ESPS: Energy Saving Power Spectrum-Aware Scheduling to Leverage Differences in Power Ratings of Physical Hosts in Datacenters

Mahendra Kumar Gourisaria
School of Computer Engineering, KIIT Deemed to be University, Bhubaneswar - 751024 Odisha, India
E-mail: mkgourisaria2010@gmail.com

Pabitra Mohan Khilar
Department of Computer Science and Technology, National Institute of Technology, Rourkela - 769008, India
E-mail: pmkhilar@nitrkl.ac.in

Sudhansu Shekhar Patra
School of Computer Applications, KIIT Deemed to be University, Bhubaneswar - 751024, Odisha, India
E-mail: sudhanshupatra@gmail.com

Keywords: cloud computing, task scheduling, energy consumption, virtual machine, resource allocation

Received: February 25, 2021

Cloud Computing has seen massive growth over the past couple of decades, leading to exponential growth in energy consumption at data centres. Data centres consuming high amounts of energy leave a carbon footprint of the same scale, hence Cloud Service Providers (CSPs) have been looking for energy-efficient solutions to task scheduling in cloud to reduce the amount of carbon dioxide emission. Saving energy not only helps reduce the carbon footprint datacentres have on the environment, but also helps cover the costs of running multiple datacentres on the CSP's end. In this paper, we propose an energy saving task scheduling heuristic for heterogeneous cloud systems which selects the optimal physical host containing virtual machines with the additional consideration of the utilization of any incoming task on that particular virtual machine. We compare the energy efficiency of our proposed heuristic with recent algorithms including ECTC, MaxUtil, Random, and FCFS on several benchmark and synthetic datasets to display its superiority in energy-efficient task scheduling in heterogeneous cloud environments. FCFS, MaxUtil, Random, and ECTC respectively consume approximately 38.65%, 33.59%, 53.02%. and 46.96% more energy in a heterogeneous cloud environment as compared to our proposed heuristic namely Energy Saving Power Spectrum-Aware Scheduling (ESPS).

Povzetek: Računalništvo v oblaku beleži močno rast, kar vodi do eksponentne rasti porabe energije v podatkovnih centrih. V tem prispevku je predlagano hevristično načrtovanje naloge varčevanja z energijo za heterogene sisteme v oblaku, ki izbere optimalnega fizičnega gostitelja, ki vsebuje navidezne stroje. Energijska učinkovitost predlagane hevristike je primerjana z nedavnimi algoritmi, vključno z ECTC, MaxUtil, Random in FCFS na več primerjalnih in sintetičnih naborih podatkov.

1 Introduction

Cloud computing enables consumers around the globe have access to remote, shared computing resources [1, 30]. With the rapid increase in the capabilities of physical hosts housing a number of virtual machines (VMs) in terms of processing speed, storage capacity, cache memory, etc. the CSPs are able to meet the equally rising demand for these resources. Due to the increasing supply and demand of these resources, various power and energy related concerns are raised on behalf of the CSP. Moreover, with the increase in energy consumption of the datacentres, another environmental concern is raised with the massive amount of CO_2 emissions [38]. It is estimated that the hardware equipment of the IT sector is responsible for as much as 2% of the global CO_2 emissions [2] and the energy consumption that is coupled with this phenomenon is expected to double every year [3]. A data centre may take up as much energy as 25,000 households or a couple hundred office spaces. One of the main objectives of CSPs is to schedule the incoming tasks given by the user in such a way that it meets the required QoS parameters such as deadline, makespan, latency, packet loss, etc. adhering to the Service Level Agreement (SLA) made between the user and the CSP. Even security of user data is a prime area of research in cloud computing [31]. Most importantly, the CSP desires that the energy consumed by the cloud resources during this workflow is minimized.

Although task scheduling plays a vital role in cloud computing, bandwidth allocation and data storage are also an important pillar in the cloud environments [36, 37]. Task or job scheduling is a NP-complete problem [4] and various approaches have been proposed to minimize the energy costs pertaining to running the resources in a

Host Model	CPU	Clock	Cores	RAM	AAMP (watt)	APLP (watt)
Fujitsu Primergy RX1330 M1	Intel Xeon E3-1275 8MB L3 Cache	2.5 GHz	4	16 GB	13.8	63.7
Inspur NF5280M4	Intel Xeon E5-2699 v3 45 MB L3 Cache	2.3 GHz	18	64 GB	44.4	301
Dell PowerEdge R820	Intel Xeon E5-4650 v2 25 MB L3 Cache	2.4 GHz	40	4 GB	71.8	374
IBM NeXtScale nx360 M4	Intel Xeon E5-2660 v2 25 MB L3 Cache	2.2 GHz	20	24 GB	497	2414

Table 1: Operating power characteristics for different host models [5-8], where AAMP and APLP refer to Average Active Mode Power and Average Peak Load Power.

datacentre [28, 29]. To tackle this issue, we devise an algorithm namely Energy Saving Power Spectrum-Aware Task Scheduling (ESPS) which takes into consideration the utilization of incoming tasks along with the awareness of the different ranges of power spectra of the physical hosts in a datacentre. Typically, a datacentre comprises m number of physical hosts, and each physical host houses n VMs. Each VM in the same physical host shares the same operating power range which makes up for different sets of ranges in a datacentre over all VMs. For example, Table 1 illustrates the different power spectra for different physical hosts over the operation range in utilization from active mode (0%) to peak load capacity (100%). The differences of their power spectra is illustrated in Fig. 1.

From Table 1 it is easily realized that different physical host models have different power spectra of operation. ESPS exploits this fact by preferring to schedule tasks to the VMs in the physical hosts having the minimal power difference between the active mode and peak load power, given the fact that the task consumes relatively lower utilization in those VMs. This causes the power difference along with resource utilization in the energy model that we describe later to be minimized to optimize the energy function. The results of this procedure have been compared with algorithms like ECTC, *MaxUtil* [9], random [33] and First Come First Serve (FCFS) [10] on both benchmark and synthetic datasets. The main contribution of our work is as follows:

- Creation of a novel on-line mode task scheduler suited for heterogeneous cloud computing environment.
- Rigorous testing by simulation of proposed algorithm ESPS on several modified benchmark [23, 24] and synthetic datasets.
- Additional consideration of power differences in the operating characteristics of different physical hosts which, to the best of our knowledge, has not been exploited in order to augment energy conservation in resource allocation problems.
- Evaluation of proposed algorithm ESPS in terms of energy consumption.
- Simulation of existing algorithms ECTC, *MaxUtil*, Random and FCFS and juxtaposing their results with proposed algorithm ESPS to show reduction in energy consumption.

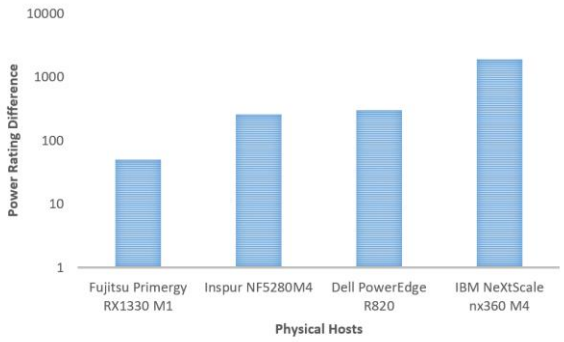

Figure 1: Power range differences of different physical hosts listed in Table 1 [34, 35]. According to the proposed algorithm, Fujitsu Primergy RX1330 M1 would be preferred more over other hosts for scheduling. The y-axis values are in logarithmic scale.

The rest of the paper is organized as follows. Section 2 describes other works relating to ours done before. Section 3 explains the cloud model used in the approach along with the energy model. Section 4 talks in detail about the proposed algorithm ESPS along with a hand-traced illustration of the model over a fixed dataset compared with other algorithms. Section 5 encapsulates the results of performance of all algorithms simulated over all the datasets. A discussion section has been added in Section 6 where our proposed work is compared with existing related work and finally, Section 7 holds the concluding remarks of our work.

2 Related work

In previous works, researchers have considered the server power efficiency to propose task scheduling heuristics for cloud environments. This means that different servers or physical hosts have different ranges of active mode power and peak load power. In this context, Lin *et al.* (2017) [35] proposed a heuristic ECOTS that uses the fact that different hosts have different power ratings and also mentioned that the active mode power and peak load power can be obtained directly by measurement of the

CPU idle power and peak power respectively, and that all the different physical hosts or servers have their own power models. This difference may arise due to different hardware configurations from server to server [5-8].

Task scheduling in cloud recently has witnessed a healthy amount of research work done to minimize energy related costs. Hsu *et al.* (2011) [11] proposed an Energy-Aware Task Consolidation (ETC) algorithm defined in a multi-cloud architecture on homogeneous computing resources. The main focus in the work was to set a fixed threshold to each VM's utilization lesnivel at 70% and migrate the tasks that could not be accommodated based on the threshold to other clouds. Due to the different bandwidths of connections between different clouds, the cost of task migration varies with recipient and receiving cloud. The idea behind setting a threshold at 70% for maximum VM utilization was that the energy vs. utilization curve increases non-linearly after 70% and for every small change in utilization there is noticed a high change in energy. However, the problem with ETC is that it includes the high costs of migration of tasks which increases energy consumption. Lee *et al.* (2010) [9] proposed two energy-saving heuristics namely ECTC (Energy-Conscious Task Consolidation) and *MaxUtil* which had two different objective functions to maximize. Both the heuristics mainly aim to consolidate tasks on fewer virtual machines but differ in their own ways. ECTC prefers to schedule tasks such that they mostly run in parallel with other tasks in the VM throughout their lifetime of execution, while *MaxUtil* prefers to schedule tasks where they may result in a higher average utilization level of the VM. However, both *MaxUtil* and ECTC increase task concentration over a few VMs which leads to higher energy consumption due to utilization levels of >70%.

Meisner *et al.* (2009) [12] propose an approach *PowerNap* to tackle the problem of idle power consumption and the overhead incurred due to the in-out transition of the low-power nap state. While this can eliminate idle power consumption, it suffers from poor performance of maintenance tasks such as buffer flushing, memory zeroing, etc. Ismail *et al.* (2018) [13] suggest an algorithm for task scheduling which is more energy efficient (EATSVM) and incorporates the idea of the augment in completion times of tasks running in a VM if the number of tasks in the same VM increase. Based on this increase, EATSVM selects the most optimized energy function for scheduling of tasks. Wu *et al.* (2013) [15] proposed a task scheduling algorithm for energy-saving by leveraging DVFS (Dynamic Voltage Frequency Scaling) which allocates resources to jobs based on the job's requirement without sacrificing system performance, which is an issue noticed with systems that use DVFS technique [16-18]. Khan *et al.* (2014) [19] propose an energy-aware task scheduling algorithm that uses reinforcement learning cooperatively on WSNs (wireless sensor networks) based applications. Their technique uses a reward-based system as done in most reinforcement learning algorithms where the model tries to maximize the reward achieved by trading the performance of the application along with the required energy consumption. Wen *et al.* (2011) [20] propose a hierarchical scheduling algorithm which minimizes energy consumption of network devices and servers, but, in their algorithm the nodes with lower temperature are selected by the application for scheduling. More recently, Panda *et al.* (2018) [14] propose a bi-objective task scheduling heuristic for heterogeneous environment which tries to minimize both makespan and energy consumption. It achieves the optimization of both objectives by taking the estimated time to compute matrix for each task along with the total utilization matrix over all VMs and normalizes the sum of each entry for each VM and finds the minimum value to schedule the task to the VM corresponding to the index of the minimum value. Mishra *et al.* (2019) [21] survey the current energy efficient service allocation techniques in cloud systems and divide the various techniques based on a taxonomy. The techniques consider tasks of either real-time or non-real time. They present a generalized system architecture for service allocation to minimize energy.

Quan *et al.* (2012) [26] proposed an algorithm that amasses data and statistics of the CO_2 emission and energy consumption of different servers in a datacentre. After this step, it predicts the server with optimal (lowest) values of energy consumption and CO_2 emission and through the use of two optimization algorithms namely Power Usage Effectiveness (PUE) and Carbon Usage Effectiveness (CUE). Finally, it migrates all the heavily loaded VMs to the most optimal server. The algorithm also incorporates SLA for the selection of the best server. However, the issue with VM migration is that it involves heavy cost in terms of energy to migrate VMs. Based on this dilemma, Zhang *et al.* (2018) [25] argue that the dynamic consolidation of VMs to as few physical machines as possible incurs a heavy overhead due to the migration costs in terms of energy. Many VM migration-based scheduling strategies do not consider this overhead and hence they propose an energy saving heuristic to exploit the fact that many tasks have loose deadlines. Thus, their heuristic postpones the execution of such tasks without the need of waking up any other physical machines and thus minimize energy without VM migration

Kliazovich *et al.* (2010) [27] propose a task consolidation heuristic based on balancing the energy consumption in datacentres by scheduling jobs to servers based on their thermal profiles or the workload and communication potential. This approach finds the optimum trade-off point between the consolidation of jobs to a few servers and avoiding hotspots in the datacentre. As seen above, we infer that there has not been much work in the literature which consider task allocation strategies in environments with different hosts. Almost all the work done on task scheduling deals with a single host consisting of a number of VMs, while in our work we also consider a number of physical hosts with varying capabilities to be a part of the system.

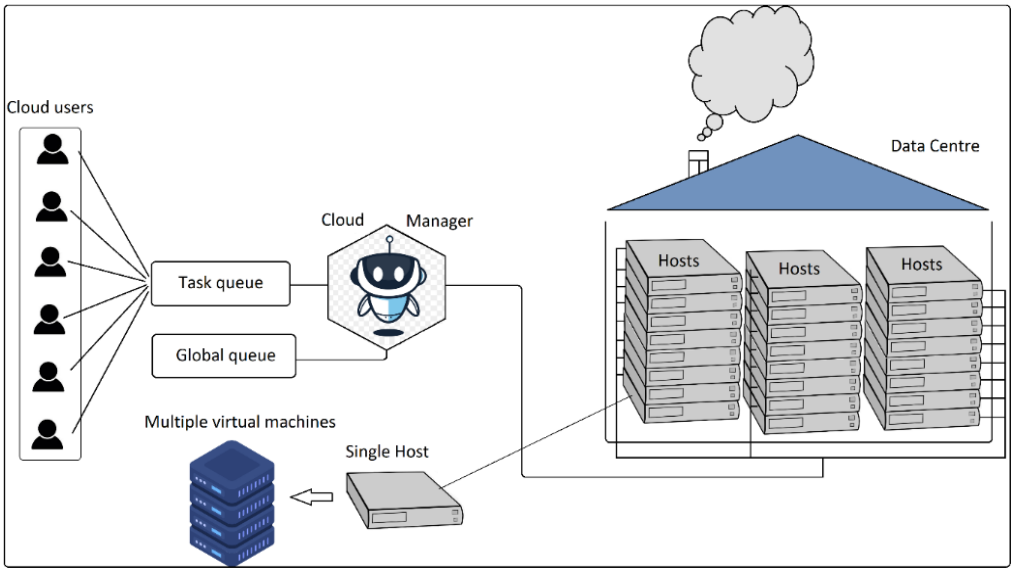

Figure 2: The proposed cloud model. Each host In a datacentre consists of multiple virtual machines as a result of virtualization.

3 System model

3.1 Cloud model

We assume the resources of the cloud model to be heterogeneous in the sense that each VM has a different processing speed, memory, etc. Consider there to be p physical hosts given by the 4-tuple, $H_p = \{ID, V_{ID}, P_{ID}^{min}, P_{ID}^{max}\}$ where ID is the identity of the host, V_{ID} refers to the set of VMs that the host with identity ID houses, , P_{ID}^{min} and P_{ID}^{max} are the minimum and maximum values of operating power respectively for the host. Each VM set V_{ID} has m VMs given by $V_{ID} = \{V_{ID}^1, V_{ID}^2, \dots, V_{ID}^m\}$. Now, each VM instance V_{ID}^j $(1 \leq j \leq m)$ has different processing speed, memory capacities, etc. as assumed earlier which means that each VM will behave differently for the same task. Fig. 2 demonstrates this cloud model. The cloud manager handles all requests that are submitted by users and schedules them according to the scheduling strategy of ESPS.

3.2 Task model

Consider there to be n tasks given by $T = \{T_1, T_2, \dots, T_n\}$ where each task T_i $(1 \leq i \leq n)$ is given by a 3-tuple as, $T_i = \{AT_i, ETC_i, TU_i\}$ where AT_i refers to the arrival time of task T_i, ETC_i is the estimated time to compute over all VMs for task T_i and TU_i is the total utilization matrix which shows the utilization consumed by the task T_i on all the different VMs. Note that the values of ETC and TU for each task will be different on each VM due to the heterogeneous nature of the cloud model. Both ETC and TU matrices are of the dimensions $n \times m$ which are demonstrated by Table 2.

T_i $(1 \leq i \leq n)$	VM_1	VM_2	…..	VM_m
1	56	23	…..	24
2	99	35	…..	73
…..	…..	…..	…..	…..
n	5	45	…..	35

Table 2: Illustration of the ETC or TU matrix for each task on each VM.

3.3 Energy model

Consider there to be n tasks and m virtual machines, then the utilization of a virtual machine j where $1 \leq j \leq m$ is given by the total sum of all the utilizations of the tasks running in virtual machine j. For now, we consider that all VMs from different hosts have been pooled in a universal VM set $V = \{VM_1, VM_2, \dots, VM_j, \dots, VM_m\}$. Mathematically, if we denote k as the time instant, we have,

$$UV_j^k = \sum_{i=1}^{n} TU_{ij} \times B_{i,j}^k \quad (1)$$

Where, $0 \leq UV_j^k \leq 100, 1 \leq TU_{i,j} \leq 100,$ and

$$B_{i,j}^k = \begin{cases} 1, if\ task\ T_i \rightarrow VM_j\ (is\ assigned\ to)\ at\ time\ k \\ 0, otherwise \end{cases} \quad (2)$$

$TU_{i,j}$ refers to the utilization taken by task T_i on virtual machine VM_j. As discussed before, the energy consumption of machine V_j at a given time k is given by [22],

$$E_j^k(UV_j^k) = \left(p_{max,j} - p_{min,j}\right) \times UV_j^k + p_{min,j} \quad (3)$$

This energy model is used in various other research works [9, 14]. The values $p_{max,j}$ and $p_{min,j}$ refer to the j^{th} VM's peak load and active mode power consumption respectively. At any instant k the energy consumed by all the resources over the cloud system is given by,

$$E^k(UV^k) = \sum_{j=1}^{m} E_j^k\left(UV_j^k\right) = \sum_{j=1}^{m}\left(p_{max,j} - p_{min,j}\right) \times UV_j^k + p_{min,j} \quad (4)$$

The energy of the system for the total makespan time period given by M can be expressed as,

$$E = \sum_{j=1}^{M} E^k(UV^k) = \sum_{k=1}^{M}\sum_{j=1}^{m}\left(p_{max,j} - p_{min,j}\right) \times UV_j^k + p_{min,j} \quad (5)$$

It is worthy to mention that VMs residing in the same host will share the same values of p_{max} and p_{min}. In this way, there are different sets of values of p_{max} and p_{min} which correspond to each physical host which parameterize each VM's energy properties.

4 Proposed approach

Our goal is to minimize the function E given by eqn. (5) through a mapping $f: T \to V$ where, we know as a preliminary that $\left(p_{max,j} - p_{min,j}\right) \times UV_j^k \gg p_{min,j}$ whenever the system (or physical host) is in non-idle condition. By non-idle condition, we assume $UV_j^k > 0.2$ (20%). Hence, to fulfil our objective to optimize E, ESPS employs two approaches in the algorithm –

i) Minimizing E as

$$\min(E) = \min\left(\sum_{k=1}^{M}\sum_{j=1}^{m}\left(p_{max,j} - p_{min,j}\right) \times UV_j^k + p_{min,j}\right) \quad (6)$$

This is possible by minimizing the term $\left(p_{max,j} - p_{min,j}\right)$ such that the product $\left(p_{max,j} - p_{min,j}\right) \times UV_j^k$ is minimized. Our algorithm chooses the physical host that has the least difference of maximum and minimum power consumptions and prefers to schedule tasks more in such hosts.

ii) Minimizing E as,

$$argmin_{UV_j^k}(E) = argmin_{UV_j^k}\left(\sum_{k=1}^{M}\sum_{j=1}^{m}\left(p_{max,j} - p_{min,j}\right) \times UV_j^k + p_{min,j}\right) \quad (7)$$

Here, the utilization of VM j at time k can be minimized by preferring to schedule tasks to such VMs where they consume lower CPU utilization.

4.1 Scheduling technique

ESPS follows the following scheduling strategy according to the proposed approach:

i) For any incoming task T_i $(1 \le i \le n)$ we have the utilization matrix TU which conveys the utilization consumed by T_i on all the different virtual machines. We proceed by normalizing all entries of TU_{ij} for T_i over all virtual machines VM_j $(1 \le j \le m)$,

$$nTU_{ij} = \frac{TU_{ij}}{\max(TU_i)} \quad (8)$$

Here, max (UT_i) refers to the maximum value of the entire row of the TU matrix for T_i. Similarly, for each physical host we assume that the values of their p_{max} and p_{min} are different. For example, let there be 2 different hosts H_1 and H_2 containing 3 VMs each such that $H_1 \leftarrow \{VM_1, VM_2, VM_3\}$ and $H_2 \leftarrow \{VM_4, VM_5, VM_6\}$. Then, we assume $p_{max}^1 \leftarrow p, p_{min}^1 \leftarrow q, p_{max}^2 \leftarrow r$ and $p_{min}^2 \leftarrow s$ where p^1 and p^2 are the power consumed by hosts H_1 and H_2 respectively.

ii) We create a new matrix A which stores the differences of each individual VM's values of p_{max} and p_{min} as,

$$A = \{(p_{max}^1 - p_{min}^1), (p_{max}^1 - p_{min}^1), (p_{max}^1 - p_{min}^1), (p_{max}^2 - p_{min}^2), (p_{max}^2 - p_{min}^2), (p_{max}^2 - p_{min}^2)\} \quad (9)$$

Or,

$$A = \{(p-q), (p-q), (p-q), (r-s), (r-s), (r-s)\} \quad (10)$$

The values of peak and active mode power consumption for each VM is taken by the underlying assumption that every VM in the same host has the same set of values of p_{max} and p_{min}. For each entry A_j in matrix A we normalize as follows,

$$nA_j = \frac{A_j}{\max(A)} \quad (11)$$

iii) To make a scheduling decision, we sum the values in the matrices nUT and nA and find the minimum element, the index of which is the selected VM for mapping,

$$\mathrm{V} = \mathrm{index}(\min(nTU + nA)) \quad (12)$$

Where index(min $(nUT + nA)$ returns the index of the minimum summed pair to which the task T_i is to be scheduled. In this manner, the objective given by $argmin_E(f: T \to V)$ is satisfied. Algorithm 1 contains the algorithm of ESPS and all the symbols used in it are described in Table 3.

Remark 1 The total normalized values given by $(nTU + nA) \in [0, 2]$.

Symbol	Meaning
T_i	i^{th} task
VM_j	j^{th} virtual machine
k	k^{th} time unit
UV_j^k	Utilization of j^{th} VM at time instant k
m	Total number of VMs
GQ	Global queue meant for tasks that are waiting
$P_{max}(VM_j)$	Maximum power at peak load capacity of j^{th} VM
$P_{min}(VM_j)$	Minimum power at active mode of j^{th} VM
Q_k	Queue of all tasks arriving at time instant k
TU_{ij}	Utilization of i^{th} task on j^{th} VM

Table 3: Symbols used in Algorithm 1.

Input: $T_{(1\sim n)}, VM_{(1\sim m)}, TU_{(1\sim n)(1\sim m)}$	**Output:** Minimized E
1. **set** $GQ \leftarrow \{\}$ 2. **set** $P_{diff} \leftarrow \{\}$ 3. **for** j = 1, 2, …, m **do** 4. **add** $\left(P_{max}(VM_j) - P_{min}(VM_j)\right)$ to P_{diff} 5. **end for** 6. **for each** $T_i \in Q_k, GQ$ **do** 7. **set** $v \leftarrow$ Call GET-OPTIMAL-VM(T_i) 8. **if** $v ==$ -1 **then** 9. **add** T_i to GQ 10. **end if** 11. **else do** 12. **assign** $T_i \to v$ 13. **end else** 14. **end for**	

Algorithm I: Energy Saving Power Spectrum-Aware Task Scheduling (ESPS).

The worst time complexity of the proposed algorithm of ESPS is $O(m + nm)$. Having n tasks and m VMs or resources, we see the lines 3-4 require $O(m)$ in Algorithm I. Then, the loop defined in line 6 requires $O(n)$ which calls Procedure I, whose lines 1-2 in turn call Procedure II. The lines 2-3 in Procedure II require $O(m)$ while in Procedure I the lines 4-5, 9-17 require the same $O(m)$.Hence, the overall time complexity of ESPS is $O(m + nm)$.

4.2 An illustration

We shall describe a dry-run based on 6 tasks as shown by Table 4. The 3 VMs shown in Table 4 are distributed

Input: T_i, P_{diff}	**Output:** Optimal VM_j for scheduling
1. **set** $P_{norm} \leftarrow$ Call GET-NORMALIZED-VECTOR(P_{diff}) 2. **set** $(TU_i)_{norm} \leftarrow$ Call GET-NORMALIZED-VECTOR(TU_i) 3. **set** $\sigma \leftarrow \{\}$ 4. **for** j = 1, 2, ..., m **do** 5. **add** $(P_{norm}[j] + (TU_i)_{norm}[j])$ to σ 6. **end for** 7. **set** $\sigma_{min} \leftarrow index(\min(\sigma)) + 1$ 8. **set** flag ← 0 9. **while** True **do** 10. **if** $\sigma_{min} == index(\max(\sigma)) + 1$ **then** 11. **set** flag ← 1 12. **break** 13. **end if** 14. **if** $UV_{\sigma_{min}}^k + TU_{i,\sigma_{min}} \leq 100$ **then** 15. **break** 16. **end if** 17. **set** $\sigma_{min} \leftarrow$ next $index(\min(\sigma)) + 1$ 18. **end while** 19. **if** flag == 1 **then** 20. **return** -1 21. **end if** 22. **return** $VM_{\sigma_{min}}$	

Procedure I: GET-OPTIMAL-VM(T_i)

Input: A	**Output:** A_{norm}
1. **set** $A_{norm} \leftarrow \{\}$ 2. **for** j = 1, 2, …, m **do** 3. **add** $(A[j]/\max(A))$ to A_{norm} 4. **return** A_{norm}	

Procedure II: GET-NORMALIZED-VECTOR(A)

separately in 3 different hosts and the power rating of these hosts is given by Table 5.

We can see from Table 5 that different hosts have different values of p_{max} and p_{min}, an aspect that the proposed algorithm ESPS exploits in order to minimize the difference of these two variables as seen in the function E. For instance, we see from Table 4 that T_0 has a very low resource utilization at VM_1 but VM_1 resides in Host 1 which has a higher power difference as seen by Table 5. Our algorithm tries to optimize this trade-off between utilization and power differences of hosts to capture the best scheduling decision.

Initially, T_0 arrives at 1 time unit. Its utilizations on all resources are 25%, 48% and 57% whose host power rating differences are (50-20), (20-10) and (70-30) respectively. The heuristic normalizes these two sequences and finds the minimum sum value, which is 1.09 (i.e. $\frac{48}{57} + \frac{10}{40}$)

		ETC			TU		
Tasks	**AT**	**VM 1**	**VM 2**	**VM 3**	**VM 1**	**VM 2**	**VM 3**
0	1	10	4	14	25	48	57
1	3	10	6	12	44	48	17
2	5	11	9	12	76	40	19
3	6	8	10	7	36	57	71
4	8	8	13	10	43	56	99
5	9	6	7	6	38	92	45

Table 4: Task table containing 6 tasks with their respective arrival time and ETC and TU matrices.

Algorithm	FCFS	*MaxUtil*	ECTC	Random	**ESPS**
Energy	59560	71740	77640	110840	**38840**

Table 6: Energy consumption (watts time) of all algorithms based on the task list given by Table 4.

Host	p_{min}	p_{max}
Host 1 (VM 1)	20	50
Host 2 (VM 2)	10	20
Host 3 (VM 3)	30	70

Table 5: Power spectra of different physical hosts used in the illustration based on active mode and peak load power.

on VM_2. Similarly, when T_1 arrives in the system, the minimum sum of normalized values is 1.25 (i.e. $\frac{48}{48}+\frac{10}{40}$) again, on VM_2. We can already see how the heuristic prefers to schedule tasks with relatively lower utilization level to VM_2 as it resides in the host having the least power difference. The rest of the scheduling of ESPS is shown by Fig. 3 (a), where each VM's utilization level per time is shown along with the tasks assigned to that VM in the particular time instant. The tasks end when they appear towards the left of the time units hence reducing the assigned VM's utilization level at that time instant. The tilde (~) mark indicates range of values, and GQ indicates the tasks that are placed in global queue at that particular time instant (row). Similarly, the same set of tasks are simulated using Random, MaxUtil, ECTC and FCFS shown by Fig. 3 (b), Fig. 4 (b), (c) and (a) respectively. We present the energy consumption on the list of 6 task taken for each algorithm in Table 6.

It is clear from Table 6 that our proposed heuristic ESPS performs better on the given list of tasks. More specifically, it saves 34.78%, 45.86%, 49.97%, and 64.95% more energy as compared to algorithms FCFS, MaxUtil, ECTC and Random respectively. Clearly, we can see the efficacy of ESPS when there are different hosts with varying power rating differences.

5 Performance evaluation

This section encapsulates all the results done through simulations of ESPS and other algorithms (ECTC, FCFS, *MaxUtil*, and Random) on several benchmark and synthetic datasets to show the efficacy of ESPS in terms of energy consumption.

5.1 Description of datasets and simulations

We have used two benchmark datasets, considering different configurations of datasets generated by Ali *et al.* (2000) [24] and Braun *et al.* (2001) [23]. The advantage of using these two benchmark datasets is that they offer a wide variety of tasks in terms of task heterogeneity and also offer estimated time to compute values by varying machine heterogeneity. These values are obtained from real world applications and are a good choice to use for testing the performance of a task scheduling heuristic in a heterogeneous cloud system. The datasets are characterized by their names given by the general form: $u\_a\_bbcc, A.u\_a\_bbcc, B.u\_a\_bbcc$ where u stands for uniform distribution, a takes the values c, i and s which refer to consistent, inconsistent and semi-consistent ETC matrices respectively, bb indicates task heterogeneity which takes values hi and lo which refer to high or low task heterogeneity respectively, and finally cc indicates machine heterogeneity which also takes values hi and lo referring to high or low machine heterogeneity respectively. A and B refer to dataset of Ali *al.* (1024×32; 1024 tasks and 32 machines) and dataset of Braun *et al.* (1024×32; 1024 tasks and 32 machines). The benchmark datasets of the form u_a_bbcc is also one of Braun *et al.'s* generated datasets but has 512 tasks to be scheduled to 16 machines (512×16). In total, the number of benchmark datasets that we perform simulations on is 36, each having different ETC instances. To pre-process the data, we divide the values by a) 100 for datasets of the form u_a_bbcc and $B.u\_a\_bbcc$, and b) 1000 for datasets of form $A.u\_a\_bbcc$. Finally we round the values to the nearest integer value for simplicity. Each dataset is coupled with its own total utilization matrix TU which is generated for $n \times m$ numbers (where n refers to number of tasks and m refers to number of machines) using a random function which fall in the range [1, 100], both limits inclusive. For each benchmark dataset we use a) 4 hosts housing 4 VMs each for the 512×16 Braun *et al.* dataset, and b) 4 hosts housing 8 VMs each for the 1024×32 Braun *et al.* and Ali *et al.* datasets. The power rating specifications of these hosts is given by Table 13.

In the case of synthetic datasets, we take 5 instances comprising 100, 500, 1000, 5000 and 10000 tasks mapped on to 10, 20, 30, 40 and 50 resources, respectively. The ETC and TU matrices generated for each dataset are done randomly by restricting the values (inclusive) in the range [1, 40]. Table 7 describes the details of the cloud model used for the synthetic dataset simulations. Table 8

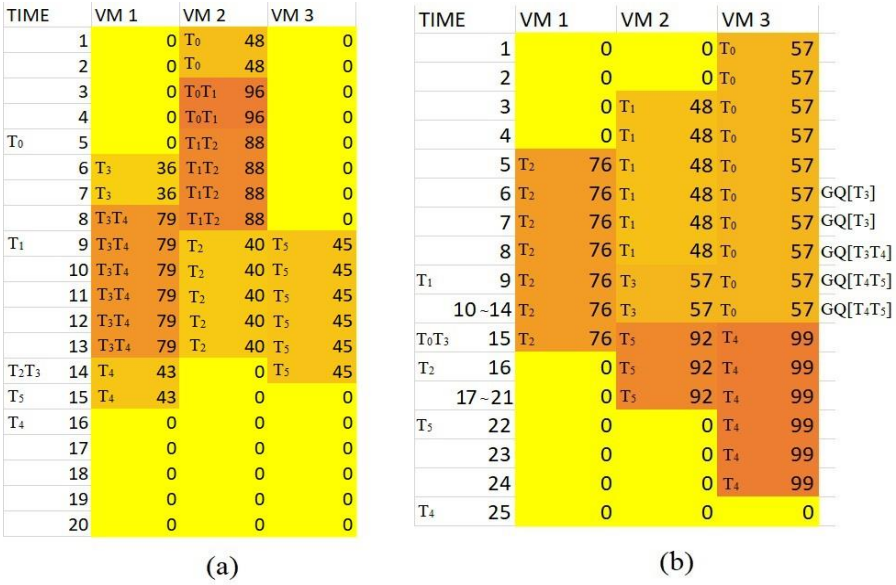

Figure 3: (a) Scheduling of the list of 6 tasks on ESPS, and (b) scheduling on random algorithm.

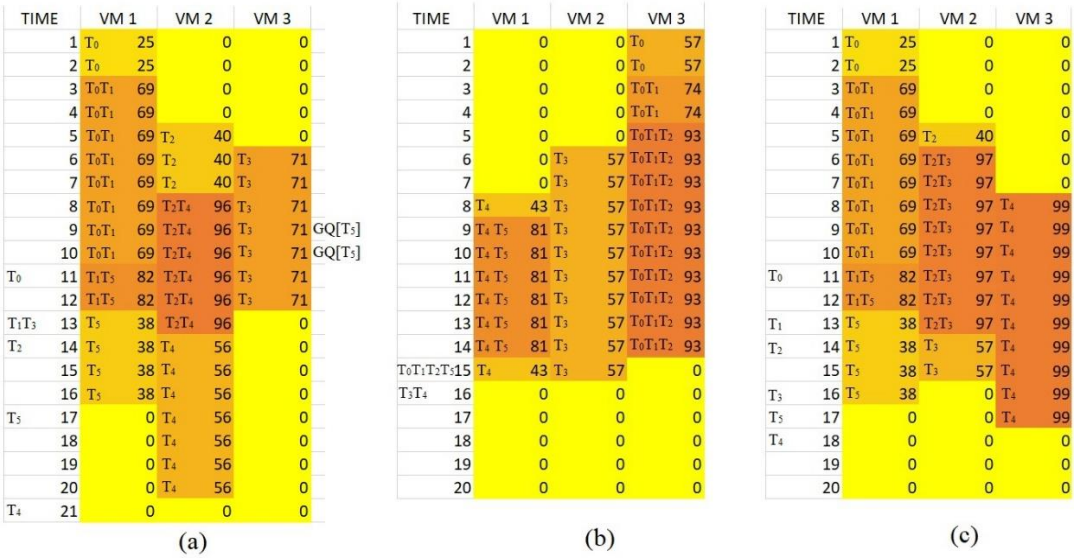

Figure 4: (a) Scheduling of the 6 tasks on by FCFS, (b) scheduling using MaxUtil, and (c) scheduling on ECTC.

Synthetic Dataset	# Tasks	# VMs	# Hosts	#VMs in each host
1	100	10	2	5
2	500	20	4	5
3	1000	30	5	6
4	5000	40	5	8
5	10000	50	5	10

Table 7: Cloud model specifications used for synthetic dataset simulations.

specifies the power ratings of the different hosts used for the synthetic dataset. All the experiments were performed using Python 3.5 on an Intel ® Core ™ i5-6200U CPU @ 2.30 GHz and 8 GB RAM on Windows 10 Pro 64-bit, x64 based processor.

5.2 Simulation results

We compare our proposed algorithm ESPS with other existing algorithms namely FCFS, *MaxUtil*, ECTC and random. The simulations were done on 36 instances of benchmark datasets and 5 instances of synthetic self-created datasets. We notice that ESPS performs better in terms of energy consumption in all instances of the datasets, benchmark or synthetic. The main reason for this is that ESPS combines two objectives, namely least utilization and least power difference to optimize the energy function E given by eqn. (5). For the benchmark datasets, we denote the percentage increase in power consumption of heuristics FCFS, *MaxUtil*, ECTC and

Dataset	Host 1		Host 2		Host 3		Host 4		Host 5	
	P_{min}	P_{max}	P_{min}	P_{max}	P_{min}	P_{max}	P_{min}	P_{max}	P_{min}	P_{max}
100×10	20	50	30	40	-	-	-	-	-	-
500×20	20	50	30	60	40	50	30	40	-	-
1000×30	20	50	30	60	40	50	30	40	20	40
5000×40	20	50	30	60	40	50	30	40	20	40
10000×50	20	50	30	60	40	50	30	40	20	40

Table 8: Host power rating (watts) taken for synthetic dataset simulations.

Instances	FCFS	MaxUtil	Random	ECTC	ESPS
u_c_hihi	8.88E+08	8.82E+08	9.00E+08	9.67E+08	**7.62E+08**
u_c_hilo	1.74E+07	1.91E+07	3.99E+07	2.05E+07	**7.84E+06**
u_c_lohi	3.97E+07	4.79E+07	6.69E+07	6.65E+07	**2.53E+07**
u_c_lolo	1.37E+06	1.04E+06	2.75E+06	1.37E+06	**8.27E+05**
u_i_hihi	1.25E+09	1.17E+09	1.11E+09	1.26E+09	**1.02E+09**
u_i_hilo	2.96E+07	2.58E+07	4.49E+07	3.68E+07	**8.95E+06**
u_i_lohi	6.63E+07	6.06E+07	7.16E+07	7.68E+07	**2.81E+07**
u_i_lolo	2.26E+06	2.68E+06	2.63E+06	2.23E+06	**8.41E+05**
u_s_hihi	1.06E+09	1.03E+09	1.08E+09	1.06E+09	**8.35E+08**
u_s_hilo	2.19E+07	2.36E+07	4.23E+07	3.01E+07	**8.64E+06**
u_s_lohi	4.96E+07	4.48E+07	7.05E+07	6.70E+07	**2.51E+07**
u_s_lolo	1.55E+06	1.93E+06	2.75E+06	1.60E+06	**8.39E+05**

Table 9: Simulation results of all algorithms on the 512×16 Braun et al. dataset.

Instances	FCFS	MaxUtil	Random	ECTC	ESPS
u_c_hihi	1.31E+09	1.41E+09	1.40E+09	1.47E+09	**1.03E+09**
u_c_hilo	2.06E+07	1.68E+07	8.76E+07	1.52E+07	**1.06E+07**
u_c_lohi	3.30E+07	4.81E+07	1.87E+08	5.76E+07	**2.66E+07**
u_c_lolo	3.66E+06	3.27E+06	6.20E+06	3.66E+06	**2.54E+06**
u_i_hihi	1.93E+09	1.72E+09	1.80E+09	1.90E+09	**1.34E+09**
u_i_hilo	5.55E+07	4.78E+07	1.01E+08	7.21E+07	**1.15E+07**
u_i_lohi	1.25E+08	1.07E+08	2.12E+08	1.91E+08	**3.23E+07**
u_i_lolo	5.52E+06	5.50E+06	6.07E+06	5.48E+06	**2.55E+06**
u_s_hihi	1.84E+09	1.86E+09	1.72E+09	1.77E+09	**1.28E+09**
u_s_hilo	3.80E+07	3.93E+07	9.27E+07	5.03E+07	**1.16E+07**
u_s_lohi	1.01E+08	9.96E+07	2.05E+08	1.56E+08	**3.28E+07**
u_s_lolo	4.62E+06	3.90E+06	5.80E+06	4.61E+06	**2.55E+06**

Table 10: Simulation results of all algorithms on the 1024×32 Braun et al. dataset.

Instances	FCFS	MaxUtil	Random	ECTC	ESPS
u_c_hihi	4.55E+08	4.52E+08	5.04E+08	4.80E+08	**3.55E+08**
u_c_hilo	3.33E+07	4.79E+07	1.86E+08	5.03E+07	**2.66E+07**
u_c_lohi	5.65E+06	5.06E+06	4.95E+06	4.65E+06	**2.36E+06**
u_c_lolo	3.65E+06	3.26E+06	3.65E+06	3.65E+06	**2.36E+06**
u_i_hihi	6.29E+08	5.98E+08	6.70E+08	6.79E+08	**4.64E+08**
u_i_hilo	1.32E+08	1.12E+08	2.10E+08	1.77E+08	**3.44E+07**
u_i_lohi	4.25E+06	4.96E+06	5.05E+06	4.75E+06	**2.36E+06**
u_i_lolo	3.85E+06	3.46E+06	3.85E+06	3.75E+06	**2.56E+06**
u_s_hihi	6.32E+08	5.69E+08	6.73E+08	6.23E+08	**4.52E+08**
u_s_hilo	1.00E+08	9.43E+07	2.06E+08	1.51E+08	**3.41E+07**
u_s_lohi	4.95E+06	4.06E+06	5.65E+06	4.55E+06	**2.36E+06**
u_s_lolo	3.65E+06	3.26E+06	3.65E+06	3.65E+06	**2.36E+06**

Table 11: Simulation results of all algorithms on the 1024×32 Ali et al. dataset.

Algorithm	100×10	500×20	1000×30	5000×40	10000×50
FCFS	9.44E+05	6.80E+06	1.48E+07	7.58E+07	1.62E+08
MaxUtil	7.18E+05	6.63E+06	1.36E+07	7.60E+07	1.61E+08
Random	9.35E+05	5.56E+06	1.12E+07	6.29E+07	1.34E+08
ECTC	1.01E+06	7.88E+06	1.66E+07	9.02E+07	1.89E+08
ESPS	2.37E+05	1.62E+06	4.03E+06	2.32E+07	5.47E+07

Table 12: Simulation results of all algorithms on the synthetic datasets.

Host	P_{min}	P_{max}
Host 1	20	50
Host 2	10	20
Host 3	30	70
Host 4	10	40

Table 13: Host power rating specifications in watts used for the simulation of all algorithms on benchmark datasets.

Benchmark	FCFS (%)	*MaxUtil* (%)	Random (%)	ECTC (%)
Braun *et al.* [23] 512×16	25.99	21.50	25.99	32.15
Braun *et al.* [23] 1032×32	44.76	41.90	53.96	50.47
Ali *et al* [24] 1032×32	45.21	37.39	79.13	58.26
Avg. increase	**38.65**	**33.59**	**53.02**	**46.96**

Table 14: Increase in power usage of other heuristics in comparison to proposed heuristic ESPS.

random in Table 14. These calculations were made by taking the average energy consumed of all the heuristics over the three benchmark datasets individually and comparing the percentage increase in energy consumption with our proposed heuristic.

The data for these percentages can be found in Table 9, Table 10 and Table 11. The same comparison has not been done for the synthetic datasets as it is evident from the values that ESPS outperforms other heuristics in terms of energy consumed. We have discussed further on synthetic dataset performance in Section 6. The results of simulations of all the algorithms for 512×16 (Braun *et al.*), 1024×32 (Braun *et al.*), and 1024×32 (Ali *et al.*) are shown in table form by Table 9, Table 10, Table 11, and Table 14 and graphically by Fig. 5, Fig. 6, and Fig. 7 respectively. The results of the synthetic dataset simulations are given by Table 12 and Fig. 8. Based on our simulations, we notice that FCFS, MaxUtil, Random and ECTC respectively consume approximately about 38.65%, 33.59%, 53.02%, and 46.96% more energy over all benchmark datasets when compared to our proposed approach. This behavior is seen throughout the variety of datasets that we use, i.e. also in case of synthetic dataset and dry run.

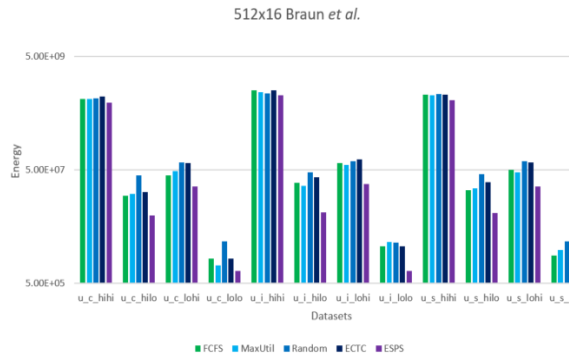

Figure 5: Energy consumption of each algorithm on the 512×16 Braun et al. dataset.

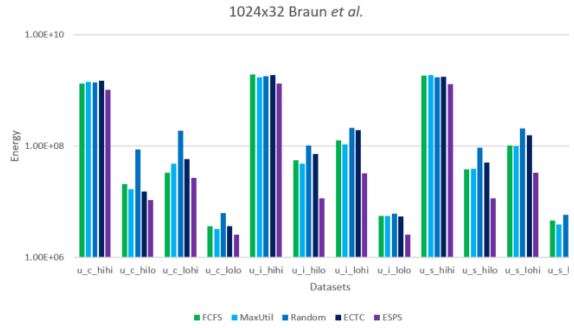

Figure 6: Energy consumption of all algorithms on the 1024×32 Braun et al. dataset.

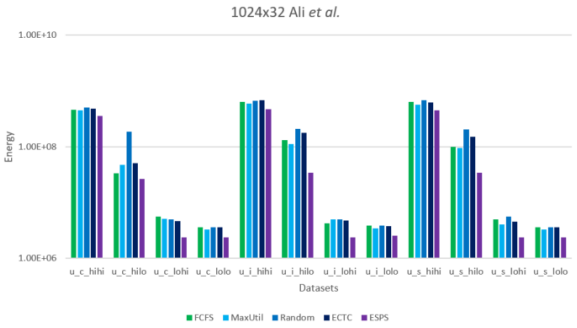

Figure 7: Energy consumption of all algorithms on the 1024×32 Ali et al. dataset.

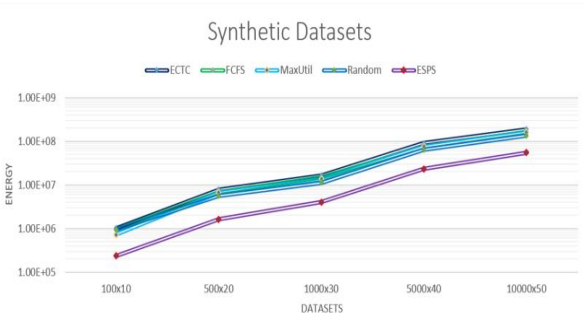

Figure 8: Energy consumption of all algorithms on the synthetic datasets.

6 Discussion

The main novelty of our proposed work is that it takes into account the power difference of different physical host in peak and active mode. Real life datacenters consist of different physical host which may have different power rating (as seen in Fig. 1). It is also clearly understood from Table 1 that in real life scenario, the physical host may have different power rating in peak and idle mode. Other implementations focuses mainly on energy where as our proposed work focuses on two important objectives namely least power difference and least utilization. First it minimizes energy by choosing an optimal physical host which has least difference of maximum and minimum power consumption in peak and active mode, respectively. Secondly, it minimizes energy through minimization of utilization by preferring to schedule tasks to such VMs where they consume lower CPU utilization. It is economical in nature when compared with other existing algorithm. Even in case of synthetic dataset, our proposed algorithm saves around 72.41%, 67.69%, 67.38% and 60.84% more energy when compared with ECTC, FCFS, MaxUtil and Random, respectively. We have also compared our approach with other related work in Table 15. Most of the related work uses either synthetic data set or benchmark dataset, however, we have used one synthetic dataset and two benchmark datasets. Additionally, we have dry run our algorithm on some sample values. Our proposed work outperform in all cases of the datasets.

7 Conclusion and future directions

We propose a new heuristic namely Energy Saving Power Spectrum-Aware Task Scheduling (ESPS) that keeps track of the least power difference in the operating power characteristics of different hosts in a cloud datacenter. ESPS has a time complexity of $O(m + nm)$ if we have n tasks to be mapped to m virtual machines. Along with this, it schedules task with a relatively lower utilization level onto those VMs that are hosted by machines having a lower power rating difference in a manner to balance between the utilization level and the power difference. Note that energy consumption of the FCFS, MaxUtil, Random, and ECTC algorithm are about 38.65%, 33.59%, 53.02% and 46.96% more than our proposed algorithm respectively. ESPS normalizes the values of the arrived task's utilizations on different VMs in the system and the power differences of the peak load capacity and active mode powers. Post normalization it picks the least sum of the two to make a scheduling decision. It is worthy to mention that the power differences taken in the simulation of benchmark and synthetic dataset were chosen by us and they may vary in the case of real life applications. This implies that our algorithm has the potential to outperform any other heuristic for task scheduling solely based on the power differences of the different hosts present at a data center.

As we just mentioned that the performance of our algorithm relies heavily on the power differences of different hosts, future work may extend the idea of exploiting this aspect and apply this technique to some

Ref.	Technique	Advantage / Result	Disadvantage/ Future work
[35]	Consider the different power rating for different host, power model and performance model are combined in the proposed work.	Saves energy in the range of 21% to 22% without violating the resource requirements of the cloud task.	The power model is not adaptive to a wide range of infrastructure. Server components are not taken into account.
[11]	Fixed threshold of 70% utilization of VM is proposed and the task above threshold level is migrated to other VM.	Significant saving in power consumption. 17% better than ECTC and MaxUtil.	Increase in overall cost due to migration of task between clusters.
[9]	Proposed two different heuristic ECTC and MaxUtil. First approach considers the actual energy consumption of ongoing task and in the second approach, average utilization is considered.	Both active and idle energy utilization is considered. Both the proposed algorithm outperform Random scheduling regardless of adoption of migration by 18% and 13%.	Energy consumption increases with utilization. The decision to schedule a new task is based upon the current state of task bindings.
[12]	Introduced a new approach called *PowerNap* where the system transitions between active and idle state. Also introduced Redundant Array for Inexpensive Load Sharing (RAILS).	Reduced average server power consumption by 74%.	It suffers from poor performance of maintenance tasks such as buffer flushing, memory zeroing, etc.
[13]	Assigned task to those VMs where the increase in energy consumption is the least. This approach considers both active and idle virtual machines.	Performs better than ECTC in terms of energy consumption by 14.06%.	The algorithm does not consider the execution of the individual tasks when they are not overlapping.
[15]	In this approach priority of job scheduling is considered. Weight and SLA level is also taken into account along with the DVFS approach to control the voltage supply.	Found to be efficient in reducing the energy consumption up to 5%-25% without sacrificing the system performance. Proves 23% better than ECS in terms of energy.	No limitations or drawback of the proposed work is listed in the paper.
[19]	The proposed work makes use of cooperative reinforcement learning in WSN-based network. They use a reward-based system where the model tries to maximize the reward achieved by trading the performance of the application along with the required energy consumption	Proves better than non-cooperative reinforcement learning approach.	Server energy limitations pose a particular challenge. Not compared with other variant of reinforcement learning methods. Non-consideration of a real world motion model for the targets and data association as a task.
[20]	Energy of both network devices and server is reduced by proposing a hierarchical scheduling algorithm but, in their algorithm the nodes with lower temperature are selected by the application for scheduling.	Minimizes the energy consumption of both server and network devices. Amount of data transfer also reduced.	Limited energy aware factor is considered.
[14]	The approach tries to minimize both makespan and energy consumption. Makes use of both ETC matrix and Task Utilization matrix.	Saves considerable amount of energy and makespan as compared to ECTC, ETC and MaxUtil etc.	Energy and execution cost are not considered in this paper.
[26]	Proposed an algorithm which moves the workload to the lowest energy consumption server and CO_2 emission server.	Save up to 10% to 31% in terms of energy and 10% to 87% in terms of carbon emission.	VM migration involves heavy cost in terms of energy migration.
[25]	Proposed EDA-NMS which takes care of task deadlines without inducing VM migration overhead	Good in terms of energy saving when compared with other algorithm. No compromise in deadlines is done.	The proposed work did not experiment on real-word trace data.
[27]	The algorithm is based on DENS approach which balances the energy consumption in data centers by scheduling job based on their thermal profile and communication potential.	Able to optimize the tradeoff between job scheduling the distribution of traffic pattern.	Did not test the DENS approach in realistic setup using testbuds.
[Our]	Our proposed work is mainly based on the power difference of different physical host. It schedules task to those VMs which have the lower power rating difference. ESPS considers the least sum of normalized values of the incoming task utilization on different VM and the power difference of peak load capacity and active mode of different hosts.	Our proposed work outperforms in dry run, two benchmark datasets and one synthetic dataset and saves hefty amount of energy in the range of 35% to 72% as compared with other heuristics. The main novelty of our algorithm is that it is also considers the power difference of different physical host.	Our proposed work only focuses on power, energy and utilization. Other parameters like SLA and makespan has not been considered which may be considered as future work.

Table 15: Literature review analysis.

meta-heuristic task allocation strategies [32] that not only focus on energy but also other parameters such as makespan, conform to SLA constraints, etc.

Acknowledgement

The author would like to thank all the co-authors for their guidance, support and useful comments.

References

[1] Buyya, R., Yeo, C. S., Venugopal, S., Broberg, J., & Brandic, I. (2009). *Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility*. Future Generation Computer Systems, Vol. 25, No. 6, pp. 599–616. https://doi.org/10.1016/j.future.2008.12.001

[2] Gombiner, J. (2011). *Carbon Footprinting the Internet. Consilience*, (5), 119-124. Retrieved July 7, 2020, from www.jstor.org/stable/26167805

[3] Krug, L., Shackleton, M., & Saffre, F. (2014). *Understanding the environmental costs of fixed line networking*. Proceedings of the 5th International Conference on Future Energy Systems - e-Energy ’14. https://doi.org/10.1145/2602044.2602057

[4] Ullman, J. D. (1975). *NP-complete scheduling problems.* Journal of Computer and System Sciences, 10(3), 384–393. https://doi.org/10.1016/s0022-0000(75)80008-0

[5] SPEC, Fujitsu FUJITSU Server PRIMERGY RX1330 M1, https://www.spec.org/power_ssj2008/results/res2014q3\/power_ssj2008-20140804-00662.html.

[6] SPEC, Inspur Corporation NF5280M4, https://www.spec.org/power_ssj2008/results/res2014q4/power_ssj2008-20140905-00673.html.

[7] SPEC, Dell Inc. PowerEdge R820 (Intel Xeon E5-4650 v2 2.40 GHz), https://www.spec.org/power_ssj2008/results/res2014q2/power_ssj2008-20140401-00654.html.

[8] SPEC, IBM Corporation IBM NeXtScale nx360 M4 (Intel Xeon E5-2660 v2), https://www.spec.org/power_ssj2008/results/res2014q2/power_ssj2008-20140421-00657.html.

[9] Lee, Y. C., & Zomaya, A. Y. (2010). *Energy efficient utilization of resources in cloud computing systems.* The Journal of Supercomputing, Vol. 60, No. 2, pp. 268–280. https://doi.org/10.1007/s11227-010-0421-3

[10] Gourisaria, M. K., Gupta, P., GM, H., Patra, S. S., Khilar, P. M. (2020). *A Comparative Study of Various Task Scheduling Algorithms in Cloud Computing*. International Journal of Control and Automation, Vol. 13, No. 4, pp. 1152-1169.

[11] Hsu, C.-H., Chen, S.-C., Lee, C.-C., Chang, H.-Y., Lai, K.-C., Li, K.-C., & Rong, C. (2011). *Energy-Aware Task Consolidation Technique for Cloud Computing*. 2011 IEEE Third International Conference on Cloud Computing Technology and Science. https://doi.org/10.1109/cloudcom.2011.25

[12] Meisner, D., Gold, B. T., & Wenisch, T. F. (2009). *PowerNap* Eliminating Server Idle Power, ACM SIGARCH Computer Architecture News, Vol. 37, No. 1, pp. 205. https://doi.org/10.1145/2528521.1508269

[13] Ismail, L., & Materwala, H. (2018). *EATSVM: Energy-Aware Task Scheduling on Cloud Virtual Machines*. Procedia Computer Science, Vol. 135, pp. 248–258. doi:10.1016/j.procs.2018.08.172

[14] Panda, S. K., & Jana, P. K. (2018). *An energy-efficient task scheduling algorithm for heterogeneous cloud computing systems*. Cluster Computing. https://doi.org/10.1007/s10586-018-2858-8

[15] Wu, C.-M., Chang, R.-S., & Chan, H.-Y. (2014*). A green energy-efficient scheduling algorithm using the DVFS technique for cloud datacenters*. Future Generation Computer Systems, Vol. 37, pp. 141–147. https://doi.org/10.1016/j.future.2013.06.009

[16] Kong, J., Choi, J., Choi, L., & Chung, S. W. (2008). *Low-cost application-aware DVFS for multi-core architecture*. In 2008 Third International Conference on Convergence and Hybrid Information Technology, Vol. 2, pp. 106-111. https://doi.org/10.1109/ICCIT.2008.124

[17] Kimura, H., Sato, M., Imada, T., & Hotta, Y. (2008). *Runtime DVFS control with instrumented code in power-scalable cluster system*. In 2008 IEEE International Conference on Cluster Computing, pp. 354-359. https://doi.org/10.1109/CLUSTER.2008.4663795

[18] Genser, A., Bachmann, C., Steger, C., Weiss, R., & Haid, J. (2010). *Power emulation based DVFS efficiency investigations for embedded systems.* In 2010 International Symposium on System on Chip, pp. 173-178. https://doi.org/10.1109/ISSOC.2010.5625559

[19] Khan, M. I., & Rinner, B. (2014). *Energy-aware task scheduling in wireless sensor networks based on cooperative reinforcement learning*. 2014 IEEE International Conference on Communications Workshops (ICC). https://doi.org/10.1109/iccw.2014.6881310

[20] Wen, G., Hong, J., Xu, C., Balaji, P., Feng, S., & Jiang, P. (2011). *Energy-aware hierarchical scheduling of applications in large scale data centers*. In 2011 International Conference on Cloud and Service Computing, pp. 158-165. https://doi.org/10.1109/CSC.2011.6138514

[21] Mishra, S. K., Sahoo, S., Sahoo, B., & Jena, S. K. (2019). *Energy-Efficient Service Allocation Techniques in Cloud: A Survey.* IETE Technical Review, pp. 1–14. https://doi.org/10.1080/02564602.2019.1620648

[22] Fan, X., Weber, W.-D., & Barroso, L. A. (2007). *Power provisioning for a warehouse-sized computer*. ACM SIGARCH Computer Architecture News, Vol. 35, No. 2, pp. 13. https://doi.org/10.1145/1273440.1250665

[23] Braun, T. D., Siegel, H. J., Beck, N., Bölöni, L. L., Maheswaran, M., Reuther, A. I., … Freund, R. F. (2001). *A Comparison of Eleven Static Heuristics for*

Mapping a Class of Independent Tasks onto Heterogeneous Distributed Computing Systems. Journal of Parallel and Distributed Computing, Vol. 61, No. 6, pp. 810–837. https://doi.org/10.1006/jpdc.2000.1714

[24] Ali, S., Siegel, H. J., Maheswaran, M., Hensgen, D., & Ali, S. (n.d.). *Task execution time modeling for heterogeneous computing systems*. Proceedings 9th Heterogeneous Computing Workshop (HCW 2000) (Cat. No.PR00556). https://doi.org/10.1109/hcw.2000.843743

[25] Zhang, Y., Cheng, X., Chen, L., & Shen, H. (2018). *Energy-efficient Tasks Scheduling Heuristics with Multi-constraints in Virtualized Clouds.* Journal of Grid Computing, Vol. 16, No. 3, pp. 459–475. https://doi.org/10.1007/s10723-018-9426-6

[26] Quan D. M., Somov, A., & Dupont, C. (2012). *Energy usage and carbon emission optimization mechanism for federated data centers*. In Energy Efficient Data Centres. Lecture Notes in Computer Science, Vol. 7396. Springer, Berlin, 129–140. https://doi.org/ 10.1007/978-3-642-33645-4_12

[27] Kliazovich, D., Bouvry, P., & Khan, S. U. (2010). *DENS: Data Center Energy-Efficient Network-Aware Scheduling*. 2010 IEEE/ACM Int'l Conference on Green Computing and Communications & Int'l Conference on Cyber, Physical and Social Computing. https://doi.org/10.1109/greencom-cpscom.2010.31

[28] Gourisaria, M.K., Patra, S.S., & Khilar, P.M. (2016). *Minimizing Energy Consumption by Task Consolidation in Cloud Centers with Optimized Resource Utilization.* International Journal of Electrical and Computer Engineering, Vol. 6, No. 6, pp. 3283-3292. https://doi.org/10.11591/ijece.v6i6.12251

[29] Gourisaria, M. K., Patra, S. S., & Khilar, P. M. (2018, December). *Energy saving task consolidation technique in cloud centers with resource utilization threshold.* International Conference on Advanced Computing and Intelligent Engineering. In Progress in Advanced Computing and Intelligent Engineering, pp. 655-666. Springer, Singapore. Bhubaneswar, https://doi.org/10.1007/978-981-10-6872-0_63

[30] Gourisaria, M. K., Samanta, A., Saha, A., Patra, S. S., & Khilar, P. M. (2020, March). *An Extensive Review on Cloud Computing*. 3rd International Conference on Data Engineering and Communication Technology, In Data Engineering and Communication Technology, pp. 53-78. Springer, Singapore. https://doi.org/10.1007/978-981-15-1097-7_6

[31] Sharma, R., Gourisaria, M. K., Patra, S. S. (2021). *Cloud Computing—Security, Issues, and Solutions*. Lecture Notes in Networks and Systems, Vol. 134, pp. 687–700. https://doi.org/10.1007/978-981-15-5397-4_70

[32] Bhardwaj, A. K., Gajpal, Y., Surti, C., & Gill, S. S. (2020). *HEART: Unrelated parallel machines problem with precedence constraints for task scheduling in cloud computing using heuristic and meta-heuristic algorithms*. Software: Practice and Experience, Vol. 50, No. 12, pp. 2231-2251. https://doi.org/10.1002/spe.2890

[33] Mohialdeen, I. A. (2013). *Comparative study of scheduling al-gorithms in cloud computing environment*. Journal of Computer Science, Vol. 9, No. 2, pp. 252-263. https://doi.org/10.3844/jcssp.2013.252.263

[34] Ruan, X., Chen H., Tian Y., & Yin, S. (2019). *Virtual machine allocation and migration based on performance-to-power ratio in energy-efficient clouds*. Future Generation Computer Systems, Vol. 100, pp. 380-394. https://doi.org/10.1016/j.future.2019.05.036

[35] Lin, W., Wang, W., Wu, W., Pang, X., Liu, B., & Zhang, Y. (2017). *A heuristic task scheduling algorithm based on server power efficiency model in cloud environments.* Sustainable computing: informatics and systems, Vol. 20, pp. 56-65. https://doi.org/10.1016/jsuscom.2017.10.007

[36] Chen, H. L., & Lv, S. (2015). *Adaptive bandwidth allocation strategy under cloud platform.* Informatica. Vol. 39, No. 4, pp. 347-352.

[37] Deshpande, P., Sharma, S. C., Peddoju, S. K., & Abraham, A. (2015). *Efficient multimedia data storage in cloud environment*. Informatica, vol. 39, No. 4, pp. 347-352.

[38] Dhaini, M., Jaber, M., Fakhereldine, A., Hamdan, S., & Haraty, A.R. (2021). *Green Computing Approaches – A Survey.* Informatica, vol. 45, No. 1, pp. 1-12. https://doi.org/10.31449/inf.v45i1.2998