts, SLA violations decrease because more resources are available to execute the tasks. Results prove that CEEST algorithm demonstrates significant performance improvement in terms of major performance metrics of datacenter.

To ensure the viability of CEEST algorithm in terms of energy cost, simulations are performed with servers divided into four groups of dissimilar energy costs. Table 8 shows the energy cost of servers from

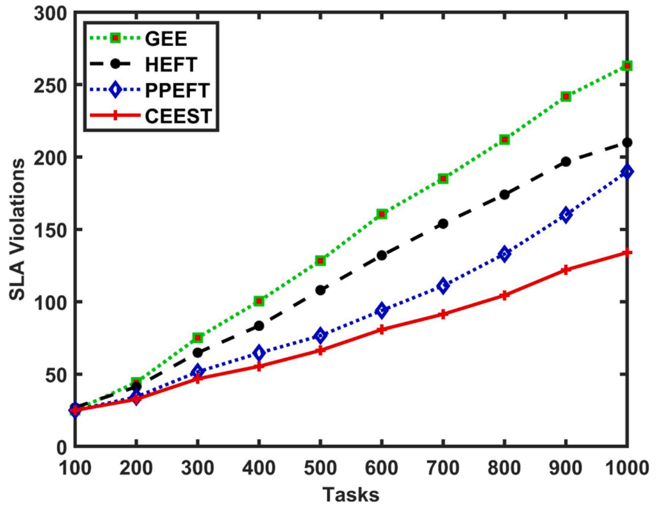

Fig. 8. SLA Violations for different workflow lengths.

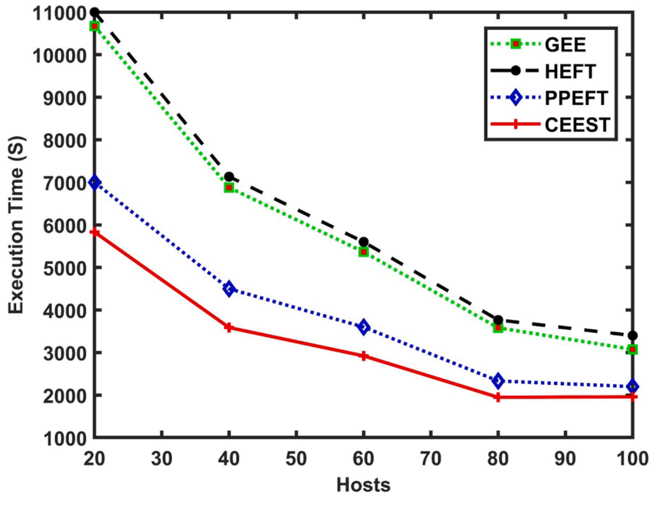

Fig. 9. Execution time when number of server changes.

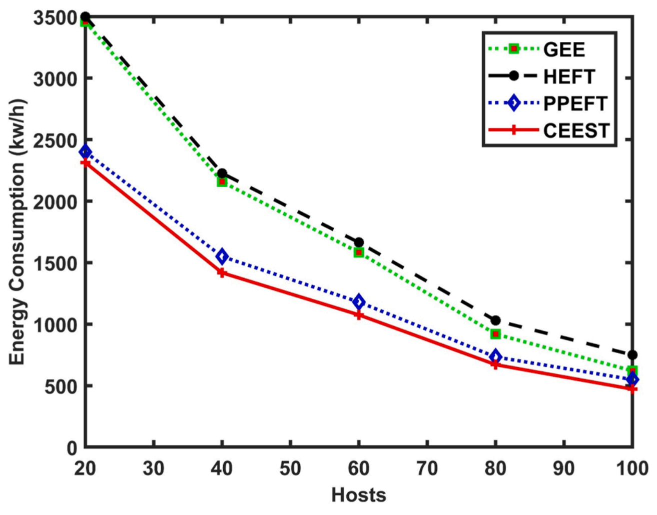

Fig. 10. Energy usage when number of server changes.

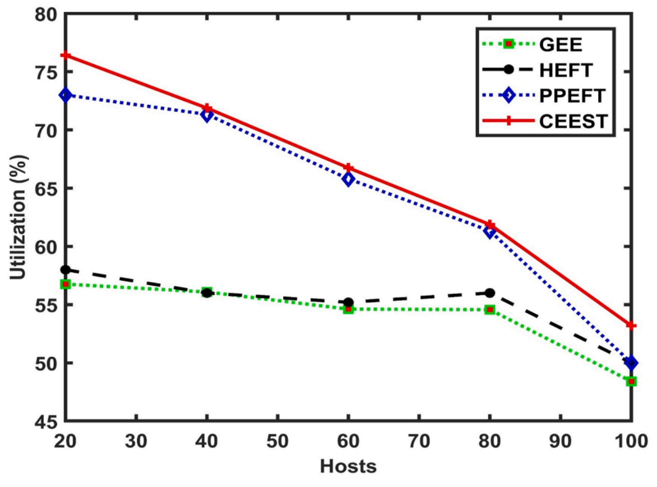

Fig. 11. Utilization of resources when number of server changes.

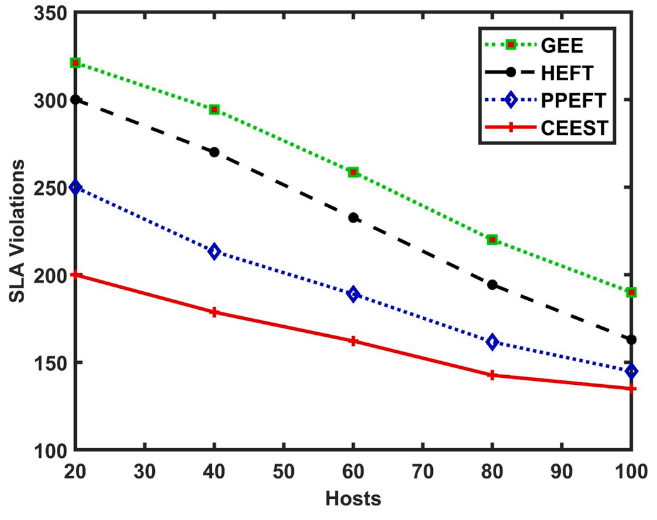

Fig. 12. SLA violations for when number of server changes.

Table 8
Energy Cost of Servers of Each Group.

Group	Cost of Servers
A	1–5
B	6–10
C	11–15
D	16–20

each group. All servers have the identical specifications, and an equal number of servers are configured for each group. Each server can host up to 2 VMs. Number of VMs is 100 and 1000 tasks are configured. Fig. 13 displays energy cost optimization of GEE algorithm. Results show that only 29% of tasks are scheduled on servers with minimum energy cost, i.e., group A. Fig. 14 presents energy cost optimization of CEEST algorithm. Simulation results show that 64% tasks are assigned to servers of low energy cost group A. CEEST optimizes itself by migrating VMs to the servers with lower cost, this results in low overall energy consumption.

CEEST algorithm is also evaluated using PlanetLab [45] workflow. PlanetLab workflow contains real-world CPU utilization traces. In this evaluation, 500 tasks and 200 virtual machines are configured. Fig. 15 shows a comparison of essential performance metrics of a datacenter. Results prove that CEEST algorithm performs far better than the GEE algorithm on real-world workload.

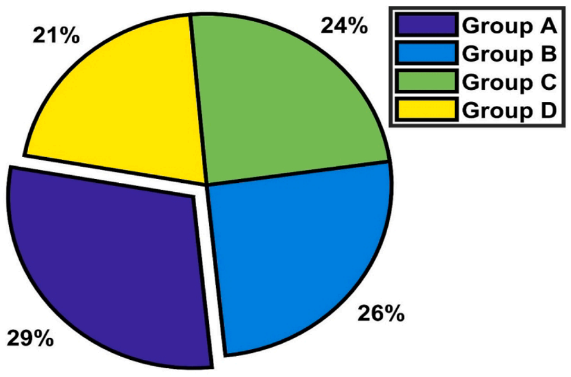

Fig. 13. Energy cost optimization of GEE algorithm.

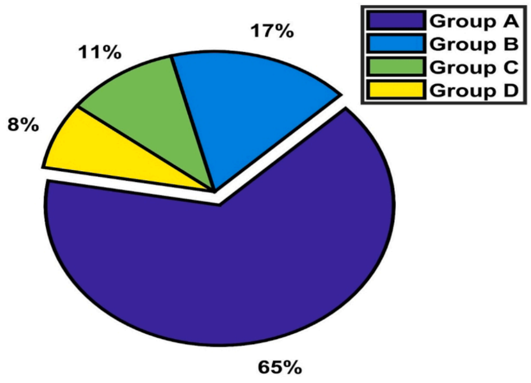

Fig. 14. Energy cost optimization of CEEST algorithm.

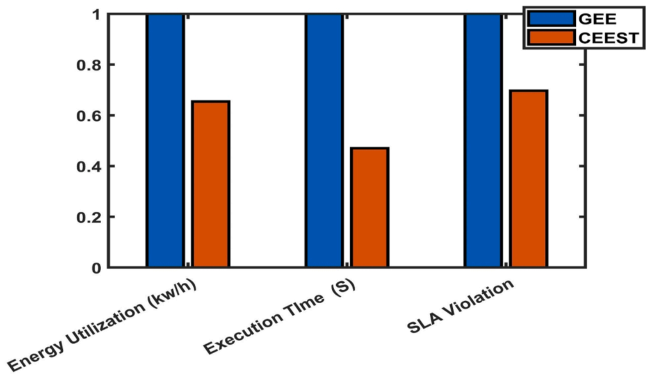

Fig. 15. Results of PlanetLab workflow on scale of 0 to 1.

Conclusion

In this paper, an energy-efficient algorithm is proposed which emphasizes task completion in pre-defined deadlines to reduce SLA violations. In the proposed algorithm (CEEST) virtual machines and hosts are chosen based on their weights and capacity to complete the job in the given deadline. This algorithm also vertically up scales servers immediately if the workload turns out to be higher. Directed acyclic graphs and real-world workloads are used to evaluate the proposed algorithm. The proposed algorithm saves energy up to 30% in comparison to existing algorithms. The utilization of resources is also significantly increased by to 30%. In terms of SLA violations, the proposed algorithm reduced SLA violations up to 50%. In future, workload prediction using machine learning is considered as potential research work. Using learning algorithms, we will predict workload demands. This will help in the efficient management of resources.

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.

Acknowledgements

All persons who have made substantial contributions to the work reported in the manuscript (e.g., technical help, writing and editing assistance, general support), but who do not meet the criteria for authorship, are named in the Acknowledgements and have given us their written permission to be named. If we have not included an Acknowledgements, then that indicates that we have not received substantial contributions from non-authors.

References

[1] Kaur K, Garg S, Aujla GS, Kumar N, Rodrigues JJPC, Guizani M. Edge computing in the industrial internet of things environment: Software-defined-networks-based edge-cloud interplay. IEEE Commun Mag 2018;56(2):44–51.

[2] Sookhak M, Yu FR, He Y, Talebian H, Sohrabi Safa N, Zhao N, et al. Fog vehicular computing: Augmentation of fog computing using vehicular cloud computing. IEEE Veh Technol Mag 2017;12(3):55–64.

[3] Hashem IAT, Yaqoob I, Anuar NB, Mokhtar S, Gani A, Ullah KS. The rise of "big data" on cloud computing: Review and open research issues. Inf Syst 2015;47:98–115. https://doi.org/10.1016/j.is.2014.07.006.

[4] Xu M, Buyya R. Managing renewable energy and carbon footprint in multi-cloud computing environments. J Parallel Distrib Comput 2020;135:191–202.

[5] Talebian H, Gani A, Sookhak M, Abdelatif AA, Yousafzai A, Vasilakos A V., et al. Optimizing virtual machine placement in IaaS data centers: taxonomy, review and open issues. vol. 0123456789. Springer US; 2019. https://doi.org/10.1007/s10586-019-02954-w.

[6] Greenberg A, Hamilton J, Maltz DA, Patel P. The cost of a cloud: research problems in data center networks. ACM New York, NY, USA; 2008. https://doi.org/10.1145/1496091.1496103.

[7] Dayarathna M, Wen Y, Fan R. Data center energy consumption modeling: A survey. IEEE Commun Surv Tutorials 2016;18(1):732–94. https://doi.org/10.1109/COMST.2015.2481183.

[8] Van Heddeghem W, Lambert S, Lannoo B, Colle D, Pickavet M, Demeester P. Trends in worldwide ICT electricity consumption from 2007 to 2012. Comput Commun 2014;50:64–76. https://doi.org/10.1016/j.comcom.2014.02.008.

[9] Amin R, Kumar N, Biswas GP, Iqbal R, Chang V. A light weight authentication protocol for IoT-enabled devices in distributed Cloud Computing environment. Futur Gener Comput Syst 2018;78:1005–19.

[10] Beloglazov A, Buyya R, Lee YC, Zomaya A. A Taxonomy and Survey of Energy-Efficient Data Centers and Cloud Computing Systems. vol. 82. 2011. https://doi.org/10.1016/B978-0-12-385512-1.00003-7.

[11] Beloglazov A, Buyya R. Adaptive threshold-based approach for energy-efficient consolidation of virtual machines in cloud data centers. Proc 8th Int Work Middlew Grids, Clouds e-Science, MGC 2010 - Held ACM/IFIP/USENIX 11th Int Middlew Conf 2010. https://doi.org/10.1145/1890799.1890803.

[12] Shaw R, Howley E, Barrett E. An energy efficient anti-correlated virtual machine placement algorithm using resource usage predictions. Simul Model Pract Theory 2019;93:322–42.

[13] Toor A, Islam Su, Sohail N, Akhunzada A, Boudjadar J, Khattak HA, et al. Energy and performance aware fog computing: A case of DVFS and green renewable energy. Futur Gener Comput Syst 2019;101:1112–21.

[14] Tang Z, Qi L, Cheng Z, Li K, Khan SU, Li K. An energy-efficient task scheduling algorithm in DVFS-enabled cloud environment. J Grid Comput 2016;14(1):55–74. https://doi.org/10.1007/s10723-015-9334-y.

[15] Ibrahim AAZA, Kliazovich D, Bouvry P. Service Level Agreement Assurance between Cloud Services Providers and Cloud Customers. Proc - 2016 16th IEEE/ACM Int Symp Clust Cloud, Grid Comput CCGrid 2016 2016:588–91. https://doi.org/10.1109/CCGrid.2016.56.

[16] Gandhi A, Chen Y, Gmach D, Arlitt M, Marwah M. Hybrid resource provisioning for minimizing data center SLA violations and power consumption. Sustain Comput Informatics Syst 2012;2(2):91–104. https://doi.org/10.1016/j.suscom.2012.01.005.

[17] Turowski M, Lenk A. Vertical scaling capability of OpenStack. Serv. Comput. 2014 Work, Springer; 2015, p. 351–62.

[18] Arunarani AR, Manjula D, Sugumaran V. Task scheduling techniques in cloud computing: A literature survey. Futur Gener Comput Syst 2019;91:407–15. https://doi.org/10.1016/j.future.2018.09.014.

[19] Gholami H, Zakerian R. A List-based Heuristic Algorithm for Static Task Scheduling in Heterogeneous Distributed Computing Systems. 2020 6th Int. Conf. Web Res., IEEE; 2020, p. 21–6.

[20] Iwendi C, Maddikunta PKR, Gadekallu TR, Lakshmanna K, Bashir AK, Piran MJ. A metaheuristic optimization approach for energy efficiency in the IoT networks. Softw Pract Exp 2020.

[21] Thomas Antony, Krishnalal G, Jagathy Raj VP. Credit based scheduling algorithm in cloud computing environment. Procedia Comput Sci 2015;46:913–20.

[22] Mao Y, Chen X, Li X. Max–min task scheduling algorithm for load balance in cloud computing. Proc. Int. Conf. Comput. Sci. Inf. Technol., Springer; 2014, p. 457–65.

[23] Mehdi NA, Mamat A, Amer A, Abdul-Mehdi ZT. Minimum completion time for power-aware scheduling in cloud computing. Proc 4th Int Conf Dev ESystems Eng DeSE 2011;2011:484–9. https://doi.org/10.1109/DeSE.2011.30.

[24] Madni Syed Hamid Hussain, Abd Latiff Muhammad Shafie, Abdullahi Mohammed, Abdulhamid Shafi'i Muhammad, Usman Mohammed Joda, Choo Kim-Kwang Raymond. Performance comparison of heuristic algorithms for task scheduling in IaaS cloud computing environment. PLoS One 2017;12(5):e0176321. https://doi.org/10.1371/journal.pone.0176321.

[25] Topcuoglu H, Hariri S, Wu MY. Performance-effective and low-complexity task scheduling for heterogeneous computing. IEEE Trans Parallel Distrib Syst 2002;13:260–74. https://doi.org/10.1109/71.993206.

[26] Arif Muhammad Shahzad, Iqbal Zeshan, Tariq Rehan, Aadil Farhan, Awais Muhammad. Parental prioritization-based task scheduling in heterogeneous systems. Arab J Sci Eng 2019;44(4):3943–52. https://doi.org/10.1007/s13369-018-03698-2.

[27] Geng X, Yu L, Bao J, Fu G. A task scheduling algorithm based on priority list and task duplication in cloud computing environment. Web Intell., vol. 17, IOS Press; 2019, p. 121–9.

[28] Mirjalili S. Genetic algorithm. Evol. algorithms neural networks, Springer; 2019, p. 43–55.

[29] Dorigo M, Stützle T. Ant colony optimization: overview and recent advances. Handb. metaheuristics, Springer; 2019, p. 311–51.

[30] Singh Gurpreet, Kumar Neeraj, Verma Anil Kumar. Antalg: An innovative aco based routing algorithm for manets. J Netw Comput Appl 2014;45:151–67.

[31] Bansal JC. Particle swarm optimization. Evol. swarm Intell. algorithms, Springer; 2019, p. 11–23.

[32] Kruekaew B, Kimpan W. Enhancing of artificial bee colony algorithm for virtual machine scheduling and load balancing problem in cloud computing. Int J Comput Intell Syst 2020;13:496–510.

[33] Jana B, Chakraborty M, Mandal T. A task scheduling technique based on particle swarm optimization algorithm in cloud environment. Soft Comput. Theor. Appl., Springer; 2019, p. 525–36.

[34] Duan K, Fong S, Siu SWI, Song W, Guan SSU. Adaptive incremental Genetic Algorithm for task scheduling in cloud environments. Symmetry (Basel) 2018;10:1–13. https://doi.org/10.3390/sym10050168.

[35] Reddy VD, Gangadharan GR, Rao G, Aiello M. Energy-Efficient Resource Allocation in Data Centers Using a Hybrid Evolutionary Algorithm. Mach. Learn. Intell. Decis. Sci., Springer; 2020, p. 71–92.

[36] Huang Q, Su S, Li J, Xu P, Shuang K, Huang X. Enhanced energy-efficient scheduling for parallel applications in cloud. Proc 12th IEEE/ACM Int Symp Clust Cloud Grid Comput CCGrid 2012;2012:781–6. https://doi.org/10.1109/CCGrid.2012.49.

[37] Beloglazov Anton, Abawajy Jemal, Buyya Rajkumar. Energy-aware resource allocation heuristics for efficient management of data centers for Cloud computing. Futur Gener Comput Syst 2012;28(5):755–68. https://doi.org/10.1016/j.future.2011.04.017.

[38] Mekala Mahammad Shareef, Viswanathan P. Energy-efficient virtual machine selection based on resource ranking and utilization factor approach in cloud computing for IoT. Comput Electr Eng 2019;73:227–44.

[39] Wu CM, Chang RS, Chan HY. A green energy-efficient scheduling algorithm using the DVFS technique for cloud datacenters. Futur Gener Comput Syst 2014;37:141–7. https://doi.org/10.1016/j.future.2013.06.009.

[40] Ding Y, Qin X, Liu L, Wang T. Energy efficient scheduling of virtual machines in cloud with deadline constraint. Futur Gener Comput Syst 2015;50:62–74. https://doi.org/10.1016/j.future.2015.02.001.

[41] Safari M, Khorsand R. Energy-aware scheduling algorithm for time-constrained workflow tasks in DVFS-enabled cloud environment. Simul Model Pract Theory 2018;87:311–26. https://doi.org/10.1016/j.simpat.2018.07.006.

[42] Hassan Hadeer A, Salem Sameh A, Saad Elsayed M. A smart energy and reliability aware scheduling algorithm for workflow execution in DVFS-enabled cloud environment. Futur Gener Comput Syst 2020;112:431–48.

[43] Filho MCS, Oliveira RL, Monteiro CC, Inácio PRM, Freire MM. CloudSim Plus: A cloud computing simulation framework pursuing software engineering principles for improved modularity, extensibility and correctness. Proc IM 2017 2017 IFIP/IEEE Int Symp Integr Netw Serv Manag 2017:400–6. https://doi.org/10.23919/INM.2017.7987304.

[44] Calheiros RN, Ranjan R, Beloglazov A, De Rose CAF, Buyya R. CloudSim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. Softw Pract Exp 2011;41:23–50.

[45] Spring Neil, Peterson Larry, Bavier Andy, Pai Vivek. Using PlanetLab for network research: Myths, realities, and best practices. ACM SIGOPS Oper Syst Rev 2006;40(1):17–24.