IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received August 14, 2020, accepted August 27, 2020, date of publication September 1, 2020, date of current version September 15, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3020843

An Energy and Performance Aware Scheduler for Real-Time Tasks in Cloud Datacentres

HASHIM ALI[1], MUHAMMAD SHUAIB QURESHI[2], MUHAMMAD BILAL QURESHI[3], AYAZ ALI KHAN[1], MUHAMMAD ZAKARYA[1], (Member, IEEE), AND MUHAMMAD FAYAZ[2]

[1]Department of Computer Science, Abdul Wali Khan University, Mardan 23200, Pakistan
[2]Department of Computer Science, School of Arts and Sciences, University of Central Asia, Bishkek 720001, Kyrgyzstan
[3]Department of Computer Science, Shaheed Zulfikar Ali Bhutto Institute of Science and Technology, Islamabad 44000, Pakistan

Corresponding author: Muhammad Zakarya (mohd.zakarya@awkum.edu.pk)

This work was supported by the University of Central Asia, Kyrgyzstan.

ABSTRACT Datacentres provide the foundations for cloud computing, but require large amounts of electricity for their operation. Approaches that promise to reduce power use by minimizing execution time, for example using different scheduling and resource management techniques, are discussed in the literature. This paper summarizes some of the most important scheduling techniques in clouds focusing on power consumption, covering VM-level, host-level and task-level scheduling where the most promising approach is task level scheduling, with energy savings by means of load filtering, consolidation, adapted CPU throughput, or host power control. We explore use of the rate monotonic (RM) and backfilling algorithms for real-time task scheduling in cloud environment because RM is the simplest fixed priority scheduling technique, and thus the choice for modern real-time systems, and prior uses of RM in task scheduling have demonstrated power efficiency with optimal results. We specifically consider deadline-based tasks scheduling for real-time clouds which, to the best of our knowledge, has not been employed previously. RM with backfilling is experimentally evaluated and results show that, compared to the classical algorithms, all tasks were scheduled with minimum power consumption (5.5% - 29.3%), on minimum resources (3.9% - 25.2% less) while majority were meeting their deadlines (93.21% - 94.7%). The approach can guarantee deadline oriented Software as a Service (SaaS) in cloud if arrival rate i.e. network transfer time can be estimated in advance. We subsequently provided an extension of the proposed approach to task-based load balancing for almost balanced resource utilization and approximately 1.0% to 1.6% energy efficiency.

INDEX TERMS Energy efficiency, clouds, performance, scheduling algorithm, DVFS.

I. INTRODUCTION

Cloud computing [1] demonstrates a convergence between information technology and computer networks and business efficiency and adaptability. Conversely, this presents the technology providers and the research community with challenges in energy efficient computation. With rising energy costs, strategies which can increase the quantity of useful compute per unit of input energy, or decrease the amount of cooling required, become of interest. Such strategies include: (i) locating datacentres to take advantage of temperatures; (ii) adopting hardware with a better energy use profile; (iii) using scheduling strategies which can leave a maximum of equipment in a very low powered mode; and (iv) improving utilization through consolidation, often by taking advantage of virtualization or similar techniques / technologies. Datacentres will have theoretical peak power consumption, based on which specific measures of efficiency may be taken, but it is more likely that power demands vary some way below this peak, requiring continuous measurement. For the cloud provider, reduced energy consumption means increased margin. However, it would likely be detrimental if this margin came at a cost to performance and users' monetary costs.

The most frequently used policy to plan tasks is the priority driven approach, which can be categorized into two kinds: (i) fixed priority and (ii) dynamic priority [2], [3]. Fixed priority algorithms allocate each task a unique priority that cannot be changed if a task reappears for execution, to prioritize all tasks, within the system. On the other hand, dynamic priority scheduling algorithms place no constraints on the order of assigning priorities to individual tasks running in the system, and each task can take a different priority on its reappearing. Similarly, if more than one task were

The associate editor coordinating the review of this manuscript and approving it for publication was Jie Tang.

assigned the same priority, then the first one in queue is selected for scheduling. Amongst deadline monotonic (DM), the earliest deadline first (EDF) and RM, the later one is a fixed priority algorithm, well studied for scheduling in real-time systems where the tasks with minimum clock cycle are executed first and so on with a risk of long-running jobs missing their deadlines. We consider in our formulation that the amount of work is known in advance, as in offline scheduling. Task scheduling in such distributed environments is an active research issue [4] as most of the resources are not fully utilized and still consumes a lot of power. Therefore, it is important to study different scheduling approaches to fully utilize the available resources in virtualization based distributed systems that will help in efficient resource management in clouds environment in a power efficient way. The studies in [2], [3], [5], [6] discusses the importance of this issue, where our work fits to target the same issue to schedule jobs on minimum resources to fully utilize them.

Real-Time Services (RTSs) are those whose precision depends not only on logical results but also on the time in which these results are made. As Cloud computing becomes growing for Anything as a Service (XaaS) model, modern real-time cloud services including financial analysis, distributed databases, gaming applications, scientific experimentation, flight-control systems or image processing are also accessible through cloud computing. RTSs need huge volume of computing resources to scale user utilization patterns and satisfy time deadlines at the same time. Cloud computing model can provide this scalability within the timing constraints to these RTSs. A usual real-time service involves numerous Real-Time Applications (RTAs) that are further divided into subtasks. As long as a group of applications or tasks for a given real-time service meet all their deadlines, the service achieves the Quality of Service (QoS) settled with customers. At virtualization level, VM provisioning & allocation is considered a real-time service which are well studied in [7]. In this work, we investigate host level power-aware provisioning of cores or PEs (Processing Elements) for real-time applications divided into real-time tasks. Our work guarantee the scheduling of real-time tasks on PEs in a power efficient way before there deadline is met while ignoring the communication cost. We cannot guarantee the fulfilment of application deadline for customer satisfaction over Internet as the underlying communication medium in cloud because the data transfer to/from cloud is not managed by the cloud service providers. The major contributions of our work are:

1) we explore the use of RM algorithm for real-time task scheduling in cloud environment because RM is the simplest fixed priority scheduling technique, and thus the choice for modern real-time systems, and prior uses of RM in task scheduling have demonstrated power efficiency with optimal results;
2) we extend the classical RM technique with a backfilling scheduling mechanism to further improve energy and performance efficiencies;
3) we specifically consider deadline-based tasks scheduling for real-time clouds which, to the best of our knowledge, has not been employed previously;
4) RM is experimentally evaluated, and results shows that all tasks were scheduled with minimum power, on minimum resources while still meeting their deadlines; and
5) the approach can guarantee deadline-oriented Software as a Service (SaaS) in cloud if arrival rate i.e. network transfer time can be estimated in advance.

The rest of the paper is organized as follows. Sec. II is devoted to scheduling background. An overview of the related work along with various scheduling algorithms is presented in Sec. III. We formulate the existing problem in Sec. IV. We propose a solution for the existing problem in Sec. V; along with study of scheduling analysis and algorithms. Simulation results and comparative study are discussed in Sec. VI. Finally, Sec. VII concludes this article with some future research directions.

II. BACKGROUND

A scheduling approach that can reduce the power consumption of a system, while still meeting jobs deadline i.e. energy oriented task assignment is called green scheduling. It is also a designing technique for servers and other ICT (Information and Communication Technology) equipment's with minimal or no environmental effect. Task scheduling approaches are normally categorized as static and dynamic [2]. In static task scheduling, workload size (number of clock cycles), the required physical resources or VMs and task priorities are determined prior to their execution. Information about workload size, the Worst Case Execution Time (WCET), task' deadline and communication time is thought to be known at execution time. Min-min & Min-max are two common static scheduling techniques. In dynamic scheduling, algorithms may change tasks priority level on reappearing and resources or VMs (Virtual Machines) to running processes are allocated dynamically to maximize resource utilization. The workload is also not known, which makes dynamic algorithms more challenging and complex. Additionally, there exists another category of scheduling i.e. real time scheduling which comprise static and dynamic priority scheduling algorithms, as shown in Fig. 1.

To make scheduling green, famous techniques like Dynamic Voltage Scaling (DVS) and Dynamic Voltage & Frequency Scaling (DVFS) are integrated to take full advantage of machine utilization at low operating cost [6], [8], [9]. These techniques slow down the CPU frequency to save power, when workload amount falls below a threshold. Processing with slow frequency means that the CPU will take extra clock cycles to complete the running job. Therefore, if deadline based tasks and real-time data is considered, they might fail to optimally schedule the tasks. Therefore, it is important to look at these solutions when running real-time deadline based applications are considered in cloud environment. Sec. VI summarizes how resource scheduling

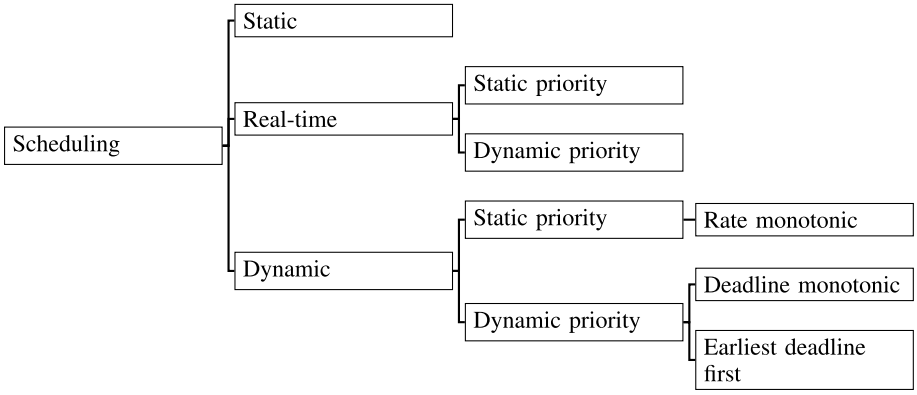

FIGURE 1. Categorization of the scheduling algorithms.

techniques are implemented in cloud datacentres for power efficiency. The local policies are implemented to make hardware more power efficient, while global policies are used with integration to local policies to schedule jobs and manage the available resources in a power efficient way. The local techniques are implemented on host level, while global policies are implemented on the virtualized level. The authors in [10], [11] have categorized schedulers in cloud systems into two major types i.e. local scheduler and global scheduler. Global scheduler receives submitted jobs from users, then depends on global scheduler policy, it chooses which job to send to which remote site. How to schedule all the jobs on its local resources is the responsibility of local scheduler of each remote site. Local scheduler implements Dynamic Power Management (DPM) techniques on a single physical host or PE to schedule jobs optimally e.g. VMware DRS (Distributed Resource Scheduler), DPM [12] and Credit scheduler [13].

In VMware VSphere DPM, the hypervisor is able to switch off and on some hosts, while the resource demand is low or high. Similarly, using DRS, the VMware VSphere hypervisor can optimize the resources to the best level by placing the new created VM on a well suitable host. DRS have also the capability of automated load balancing and optimized power consumption. The Credit scheduler in Xen hypervisor, allocate each host according to weight and cap. Each host is given a weight, and the CPU is allocated to each host according to its weight. The greater the domain weight, the more CPU is allocated to execute jobs on corresponding host. A domain with a weight of 512 will get twice as much CPU as a domain with a default weight of 256 on a contended host. A cap optionally fixes the amount of CPU, a domain will be able to consume. The cap is expressed in percentage of one physical CPU: 100 is 1 physical CPU, 50 is half a CPU, 200 is 2 CPUs, etc. The default is 0; means there is no upper cap. VM allocation policies are studied more in the literature [7], [11], [12], [14], [15]–[17] as their allocation, take over, placement in hosts and even migration affect the performance and cost of cloud infrastructure.

There are two different types of scheduling approaches in clouds: (i) host level; and (ii) VM level. In respect of (i), VMs are scheduled on hosts i.e. virtualization level; whereas, in respect of (ii) multiple CPU cores or PEs are allocated to user jobs i.e. system level. As discussed above, local policies are implemented on VM/PE level and global policies are working on virtualization level. In virtualization, a local policy controls guest Operating System (OS's) Power Management (PM) schemes i.e. hardware based techniques like DVS through scheduling jobs to different PEs on low voltage/frequency, while consolidation of VMs is controlled by global resource policies through live migration [12] to reallocate VMs. An example of local policy is the on-demand governor integrated into the Linux kernel [9]. A detail experiments on local and global policies can be found in [12], [17].

Time-shared and Space-shared are two scheduling policies well studied for resource management i.e. tasks allocation to VM and vice versa. Time-shared is similar to multi-tasking approach where VM is shared amongst different cloudlets that can further be shared in terms of PEs when threaded applications are considered, in that case time-sharing policy need not be enforced. Similarly, space shared implements First Come First Serve (FCFS) policy where resources are fully allocated to the running cloudlet. With confidence both approaches involve time-sharing of a short given a period T, two tasks A and B will both be completed in both approaches. In time-shared approach, this will be through swapping slices of tasks A and B. In space-shared, task A will be completed first. Fig. 2 shows the scheduling approach in CloudSim. A VM scheduler allocates multiple VMs to host and a cloudlet scheduler schedules all cloudlets to VMs for execution while agreeing Service Level Agreement (SLA).

III. RELATED WORK

Efficiency of scheduling approach affects cloud performance and maximizes profits for cloud service providers. In this section we have explained some common approaches that are studied for power efficiency in cloud datacentres. We follow the following taxonomy in Fig. 3 to discuss the existing literature and state-of-the-art energy efficiency techniques [18]. Real-time scheduling has been studied extensively in uniprocessor and multiprocessor system, but cloud environment lacks such study. Clouds are soft

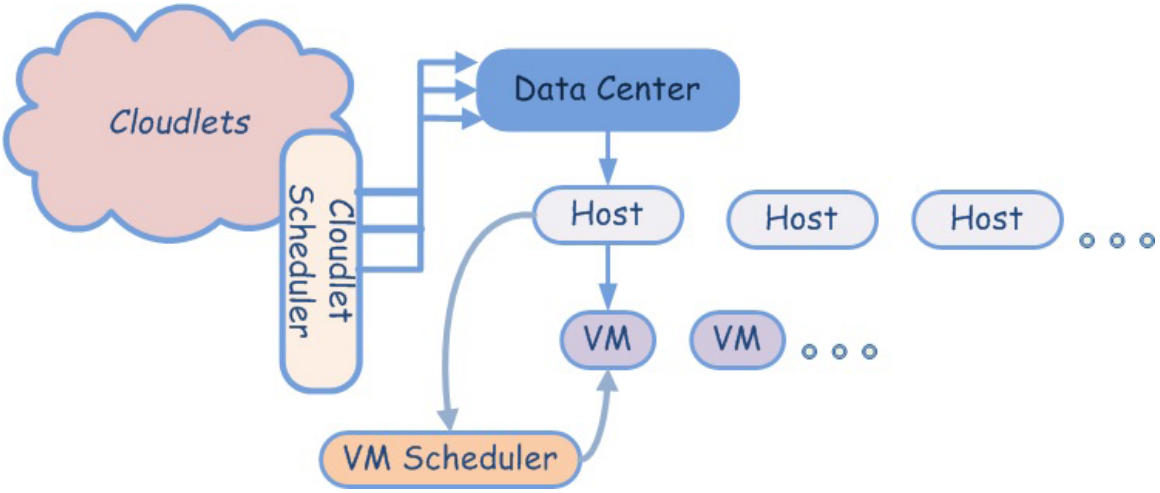

FIGURE 2. Default scheduling models in the CloudSim simulator.

real-time, means if a customer was not satisfied with a service provider due to service failure or delay, the customer will definitely move to another service provider which leads to the study of real-time clouds. The facility to fulfil timing constraints of real-time tasks plays an important role in cloud computing. To the best of our knowledge, the available cloud scheduling techniques are not appropriate for real-time tasks since they lack strict requirement of hard deadlines. A real-time scheduling approach must guarantee that processes meet deadlines, independent of system workload, for successful completion of job. Real-time scheduling is divided into two types: (i) Fixed priority algorithms like Rate Monotonic (RM); Deadline Monotonic (DM) [2]; and (ii) Dynamic priority algorithms like Earliest Deadline First (EDF). Fixed priority algorithms allocate each task a unique value that cannot be changed if a task reappears for execution, to prioritize all tasks, within the system. Dynamic priority scheduling algorithms place no constraints on the order of assigning priorities to individual tasks running in the system. RM prioritizes all the tasks based on the number of clock cycles (c_i) required, i.e. minimum c_i means high priority. DM considers the deadline (d_i), the nearer the tasks d_i, the higher the priority. EDF is same to DM, but it can re-prioritize the tasks upon its reappearing for execution. We consider VM level deadline based real-time tasks scheduling where PEs are allocated to user tasks with minimum feasible speed. The suitability of RM algorithm is studied for multicore systems in [8], [19]. They implemented RM technique to find the lowest core speed to schedule individual tasks on a multicore CPU. Then, the lightest task shifting policy is adapted to balance the core utilization, which is utilized to determine the uniform system speed for a given task set. Their work guarantees that all the tasks fulfil their deadlines with reduced system power consumption.

Servers or processing units are the most power consuming equipment's in datacentres [20]. It is very clear from our previous figures that the ratio of power is very small, when a CPU is 100% utilized and when it is idle. Therefore, it is more efficient to make the CPU engaged all the time, or even switch it off to save more power. In [5] the authors have reported on scheduling policies to decrease energy consumption of parallel tasks. In such systems, critical tasks cannot miss their deadline and should be executed before their deadline. The execution of non-critical tasks can be delayed that extend their total execution time. Therefore, they have used DVFS to scale the resource voltage or frequency for non-critical tasks which reduces the total energy consumption. Their Power Aware Task Clustering (PATC) algorithm can schedule task clusters on homogeneous PEs with an energy reduction of 39.7% due to PE idle state, task clusters for less communication and extending the make span. In [21] the authors have extended OS's power manager by adaptive power manager APM) that uses the CPU's DVS abilities to drop or upsurge its frequency to minimize overall power consumption. This hardware based techniques can be integrated with some scheduling policy, to make the system power efficient. For example, The DVS approach at the processor level in cooperation with turn on/off approach at cluster level is proposed in [22] to attain approximately 45% total energy savings while maintaining the response time.

Minimization of the total execution time of tasks is also studied in the literature. Such approaches maximize the performance of the systems using different optimization techniques. The objective of PSO-based Genetic Algorithm (PGA) [23], is to find a schedule that minimizes the WCET of all tasks scheduled on heterogeneous systems by combining the heuristic based Particle Swarm Optimization (PSO) policy and modified genetic operators. Their approach shows better outputs over Heterogeneous Earliest Finnish Time (HEFT) and Genetic Algorithm (GA). HEFT is a greedy approach to schedule a set of dependent

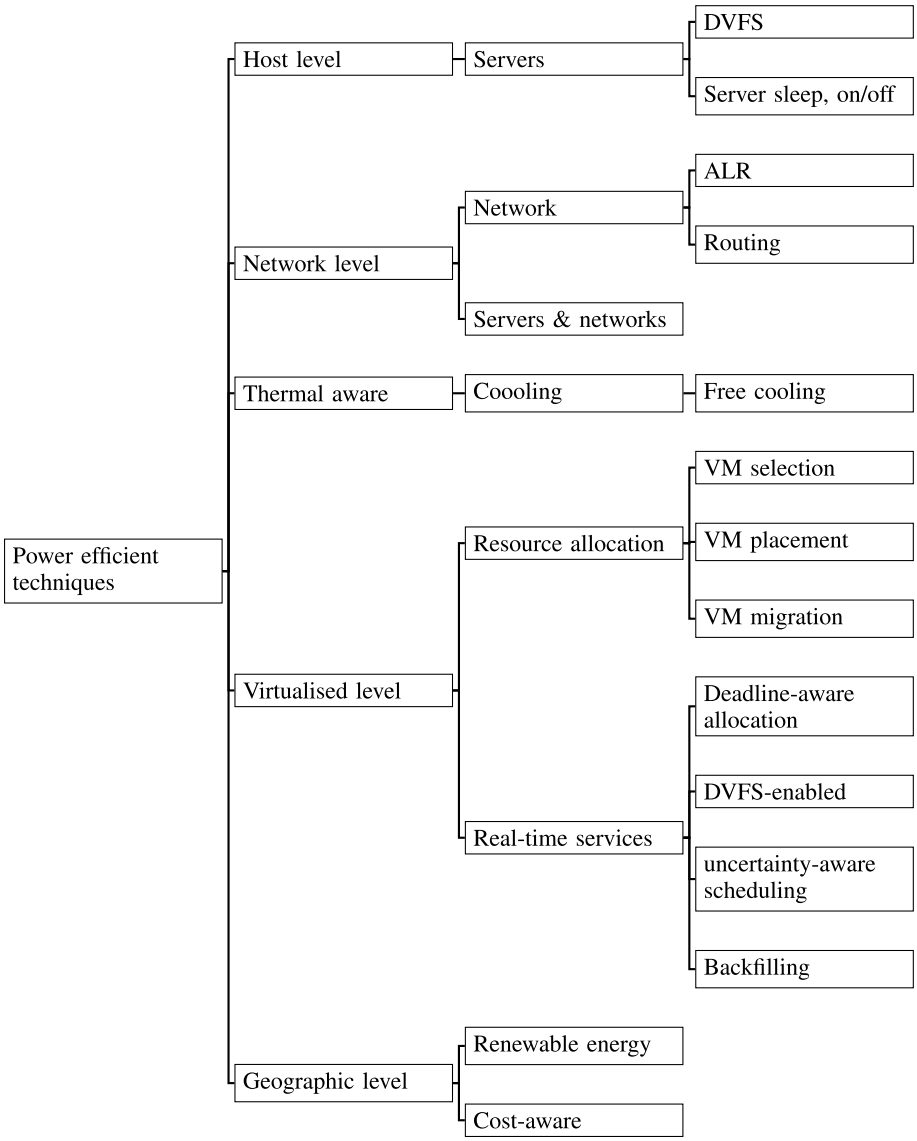

FIGURE 3. State-of-the-art energy efficient techniques.

tasks onto a system of heterogeneous processors taking communication time into account. All prioritized tasks are scheduled each one, starting with the highest priority. The task with the highest priority for which all dependent tasks have finished is scheduled on the processor which will result in the earliest finish time of that task [24]. GA is evolutionary-based optimization technique. In [25] a multi-objective GA (MO-GA) that optimizes the power consumption, CO_2 emissions and maximizes profit of a geographically dispersed cloud computing setup is presented. The results suggest that this approach beats the greedy approach in terms of energy consumption and Green House Gas (GHG) emissions and is slightly better in terms of scheduling more tasks per unit time. CO_2 emissions are calculated based on EPA's eGRID emission factors [26], where on average, electricity sources emit 1.222lbs CO_2 per kWh i.e. 0.0005925 metric tons CO_2 per kWh. Similarly, profit is estimated by using different price models, dependent on the use of services provided in $/CPU/hour. The study claims 10.85% reduction in CO_2 emission, 4.66% reduction in energy and 1.62% profit maximization. In [27] a PSO based scheduling algorithm maximizes profit by shortening the average operation time of tasks for cloud service provider with the lowest system costs while maintaining customer QoS over minimum number of fully utilized resources.

Much of the energy can be saved by maximum utilization of less number of machines, while switching off underutilized machines, in a datacentre. The objective can be achieved using migration techniques. In [14], [28], [29], the authors have defined a policy that tries to consolidate workloads from separate machines into smaller number of physical nodes while keeping most of the servers switched-off for long time and still satisfying resource requirements to maintain QoS of each job. This policy also considers virtualization overheads

such as VM creation, fault tolerance, and VM migration to periodically decide whether to transfer jobs and to turn off the less utilized servers resulting in 15% decrease in overall datacentre energy.

Efficient resource provisioning and management can save a lot of energy, if efficient algorithms are selected for VM selection, placement and migration. In VM selection, a proper VM is selected while during placement, a suitable host is selected to accommodate the selected VM. The objective is to manage the hosts in datacentre through balancing the workload or migration of VMs from underutilized host to switch them off, to save energy. When to migrate? Where to migrate? Which to migrate? Are some most challenging questions that are discussed in [17]. Research interests have been rising in introducing renewable energy sources into current datacentres. The basic challenge of such incorporation is that power sources are irregular due to seasonal effects. In [30] a holistic workload scheduling policy is presented to reduce the brown energy consumption in geographically dispersed datacentres with renewable power sources. In [30] the experiments with real workload traces show that the proposed policy significantly reduces brown energy consumption by up to 40% as compared to other techniques.

A newer approach uncertainty-aware online scheduling algorithm (ROSA) by Chen *et al.* [31] discusses an architecture for the execution of dynamic workflows having uncertain execution time for task as well as the time to transfer data within cloud environment. The uncertainties about unknown time to start, execute or finish a task is discussed in the proposed method. It helps to achieve optimal cost for service renting, changes in scheduling, reduced resource usage and fairness in resource usage for real time workflows having uncertain time for execution and time to transfer data in cloud environment. Results show that improvement in performance in comparison to existing algorithms up to 56% of cost, changes in scheduling up to 70%, resource usage up to 37% and the fairness up to 37% is achieved by the proposed method. Similarly, Chen *et al.* [32] presents an uncertainty aware framework to schedule real time tasks in the cloud environment. The work proposes a newer algorithm Proactive and Reactive Scheduling (PRS) which focuses on real time task scheduling along with computing resources in context to uncertainties present in the system. The effectiveness of PRS is simulated through experiments conducted on synthetic and Google workloads respectively. PRS uses proactive and reactive scheduling in a dynamic way to schedule real time, aperiodic and independent tasks. It also incorporates policies to scale-up and scale-down resources used in computing in connection to workload in order to improve energy efficiency, resource usage and reduced energy usage for datacentre in cloud environment is achieved.

The work by Nayak *et al.* [33] discusses multi criteria decision-making (MCDM) based task scheduling in collaboration with VIKOR, a multi criteria optimization method. VIKOR tries to find out the best task that is backfilled from the available similar tasks. It further helps to give optimal resources to the best task that is already allocated to the task that are deadline based. This results in increased resource utilization as well as helps to reduce tasks rejection. In the work by Safari and Khorsand [34], a power-aware and list based scheduling method is combined with DVFS (PL-DVFS) for real time tasks is proposed. PL-DVFS maintains QoS using tasks deadlines. It improves performance as well as reduces overall energy usage for execution time and communication energies respectively especially for the case where tasks are high in number. It also eliminates those hosts/VMs/CPUs which are inefficient in order to improve resource utilization levels.

In [35], a dynamic cost-efficient deadline-aware (DCEDA) tries to minimize execution of cost and time. It uses cost effective scheduling decisions along with continual update status of each job in order to avoid violation of deadline. In comparison with JIT-C and CEDA, DCEDA yields cheaper scheduling in order to satisfy deadline given by clients. In [36], authors have combined RMS and EDF algorithms to give a hybrid form of EDF-RM scheduling method. It makes it more flexible and optimal tool because of non-schedulable jobs are less in number and individual job's turnaround time rather than taking set of jobs in RMS/EDF. Authors in [37], have considered user's dynamic behaviour in Fog computing environment for dynamic changes in user requirements. The proposed algorithm tries to minimize average execution time by 12% as well as cost by 15% as compared with existing solutions like resource and latency aware methods. Reference [38] proposes an energy efficient scheduler DEWTS which is based over DVFS. It is applied on DVFS based processors in the datacentres. In context to previous methods, here jobs are placed over idle slots where voltage and frequency are low and it does not violate dependency limitations and increasing the delay time for a job to complete within the due time for a batch of jobs. Furthermore, [39] discusses heterogeneous VMs resource allocation for real time embedded systems. It executes the gathered data within the deadline from various sources like heterogeneous, distributed and decentralized nodes in cost and time effective manner. The proposed avoids data replication and embeds genetic operators for cuckoo algorithm in order to solve job to VM limitation optimization issue. A classical survey of the different scheduling algorithms is presented, in [40], using various metrics used within the cloud computing environment. An analysis of their limits and time complexity are observed. It is pertinent from the study that none of the scheduling algorithm captures all of the aspects simultaneously. Hence, they are open for further modifications to be used in the future. A new CbCP algorithm is proposed, in [41], where focus is given to minimize the total cost for execution in context to user's satisfaction over specific deadline and reliability. As compared with DRR and QFEC+, it overpasses both in terms of low sub reliability within the clusters. It also provides optimal solution wherever task graph for a simple condition is satisfied.

IV. PROBLEM FORMULATION

Consider a datacentre which comprises M VMs given by $M = \{m_1, m_2, \ldots, m_n\}$, and assume that each m_i is DVS enabled module where the frequency of each VM is f_i that is measured in cycles per unit time. Having DVS enabled, f_i can vary from $f_{i_{min}}$ to $f_{i_{max}}$, where $0 < f_{i_{min}} < f_{i_{max}}$. It is easy to get speed S_j of the VM that is proportional to the frequency f_i. Furthermore, heterogeneity of the VMs is given by $H(m_j)$ which includes processor architecture, supported buses types, and processor speed in GHz, I/O and memory in bytes.

Consider a workload T which includes n tasks given by $T = \{t_1, t_2, \ldots, t_n\}$ and $t_i = (c_i, p_i, d_i)$ where c_i is the number of CPU cycles that are needs to complete t_i execution. Similarly, p_i is the time period and d_i is the task's deadline. We assume that c_i is known in advance. $H(t_i)$ is the VM that complete t_i execution before d_i, which is absolute deadline for t_i. Relative deadline D of workload T is met if and only if all d_n for all tasks t_n are met individually on each VM with $H(t_n)$.

The total number of CPU cycles required by t_i to execute on VM m_j is assumed to be a finite positive number, denoted by c_{ij}. The execution time of t_i under a constant speed is $S_{ij} = \frac{t_{ij}}{c_{ij}}$, calculated in cycles per second. We also assume that the processor always retrieve t_i from the primary cache, reducing communication overhead. Assume that task t_i when executed on machine m_j consumes p_{ij} power. Reducing p_{ij} will also diminish f_i, and consequently will decrease S_{ij} and might cause t_i to probably miss its deadline d_i. If t_i is mapped to $H(t_i)$, we say that the architectural mapping is fulfilled, otherwise not. Our goal is to minimize the power consumption of VMs such that the performance in not affected, given by Eq. 1:

$$min \sum_{i=1}^{n} \sum_{i=1}^{m} p_{ij}.x_{ij} \tag{1}$$

where $x_{ij} \in \{0, 1\}$, that shows a boolean factor for architectural mapping, if a mapping occurs (i.e. a task is allocated to a VM) then $x_{ij} = 1$ otherwise $x_{ij} = 0$.

V. PROPOSED WORK

It is clear from [2], [19] that RM is the simplest and applicable amongst fixed priority scheduling technique and thus is a suitable choice for present real time systems and applications. We have studied the suitability of RM algorithm for real-time task scheduling in cloud systems. Our main objective is to execute user jobs on minimum power i.e. low frequency with increased utilization by assigning minimum resources. Our algorithm runs in three phases. In the first phase, a minimum number of VMs are allocated to execute jobs. In the second phase, all the feasible schedulable points are calculated according to RM algorithm for individual tasks and are compared with its energy requirements. A feasible schedulable point with minimum power consumption is chosen to execute the user job. In the third phase, the empty slots (CPU cycles) created by terminating tasks is are allocated through a backfilling policy. The algorithm is explained later in more detail. Initially, a single host with some VMs is activated while other hosts are kept switched off to save energy. When some tasks are submitted for execution, the available VMs are allocated using default simple VM allocation policy (first fit) one by one until specific threshold utilization is achieved. It is clear from our experiments that VMs have limited effect on utilization unless they are executing large tasks/cloudlets. Again it is logical, that one host with a number of idle VMs utilizes the host only a little bit, because the utilization is often calculated by the workload.

Consider a datacentre containing H heterogeneous hosts i.e. $H = \{h_1, h_2, h_3, \ldots, h_n\}$. Each host, following a round of scheduling, will be running zero or more isolated VMs given by $VM_h = \{vm_1, vm_2, vm_3, \ldots, vm_n\}$. Being heterogeneous, we consider each host as belonging to a subset of hosts, with each subset differentiated by processor architecture (CPU family and model). A specific subset can be further differentiated by multiples of the CPU, in terms of CPU cores which we may refer to as processor elements (PE) and CPU speed. The maximum work that each host can undertake per unit of time is then a factor of the architecture, number of PE, and speed of each PE. A VM, then, is allocated some number of PEs which gives it the possibility to undertake a specific proportion of the maximum for the host. To simplify concerns, we assume that hosts are comparable by a single measure which could be calculated in this manner such that performance ranking would be possible; for the sake of simulations only, we will use the MIPS (Millions of Instructions Per Second) specification as a proxy for such a calculated value, however we would not be able to endorse this as a good performance indicator for real systems. Consequently, each PE in a host would be capable of delivering inconsistently with respect to other PEs. We assume, further, that each PE is DVFS enabled, where decrease in voltage or frequency for that PE impacts linearly on the achievable work. The actual workload is not changed but it will take more execution time that will directly affect the scheduling approach. In deadline based scheduling, the task execution is flexible, so we focus only on increasing utilization by reducing PE frequency/voltage. Similarly, we consider offline scheduling, where all workload is already known with their deadlines. In online approach the problem is that incoming tasks will not be allocated for processing, but that's only true in space shared policy. A time-shared policy and dynamically deleting the completed tasks will resolve this issue while considering online scheduling approach. The most important question that who and when DVFS-enable decision is taken is not studied in our approach. We consider theoretically that each PE can execute cloudlets at different speed i.e. PE have predefined frequencies, and all tasks are scheduled from the start with a minimum frequency having minimum power consumption, feasible schedulable point. That's looks like the power save mode of DVFS in Linux kernel, which is discussed below in more detail. Utilization based power model was considered a proxy to predict the host

total power, the reason being that the CPU is the principal consumer of dynamic power and that its power is determined largely by its power state (active or sleeping). For non CPU intensive workload this assumption will fail. The utilization based power model is given by:

$$Power_{used} = P_{min} + (P_{max} - P_{min}) \times Utilization \quad (2)$$

where P_{min} is minimum power consumed at idle state and P_{max} is the maximum power consumed at peak load. Utilization is between 0 and 1. The power at task level scheduling was calculated using a very standard power model that shows when the clock frequency f and voltage V is changed then it will affect power consumption accordingly, where C is capacitance, and is given by:

$$P = C \times V^2 \times f \quad (3)$$

DVFS control looks very easy but indeed it is a more complex operation. Reducing CPU rate has a robust influence on performance that consumers may not accept. Likewise, if reducing CPU rate reduces power consumption, then the resulting slowdown may in fact lead to increased energy consumption as energy depends on power and execution time both [42]. Therefore, DVFS control is hard and needs exact policies to attain major power savings. DVFS is integrated as SpeedStep on Intel processors, or as Cool'n'Quiet on AMD processors. Besides, software support for DVFS is common amongst all major OS including Linux that comes with *cpufreq* permitting users to set the wanted frequency at any time. Linux kernel, support five diverse modes: (i) performance; (ii) power-save; (iii) user-space; (iv) conservative; and (v) on-demand to activate DVFS. A lower frequency implies a weaker voltage, decreases CPU power consumption but slows down the CPU computation capacity. Regarding the time spent with I/O operations, the efficiency of DVFS technique depends on the system architecture [9]. A very brief discussion, implementation on real host and simulations of such operating modes using DVFS is discussed and results are compared in terms of execution time and power saving in [9].

Assuming that each vm_i is DVS module enabled (DVS relates to the PE, so it means that each VM running on top of a CPU core i.e PE is DVS enabled with power save mode) where the frequency of each PE related to a specific VM i.e. vm_i is f_i which is measured in cycles per unit time. For the sake of simulation purposes only, we create one PE in one VM, hence VM and PE is used interchangeably. Although DVS is hardware based approach but considering that each VM gets a view of what is offered to it through hypervisor. If the workload can be scheduled with low speed, permitted by flexible or far away deadline, then VM gets lesser share of CPU to extends its completion time. Having DVS enabled, each vm_i have frequency f_i that can vary from $f_{i_{min}}$ to $f_{i_{max}}$, where $0 < f_{i_{min}} < f_{i_{max}}$. For simplicity, we consider the lower value of f_i as 0.1, otherwise having multiple VMs on a single host, we have a maximum number of cycles per second to share amongst the VMs and in some cases we also need to cycle the VMs in and out i.e. VM migration. In our scenario VM migration is not enabled but the cloudlet scheduler is time-shared enabled. It is easy to obtain speed S_j [Table 1] [8], of the VM corresponding to PE, which is proportional to the frequency f_i. Table 1 shows how the speed of each VM is calculated against the frequency f_i, which the VM is running with.

Consider a workload $W(C_i, T_i)$, where C_i is the total computational demand and T_i is the time period required to execute another instance of W, which also shows the deadline for W, if not fulfilled, the workload is not scheduled. Moreover, W include different tasks given by $W = \{w_1, w_2, w_3, \ldots, w_n\}$ and $w_i = (c_i, t_i)$, where c_i is the number of CPU cycles needed to complete w_i execution and t_i is the time of PE allocated for a short interval to execute a partial part of c_i. The time taken will depend on the useful amount of work doable per cycle on a given host (for a given microprocessor architecture), this becomes highly dependent in a heterogeneous setup. We assume that c_i is known in advance, which is the total number of MIPS required to complete the execution of w, in our simulation set-up. Predicting the workload MIPS is quite complex, but using AI techniques we can estimate it using different parameters. Some authors [15] consider the bandwidth to calculate the workload to assign only the optimal resources in cloud environment. We assume the number of MIPS is known as in offline scheduling. The total number of MIPS required by w_i to execute on VM vm_j is assumed to be a finite positive number as task must require some CPU cycles, denoted by c_{ij}. This suggests that c_i is an independent standard for measuring CPU cycles in our simulations. The execution time of w_i under a constant speed is $S_{ij} = \frac{w_{ij}}{c_{ij}}$, calculated in cycles per second, as wij is measured in (cycle per second) and c_{ij} denotes the number of MIPS in cycle per second. Our work only consider the scheduling cost, therefore we assume zero communication overhead and ignore the time to retrieve the cloudlets from future queue to deferred queue and from deferred queue to execution unit. Utilization of a VM is given by u_i leading to the total utilization U of a host. A datacentre contains large number of hosts and VMs where a single 100% utilized VM is not going to give us a sensible figure on datacentre overall utilization unless only one VM exists per host. Assume that task w_i when executed on VM vm_j consumes p_{ij} power per second. P is the total peak power of a host when a PE corresponding to a VM is 100% utilized until PE is less than the complete CPU. Note that, p_{ij} is bounded as shown.

$$\sum_{i=1}^{n} \sum_{j=1}^{m} p_{ij<p} \quad (4)$$

p_{ij} must be greater than or equal to 0 because negative power would breach the laws of physics. Similarly, u_i is directly proportional to p_{ij} i.e. the more a server i.e. host is utilized the more power is consumed. In some cases direct proportionality might not be true depending on consistency of voltage and

other system factors like fan speed etc. Reducing p_{ij} will also diminish f_i, and consequently will decrease S_{ij}. It is clear that reducing the power would cause problems for the hardware, but our purpose is to schedule the tasks with slow speed until their deadline is not missed. Our goal is to minimize the power consumption of a host or PE in a way that the performance in not affected.

$$\sum_{i=1}^{n}\sum_{j=1}^{m} p_{ij}.x_{ij} \tag{5}$$

where x_{ij} belong to 0 and 1, that shows a Boolean factor for architectural mapping or scheduling, if a mapping occurs i.e. a task is scheduled to a PE then $x_{ij} = 1$ otherwise $x_{ij} = 0$. In later case, when $x_{ij} = 0$, the above equation will result in 0 power, as no task was executed. In other way, these are the constraints of our scheduling problem.

The RM algorithm allocates static priorities on the basis of task periods such that for any two tasks t_i and t_j, priority (t_i) > priority (t_j) and period(t_i) < period(t_j). A task system is schedulable using RM algorithm iff:

$$U \leq n\left(2^{\frac{1}{n}} - 1\right) \tag{6}$$

where n denotes the number of tasks. It means that any task set of static priority is optimally feasible on a uniprocessor system using RM iff: U is not larger than 0.693, but, it has been proven in the literature that in average case RM is feasible for task set having $U = 0.88$ [43]. We assume that the workload is initially scheduled on a single PE, where $D_i = P_i$. At critical instant $t = 0$, the workload of task i at time t running at speed f_i is given by:

$$L_i(\tau) = C_i + \sum_{j=1}^{i-1} \lceil \frac{t}{P_j} \rceil \frac{(C_j)}{f_i} \tag{7}$$

This is the case for phase one when VM will distribute the workload equally amongst different available PEs. If the task is schedulable at f_i, it means it is also schedulable at some other schedulable point at different time t that is $< f_i$, and each task have different workload at different schedulable point, which leads to calculate the lowest PE speed as:

$$\min_{t \in S_i} \left(\frac{C_i + \sum_{j=1}^{i-1} \lceil \frac{t}{P_j} \rceil}{\tau} C_j \right) \leq f_t \tag{8}$$

In second phase all the schedulable points are calculated using Algorithm 1, where task i is always feasible on a single processor iff:

$$\min_{t \in S_i} L_i(t) \leq t \tag{9}$$

where t is a single schedulable point and S_i denotes a list of all schedulable points calculated as given below.

$$S_i = lP_j \left(J = 1, \ldots i \& l = 1, \ldots \lceil \frac{P_i}{P_j} \rceil \right) \tag{10}$$

Algorithm 1 Rate Monotonic Schedule (Rms)

Require: cloudletList
Ensure: feasiblePoints, infeasiblePoints

```
N = cloudletList.size()
C_i = cloudletList.cloudlet(i).getMIPS()
P_i = cloudletList.cloudlet(i).getPeriod()
index = 1
for each i from 1 to N do
    TP_i = P_i
    for each i from 1 to N do
        TP_i = P_i
        k = TP_i / TP_j
        for each m from 1 to k do
            schedule[index] = m × P_j
        end for
    end for
    if cloudletList.cloudlet(i).getMIPS()
                                <
    schedule[i] then
        feasiblePoints.add(schedule[i])
    else
        infeasiblePoints.add(schedule[i])
    end if
end for
return feasiblePoints, infeasiblePoints
```

where f_i is associated with task speed and feasible schedulable points, t is a scheduling point and S_i denotes a set of all the scheduling points. The feasibility of the workload is also checked i.e. if $L(i) \leq$ schedulable point, it means that the task is feasible otherwise it is added to the infeasible task list, that are considered for next iteration. In last, speed of every feasible task is calculated as:

$$Speed(i) = \frac{L_i}{SchedulabePoint} \tag{11}$$

We have scheduled individual task to PEs according to [2], [8]. The above equation for a single processor can be converted for multiprocessors as below.

$$\max_{t=1\ldots k} \{\min_{t \in S_i} \frac{(t)}{t}\} \leq f_t \tag{12}$$

The following steps, in Algorithm 1, show the process of finding schedulable points and allocating PEs for VM to execute W. In our approach some VMs are considered consolidated on a single host while other hosts are kept switched off to save energy. When there are some cloudlets to execute, the resource are made available in one-utilized-switch-on-another mode. We implement DVS technique with RM algorithm to schedule the cloudlets on a low power consumption schedulable point. At this level, we are not dealing with how much power was consumed, but we only focus on optimal schedulabilty of real-time cloudlets. Fig. 4 shows the basic flow diagram of RM scheduling technique.

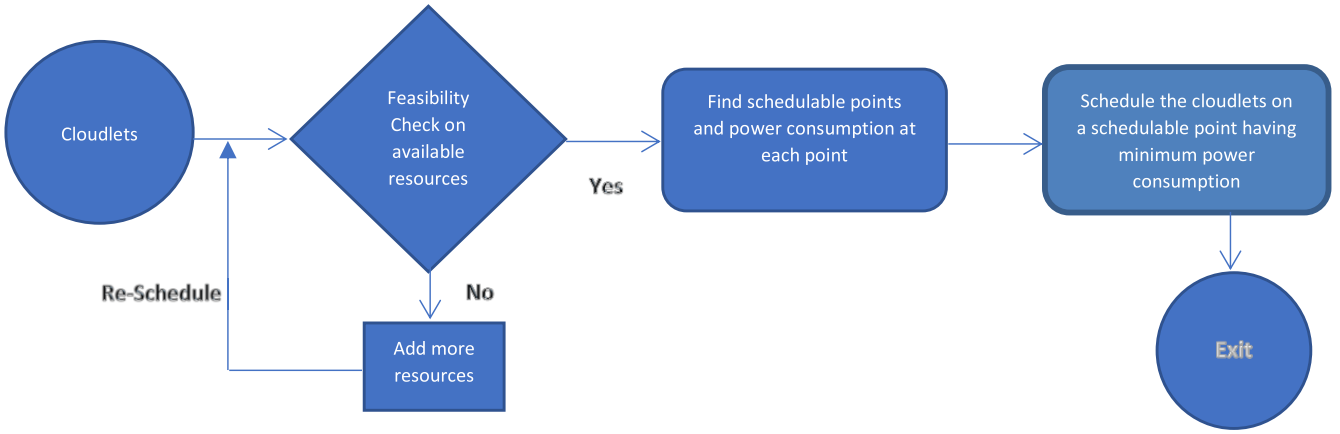

FIGURE 4. Flow diagram of the proposed scheduling algorithm.

A. THE BACKFILLING APPROACH

The backfilling approach can be used to schedule jobs from the wait queue if certain stranded resources or VMs, which cannot be allocated to the next job in the queue, exist in the system [44]. This ensures that the available resources are highly utilized and the chances of resource wastage are minimized. The pseudocode for the backfilling approach is shown in Alg. 2. The scheduling is achieved by initially sorting all tasks on their arrival times in accordance with their execution times. The tasks are checked whether their requirement is met with available free nodes, and also that they will finish before the next task from the queue is scheduled. It also checks requirement of minimal current free nodes as well as additional nodes. Afterwards, such task is used for application of backfilling. Backfilling gives distinct improvements in performance as well as in energy efficiency. Using the backfilling technique, the first incoming task is selected and, then, it continues by accepting next task having smaller execution time; and the process repeats itself further in a similar fashion. The mechanism adapted, here, is the pipelined method of execution where various tasks are executed simultaneously. In a conventional extended large systems, backfilling increases system usage about 20% as well as increased amount of turn around time.[1] The typical tendency of the backfilling method is to favour small size tasks having smaller execution times and resource requirements than that of larger tasks having higher execution times and resource requirements. On contrary, sites foresee improvement in service delivery for smaller tasks and no such improvement for larger tasks. Moreover, execution of such larger tasks often tend to have higher priority which is regretted by backfill method.

Algorithm 2 The Backfilling RM Approach

Require: cloudletList
Ensure: feasibleBackfill
1: sort cloudletList based on arrival times
2: $N_{vm} \leftarrow$ number of free VMs
3: firstJob $\leftarrow$ cloudletList[pop]
4: **while** cloudletList is not empty **do**
5: schedule firstJob using Alg. 1
6: $N_{vm} = N_{vm} - firstJob.NumberOfVMs$
7: **for** each job $\in$ cloudletList **do**
8: **if** job.NumberOfVMs $\leq N_{vm}$ **then**
9: feasibleBackfill $\leftarrow$ job
10: schedule job using Alg. 1
11: cloudletList[job] $\leftarrow$ NULL
12: **end if**
13: **end for**
14: no suitable job to backfill
15: feasibleBackfill $\leftarrow$ NULL
16: **end while**
17: **return** feasibleBackfill

B. THE LOAD BALANCING APPROACH

The backfilling approach, as presented in Sec. V-A, guarantees that free VM resources are allocated; however, it still does not ensure that the entire workload and resources are well-balanced. The rationale behind balancing the resources is that idle resources consume approximately 60% of the total energy consumption i.e. 100% utilized. Therefore, if utilization levels can be increased, then, the workload will essentially run on lower energy. Alg. 3 presents an approach to balance all tasks across all VM resources. An easy approach to well balance the utilization levels of all VMs is to compute the average utilization of the current resources and tasks [step 1 to 5]. In step 6 to 12, all VMs are categorized based on their utilization levels against the average. This marks all VMs with higher utilization level from which migration of tasks will occur. Finally, from step 13 to 22, those tasks which: (i) can be placed on lower utilized VMs; and (ii) the VM utilization level does not exceed the average utilization; are being migrated. The process repeats itself until all VMs in use are equally balanced and utilized.

[1] https://www.cs.huji.ac.il/course/2005/oop/exercises/ex2/ex2.html

Algorithm 3 The Load balancing Approach

Require: cloudletList, vmList
Ensure: loadBalance

```
totalUtil = 0
for each vm ∈ vmList do
    totalUtil ← totalUtil + vm.getUtil()
end for
averageUtil ← totalUtil ÷ vmList.size()
for each vm ∈ vmList do
    if vm.getUtil() > averageUtil then
        vmListHigh.add(vm)
    else
        vmListLow.add(vm)
    end if
end for
for each vm i ∈ vmListHigh do
    for each vm j ∈ vmListLow do
        for each task ∈ i do
            if j can host i.task & j.Util ≯averageUtil then
                j.add(task) i.e. allocate task using Alg. 1
            end if
        end for
    end for
end for
return loadBalance
```

C. NUMERICAL EXAMPLE

Given three cloudlets $cl_1(1.1, 3)$, $cl_2(1, 5)$ and $cl_3(1, 12)$, when RM scheduling theory is considered, cloudlet cl_3 is schedulable if and only if it satisfies:

$$\min_{t \in S_i} \left(\frac{C_i + \sum_{j=1}^{i-1} \lceil \frac{t}{P_j} \rceil}{\tau} C_j \right) \leq f_t \tag{13}$$

Here t_3 schedulable points are $S_3 = \{3, 5, 6, 9, 12\}$. It shows that due to the workload of cl_1 and cl_2, cloudlet cl_3 is also schedulable at 5, 6, 9, and 12 with speed of 0.86, 0.88, 0.72, and 0.76, respectively. The lowest speed is 0.72 that is achieved at the scheduling point 9. The giant charts are drawn for the cloudlets at the speeds of 0.86 and 0.72, respectively. When executed on maximum speed, C_i is less while when executing on lowest speed, the processor takes more time to complete one clock cycle. The task set when executed at the speed of 0.86 becomes $cl_1(1.33, 3)$, $cl_2(1.22, 5)$, $cl_3(1.22, 10)$ while at speed 0.72 task set is transformed into $cl_1(1.6, 3)$, $cl_2(1.45, 5)$, $cl_3(1.45, 10)$. The speed is calculated according to Table 1 [8]. It is clear from the giant charts, as shown in Fig. 5 below, that when executing jobs with slower CPU speeds, the PE is more utilized.

VI. EXPERIMENTAL RESULTS

There are two different scheduling policies that are studied in cloud computing: (i) host based scheduling; and (ii) VM based scheduling. On host level, VM are scheduled while on VM base, multiple cores or PEs are allocated to execute users tasks. Our approach is VM based scheduling policy for deadline based real-time tasks. We have checked the feasibility of this algorithm using MATLAB programming language. In MATLAB, a workload of 1,000 tasks was considered. Fig. 6 show total number of VMs that were allocated during the experiment (left) and power consumption (right), respectively. Fig. 6 only shows the ratio of power when the cloudlet was scheduled to a VM, not the actual power consumption of the overall system i.e. datacentre. All VMs are initially in off state and tasks are assigned in round robin fashion. When upper threshold utilization of 0.9 is achieved or submission of a task increases the upper threshold value, that task and other following tasks are assigned to a new VM. Power consumption was calculated with a specific value against each and every suitable point. In this scenario tasks were scheduled on availability of VM, means minimum frequency and minimum power suitable point. The entire workload was run over 28 VMs while most of them were maximally utilized i.e. approximately between 90% and 98.1%. To our best literature knowledge, we claim no such study is available in cloud systems, with the notable exception of [7] where VMs are considered as running real-time services while we, in this paper, consider real-time cloudlets.

After that CloudSim simple resource allocation policy was used to verify the feasibility of this approach. The period of cloudlet was considered the VM time using VMSchedulerTimeShared policy. In CloudSim, a cloudlet denotes a task that is submitted to a VM. A datacentre is a set of physical machines connected by a network available to receive the VMs and workloads for processing. The simulations to evaluate the RM scheduling algorithm were conducted with small datacentres having homogeneous and heterogeneous hosts in terms of VM available MIPS. For small datacentres, we considered that they have 5 hosts, 10 VMs, and a total of 10 cloudlets. Each VM was bind to a single cloudlet. Each host has 1 PE (the processor), 10,000 GB of disk space, 4 GB of RAM and gigabit Ethernet.

TABLE 1. Operational levels & speed range [8].

Level	f_i	speed range
0	0.1	0.01, 0.02, ... 0.09, 0.10
1	0.2	0.11, 0.12, ... 0.19, 0.20
2	0.3	0.21, 0.22, ... 0.29, 0.30
3	0.4	0.31, 0.32, ... 0.39, 0.40
4	0.5	0.41, 0.42, ... 0.49, 0.50
5	0.6	0.51, 0.52, ... 0.59, 0.60
6	0.7	0.61, 0.62, ... 0.69, 0.70
7	0.8	0.71, 0.72, ... 0.79, 0.80
8	0.9	0.81, 0.82, ... 0.89, 0.90
9	1.0	0.91, 0.92, ... 0.99, 1.00

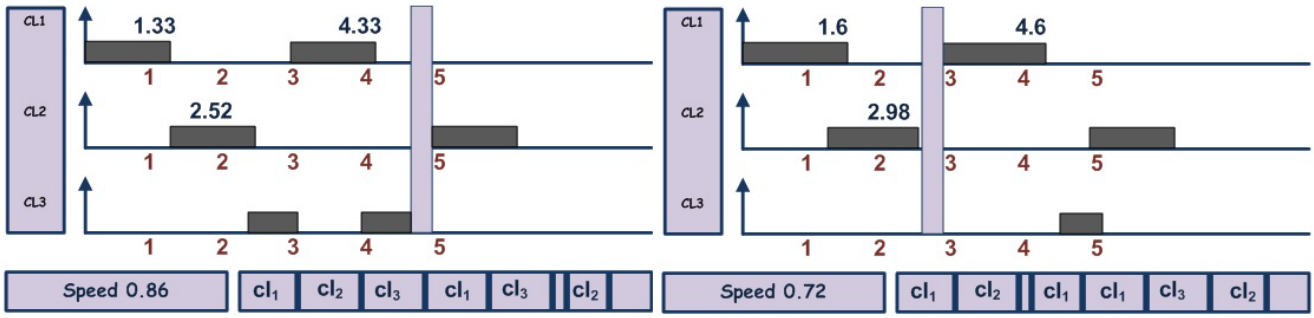

FIGURE 5. Comparison of running three cloudlets with different speeds.

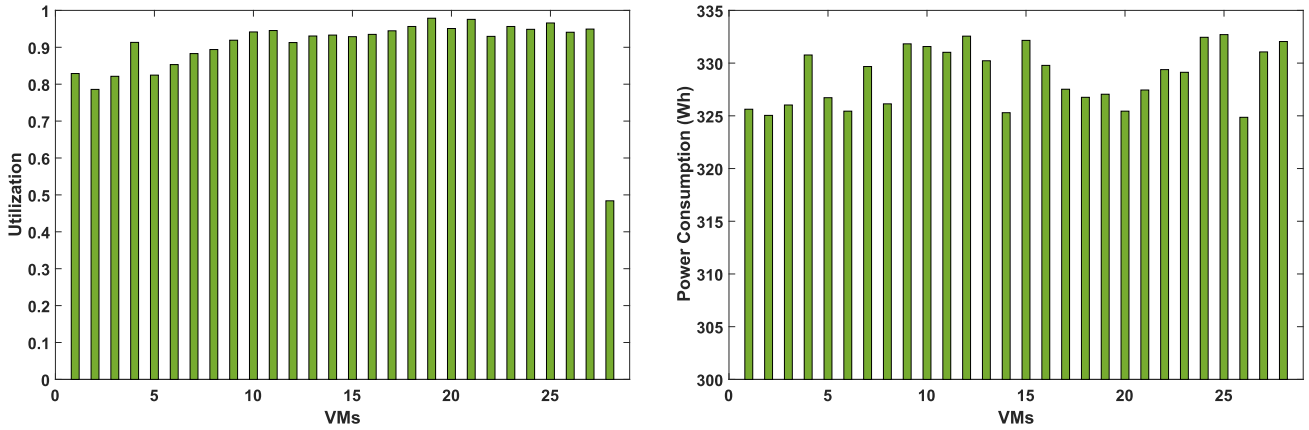

FIGURE 6. Number of VMs allocated during the experiment, VM utilization and VM level power consumption.

It is assumed that the hosts are running on a datacentre with x86 architecture, Xen as VM monitor, and Linux as OS. Each cloudlet uses a single PE, having 300 bytes data before processing and 300 bytes of data after the processing (standard of CloudSim models). Each cloudlet in a datacentre has 10,000 Millions of Instructions in a round-robin fashion. The datacentre VMs have processing capabilities of 1,000 and 2,000 MIPS in a round-robin fashion. For example, in a simulation where 6 VMs are generated, 3 VMs are created with 1,000 MIPS and 3 VMs with 2,000 MIPS. Moreover, it was adopted that each VM has 256 MB of RAM, 1,000 Kbps of bandwidth, 2,500 MB of image size and Xen as VM monitor. For assessment purposes, the important metrics of the simulation are the cloudlets completion time, the speed at which it was scheduled and power consumption. Fig. 7 shows the results in terms of cloudlet length, speed at which VM executed these cloudlets and the power each VM consumed. It is clear that all tasks were scheduled optimally with a uniform speed. In the literature, we do not find any deadline based real-time tasks schedulers for cloud datacentre, therefore this work is an initial effort on scheduling deadline based real-time tasks for VMs. The approach in [7] considers real-time VM allocation and not real-time tasks.

The presented RM with backfilling technique was further extended to task based load balancing to achieve power efficiency and increased levels of VM resource utilization. The load balancing approach was implemented as an optimization module in the CloudSim simulator, that periodically checks whether certain VMs are mostly loaded than others and vice versa. From implementation point of view, this was implemented as part of the RM scheduling policy. Each after five minute intervals, tasks were balanced across all running VMs. It is concluded in [45] that task based load balancing is more effective in terms of memory transfer during VM migration. Based on these results, we simulated migration of different tasks to balance the load amongst different VMs. The results of the balancing algorithm are shown in Fig. 8. The amount of energy saved, tasks deadlines, and total number of used VMs are shown in Table 2. Compared to RM with backfilling approach (RM-BF) the addition of load balancing (RM-BF-LB) could save approximately 1.0% to 1.6% energy.

A. COMPARISON WITH THE CLOSEST RIVALS

In this section, we evaluate the performance of the proposed RM with backfilling technique against the: (i) classical RM; (ii) No DVFS (first come first serve - FCFS); (iii) No DVFS (first fit - FF); and (iv) No DVFS (Random) scheduling algorithms. All these algorithms were simulated using similar datacentre setup, parameters and hosts' characteristics such as energy consumption etc. These policies were implemented

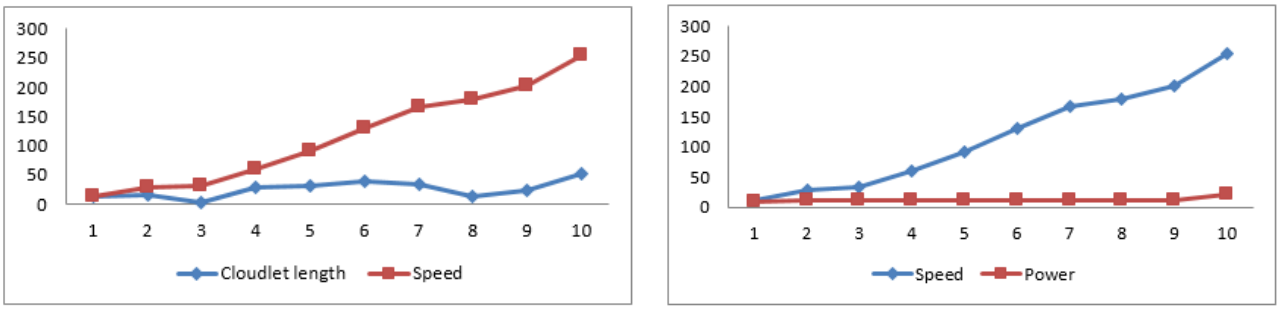

FIGURE 7. RM speed and RM power vs. cloudlet length (C_i).

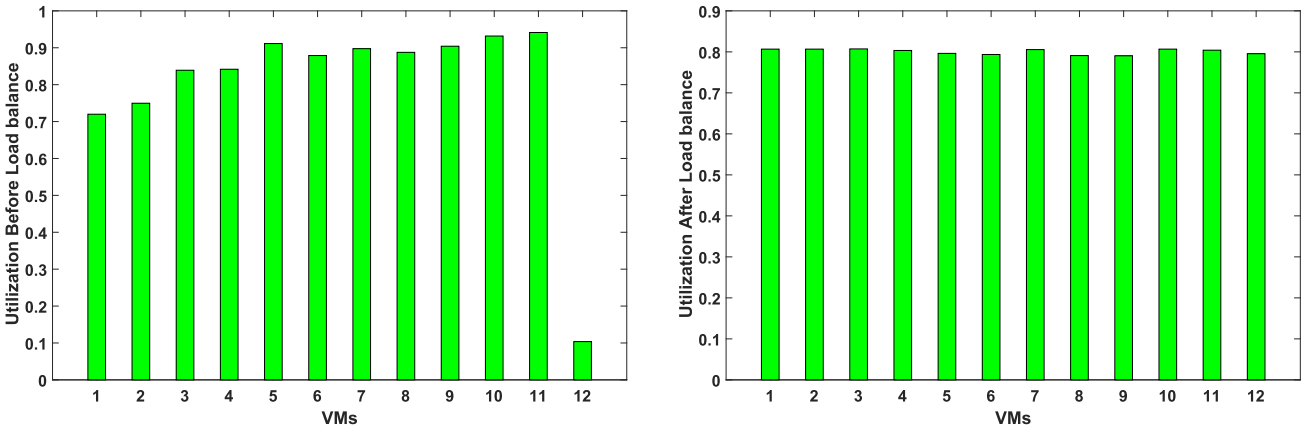

FIGURE 8. Utilization of VMs before (left) and after (right) load balancing.

TABLE 2. Comparison with the closest rivals and classical non-energy-aware scheduling techniques [the experiments were run ten times with different number of tasks, VMs; and the results are averaged over various runs].

Scheduling method	Number of tasks	Number of hosts	Number of VMs	Energy usage (KWh)	% of tasks with met deadlines	VMs used
Random	900	50	100	198.34	59.7%	100
FCFS				177.3	63.56%	100
FF				171.98	60.3%	79
RM				134.56	92.8%	77
RM-BF				134.89	93.21%	71
RM-BF-LB				133.54	93.04%	70
Random	2000	300	600	1297.78	56.9%	600
FCFS				1153.67	59.2%	600
FF				1083.45	72.88%	498
RM				971.23	93.76%	467
RM-BF				917.89	94.7%	449
RM-BF-LB				903.81	94.61%	437

through extending the abstract classes of the VM scheduler class in the CloudSim package. Various evaluation metrics including total energy consumption, and number of tasks which met their deadlines. The results are shown in Table 2. Our experimental evaluation of the proposed RM with backfilling suggests that, compared to the classical algorithms i.e. Random and RM with no backfilling, all tasks were scheduled with minimum energy consumption (5.5% - 29.3%), on minimum resources (3.9% - 25.2% less) while majority were meeting their deadlines (93.21% - 94.7%), respectively. We also observed a slight increase in the energy consumption of datacentre in the implementation of the backfilling approach which is reasonable due to higher resource utilization.

VII. CONCLUSION AND FUTURE WORK

In this paper, we have reported on the feasibility study of RM scheduling algorithm. In modern ages, there has been a rush in research on power efficient scheduling with high focus on grid computing and cloud datacentres. PSO, game theory, discrete optimization, heuristics, list scheduling, genetic algorithms, clustering algorithms, goal programming, task duplication based approaches and linear optimization have been widely studied in the literature to find solutions to achieve energy efficiency in high performance computing [16], [19], [46], [47], [48], [49]. A real-time scheduling approach must guarantee that processes meet deadlines, unrelated of system workload. The surviving cloud scheduling techniques in CloudSim are not appropriate for real-time tasks, since they lack strict requirement of hard deadlines. The facility to fulfil timing constraints of such real-time requests plays an important role in cloud atmosphere. Current SLAs cannot offer cloud customers with real-time control over their applications, therefore flexible and transparent SLAs are needed.

In the future, we will work around improving the load balancing approach for the current workloads across VMs in order to account for migration costs, performance degradation and user costs - since users costs are subject to execution times. Lower utilization levels can decrease the energy efficiency while deadlines are largely met due to availability of resources. However, increasing utilization levels may decrease workload performance, in particular, if co-located workloads on a particular host compete for similar or same resources. Lower performance means longer execution times that will subsequently cost more money for cloud customers and deadline misses. Deadline misses could, at least, result in SLA violations that may result in penalties to service providers, therefore, lower revenues. Our future work will investigate these types of scheduling impacts over energy efficiency, workload performance, and users' monetary costs [20], [50].

ACKNOWLEDGMENT

The authors are thankful to anonymous reviewers for their constructive comments on our work in order to produce a high quality manuscript for publication.

REFERENCES

[1] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, and I. Brandic, "Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility," *Future Gener. Comput. Syst.*, vol. 25, no. 6, pp. 599–616, Jun. 2009.

[2] N. Min-Allah, S. U. Khan, and W. Yongji, "Optimal task execution times for periodic tasks using nonlinear constrained optimization," *J. Supercomput.*, vol. 59, no. 3, pp. 1120–1138, Mar. 2012.

[3] J. Goossens, S. Funk, and S. Baruah, "Priority-driven scheduling of periodic task systems on multiprocessors," *Real-time Syst.*, vol. 25, nos. 2–3, pp. 187–205, 2003.

[4] R. Buyya, D. Abramson, J. Giddy, and H. Stockinger, "Economic models for resource management and scheduling in grid computing," *Concurrency Comput., Pract. Exper.*, vol. 14, nos. 13–15, pp. 1507–1542, Nov. 2002.

[5] L. Wang, J. Tao, G. von Laszewski, and D. Chen, "Power aware scheduling for parallel tasks via task clustering," in *Proc. IEEE 16th Int. Conf. Parallel Distrib. Syst.*, Dec. 2010, pp. 629–634.

[6] K. H. Kim, R. Buyya, and J. Kim, "Power aware scheduling of bag-of-tasks applications with deadline constraints on DVS-enabled clusters," in *Proc. 7th IEEE Int. Symp. Cluster Comput. Grid (CCGrid)*, May 2007, pp. 541–548.

[7] K. H. Kim, A. Beloglazov, and R. Buyya, "Power-aware provisioning of virtual machines for real-time cloud services," *Concurrency Comput., Pract. Exper.*, vol. 23, no. 13, pp. 1491–1505, Sep. 2011.

[8] N. Min-Allah, H. Hussain, S. U. Khan, and A. Y. Zomaya, "Power efficient rate monotonic scheduling for multi-core systems," *J. Parallel Distrib. Comput.*, vol. 72, no. 1, pp. 48–57, Jan. 2012.

[9] T. Guérout, T. Monteil, G. Da Costa, R. Neves Calheiros, R. Buyya, and M. Alexandru, "Energy-aware simulation with DVFS," *Simul. Model. Pract. Theory*, vol. 39, pp. 76–91, Dec. 2013.

[10] K. Ranganathan and I. Foster, "Simulation studies of computation and data scheduling algorithms for data grids," *J. Grid Comput.*, vol. 1, no. 1, pp. 53–62, 2003.

[11] J. Hu, J. Gu, G. Sun, and T. Zhao, "A scheduling strategy on load balancing of virtual machine resources in cloud computing environment," in *Proc. 3rd Int. Symp. Parallel Archit., Algorithms Program.*, Dec. 2010, pp. 89–96.

[12] R. Nathuji, C. Isci, and E. Gorbatov, "Exploiting platform heterogeneity for power efficient data centers," in *Proc. 4th Int. Conf. Autonomic Comput. (ICAC)*, Jun. 2007, p. 5.

[13] L. Cherkasova, D. Gupta, and A. Vahdat, "Comparison of the three CPU schedulers in xen," *ACM SIGMETRICS Perform. Eval. Rev.*, vol. 35, no. 2, pp. 42–51, Sep. 2007.

[14] I. Goiri, F. Julia, R. Nou, J. L. Berral, J. Guitart, and J. Torres, "Energy-aware scheduling in virtualized datacenters," in *Proc. IEEE Int. Conf. Cluster Comput.*, Sep. 2010, pp. 58–67.

[15] J. Shuja, K. Bilal, S. A. Madani, and S. U. Khan, "Data center energy efficient resource scheduling," *Cluster Comput.*, vol. 17, no. 4, pp. 1265–1277, Dec. 2014.

[16] Y. Kessaci, N. Melab, and E.-G. Talbi, "A multi-start local search heuristic for an energy efficient VMs assignment on top of the OpenNebula cloud manager," *Future Gener. Comput. Syst.*, vol. 36, pp. 237–256, Jul. 2014.

[17] A. Beloglazov, "Energy-efficient management of virtual machines in data centers for cloud computing," Ph.D. dissertation, 2013. [Online]. Available: https://minerva-access.unimelb.edu.au/handle/11343/38198

[18] M. Zakarya, "Energy, performance and cost efficient datacenters: A survey," *Renew. Sustain. Energy Rev.*, vol. 94, pp. 363–385, Oct. 2018.

[19] N. Min-Allah, S. U. Khan, N. Ghani, J. Li, L. Wang, and P. Bouvry, "A comparative study of rate monotonic schedulability tests," *J. Supercomput.*, vol. 59, no. 3, pp. 1419–1430, Mar. 2012.

[20] M. Zakarya and L. Gillam, "Managing energy, performance and cost in large scale heterogeneous datacenters using migrations," *Future Gener. Comput. Syst.*, vol. 93, pp. 529–547, Apr. 2019.

[21] T. F. Abdelzaher and C. Lu, "Schedulability analysis and utilization bounds for highly scalable real-time services," in *Proc. 7th IEEE Real-Time Technol. Appl. Symp.*, 2001, pp. 15–25.

[22] C. Rusu, A. Ferreira, C. Scordino, and A. Watson, "Energy-efficient real-time heterogeneous server clusters," in *Proc. 12th IEEE Real-Time Embedded Technol. Appl. Symp. (RTAS)*, 2006, pp. 418–428.

[23] Y. Kang, H. Lu, and J. He, "A PSO-based genetic algorithm for scheduling of tasks in a heterogeneous distributed system," *J. Softw.*, vol. 8, no. 6, pp. 1443–1450, Jun. 2013.

[24] H. Zhao and R. Sakellariou, "An experimental investigation into the rank function of the heterogeneous earliest finish time scheduling algorithm," in *Proc. Eur. Conf. Parallel Process.* Berlin, Germany: Springer, 2003, pp. 189–194.

[25] Y. Kessaci, N. Melab, and E.-G. Talbi, "A Pareto-based GA for scheduling HPC applications on distributed cloud infrastructures," in *Proc. Int. Conf. High Perform. Comput. Simul.*, Jul. 2011, pp. 456–462.

[26] M. Hendryx, E. Fedorko, and J. Halverson, "Pollution sources and mortality rates across rural-urban areas in the united states," *J. Rural Health*, vol. 26, no. 4, pp. 383–391, Jul. 2010.

[27] S. Zhan and H. Huo, "Improved PSO-based task scheduling algorithm in cloud computing," *J. Inf. Comput. Sci.*, vol. 9, no. 13, pp. 3821–3829, 2012.

[28] E. Pinheiro, R. Bianchini, E. V. Carrera, and T. Heath, "Load balancing and unbalancing for power and performance in cluster-based systems," Dept. Comput. Sci., Rutgers Univ., Piscataway, NJ, USA, Tech. Rep. DCS-TR-440, 2001. [Online]. Available: https://rucore.libraries.rutgers.edu/rutgers-lib/59449/, doi: 10.7282/t3-agfw-yt73.

[29] A. A. Khan, A. Ali, M. Zakarya, R. Khan, M. Khan, I. U. Rahman, and M. A. A. Rahman, "A migration aware scheduling technique for real-time aperiodic tasks over multiprocessor systems," *IEEE Access*, vol. 7, pp. 27859–27873, 2019.
[30] C. Chen, B. He, and X. Tang, "Green-aware workload scheduling in geographically distributed data centers," in *Proc. 4th IEEE Int. Conf. Cloud Comput. Technol. Sci.*, Dec. 2012, pp. 82–89.
[31] H. Chen, X. Zhu, G. Liu, and W. Pedrycz, "Uncertainty-aware online scheduling for real-time workflows in cloud service environment," *IEEE Trans. Services Comput.*, early access, Aug. 21, 2019, doi: 10.1109/TSC.2018.2866421.
[32] H. Chen, X. Zhu, H. Guo, J. Zhu, X. Qin, and J. Wu, "Towards energy-efficient scheduling for real-time tasks under uncertain cloud computing environment," *J. Syst. Softw.*, vol. 99, pp. 20–35, Jan. 2015.
[33] S. C. Nayak and C. Tripathy, "Deadline based task scheduling using multi-criteria decision-making in cloud environment," *Ain Shams Eng. J.*, vol. 9, no. 4, pp. 3315–3324, Dec. 2018.
[34] M. Safari and R. Khorsand, "PL-DVFS: Combining power-aware list-based scheduling algorithm with DVFS technique for real-time tasks in cloud computing," *J. Supercomput.*, vol. 74, no. 10, pp. 5578–5600, Oct. 2018.
[35] W. Ahmad, B. Alam, S. Ahuja, and S. Malik, "A dynamic VM provisioning and de-provisioning based cost-efficient deadline-aware scheduling algorithm for big data workflow applications in a cloud environment," *Cluster Comput.*, to be published, doi: 10.1007/s10586-020-03100-7.
[36] R. Sharma, N. Nitin, M. A. R. AlShehri, and D. Dahiya, "Priority-based joint EDF—RM scheduling algorithm for individual real-time task on distributed systems," *J. Supercomput.*, to be published, doi: 10.1007/s11227-020-03306-x.
[37] R. K. Naha, S. Garg, A. Chan, and S. K. Battula, "Deadline-based dynamic resource allocation and provisioning algorithms in fog-cloud environment," *Future Gener. Comput. Syst.*, vol. 104, pp. 131–141, Mar. 2020.
[38] Z. Tang, L. Qi, Z. Cheng, K. Li, S. U. Khan, and K. Li, "An energy-efficient task scheduling algorithm in DVFS-enabled cloud environment," *J. Grid Comput.*, vol. 14, no. 1, pp. 55–74, Mar. 2016.
[39] N. Min-Allah, M. B. Qureshi, S. Alrashed, and O. F. Rana, "Cost efficient resource allocation for real-time tasks in embedded systems," *Sustain. Cities Soc.*, vol. 48, Jul. 2019, Art. no. 101523.
[40] A. Arunarani, D. Manjula, and V. Sugumaran, "Task scheduling techniques in cloud computing: A literature survey," *Future Gener. Comput. Syst.*, vol. 91, pp. 407–415, Feb. 2019.
[41] S. S. Mousavi Nik, M. Naghibzadeh, and Y. Sedaghat, "Cost-driven workflow scheduling on the cloud with deadline and reliability constraints," *Computing*, vol. 102, no. 2, pp. 477–500, Feb. 2020.
[42] J.-P. Halimi, B. Pradelle, A. Guermouche, N. Triquenaux, A. Laurent, J. C. Beyler, and W. Jalby, "Reactive DVFS control for multicore processors," in *Proc. IEEE Int. Conf. Green Comput. Commun. IEEE Internet Things IEEE Cyber, Phys. Social Comput.*, Aug. 2013, pp. 102–109.
[43] C. L. Liu and J. W. Layland, "Scheduling algorithms for multiprogramming in a Hard-Real-Time environment," *J. ACM (JACM)*, vol. 20, no. 1, pp. 46–61, Jan. 1973.
[44] D. Tsafrir, Y. Etsion, and D. G. Feitelson, "Backfilling using system-generated predictions rather than user runtime estimates," *IEEE Trans. Parallel Distrib. Syst.*, vol. 18, no. 6, pp. 789–803, Jun. 2007.
[45] F. Ramezani, J. Lu, and F. K. Hussain, "Task-based system load balancing in cloud computing using particle swarm optimization," *Int. J. Parallel Program.*, vol. 42, no. 5, pp. 739–754, Oct. 2014.
[46] S. Khan and C. Ardil, "A game theoretical energy efficient resource allocation technique for large distributed computing systems," in *Proc. PDPTA*, Jul. 2009, pp. 48–54.
[47] D. Kliazovich, P. Bouvry, and S. U. Khan, "DENS: Data center energy-efficient network-aware scheduling," *Cluster Comput.*, vol. 16, no. 1, pp. 65–75, 2013.
[48] S. U. Khan and N. Min-Allah, "A goal programming based energy efficient resource allocation in data centers," *J. Supercomput.*, vol. 61, no. 3, pp. 502–519, 2012.
[49] Y. Gao, Z. Qi, Y. Wu, R. Wang, L. Liu, J. Xu, and H. Guan, "A power and performance management framework for virtualized server clusters," in *Proc. IEEE/ACM Int. Conf. Green Comput. Commun.*, Aug. 2011, pp. 170–175.
[50] A. A. Khan, M. Zakarya, R. Khan, I. U. Rahman, and M. Khan, "An energy, performance efficient resource consolidation scheme for heterogeneous cloud datacenters," *J. Netw. Comput. Appl.*, vol. 150, Jan. 2020, Art. no. 102497.

HASHIM ALI received the M.Phil. (M.S.) degree in computer science from the COMSATS Institute of Information Technology (CIIT), Abbotabad, Pakistan, where he is currently pursuing the Ph.D. degree. He is currently an Assistant Professor with the Department of Computer Science, Abdul Wali Khan University Mardan (AWKUM), Pakistan. His research interests include cloud computing, software testing, agile processes, energy and performance efficient computing, as well as distributed systems and enterprise systems. He is proficient in computer systems both theoretically and practically.

MUHAMMAD SHUAIB QURESHI is currently an Assistant Professor with the Department of Computer Science, University of Central Asia, Kyrgyzstan. He has been working with the HPC Research Group KAU, Saudi Arabia, on many research and development projects, spanning the areas of resource allocation in HPC systems, RTS scheduling theory, as well as eHealth and mHealth systems. Having vast research experience, he has authored more than 40 research articles and book chapters published in premier journals, including the IEEE Access, the *Journal of Grid Computing*, *Fuzzy Sets and Systems*, and *Symmetry*. He is also an active member of the Pakistan Information Security Association (PISA), the National Academy of Young Scientists (NAYS) Pakistan, the British Science Association for the Advancement of Science, the Center of Excellence & Information Assurance (CoEIA) Kingdom of Saudi Arabia, the International Association of Computer Science & IT (IACSIT), and the International Association of Engineers (Computer Science). In addition to teaching and research, he has also completed professional trainings on executing cyber-crimes and laws, digital forensics, e-mail tracing, electronic crime scene investigation, communication and leadership, and immigration & human trafficking. He is a recipient of the Exceptional Research Productivity Awards 2014, 2015, 2016, 2017, 2018, and the Best Researcher of the Year Awards 2012, 2013, 2015, 2017, and 2018 from KAU, Saudi Arabia, for his outstanding research performance. He has a distinguished career in education, research, and administration, having more than 12 years of technical and research experience globally.

MUHAMMAD BILAL QURESHI is currently working as an Assistant Professor with SZABIST Islamabad, Pakistan. He is a recipient of many prestigious awards, including the Gold Medal in undergrad degree, the Higher Education Commission Pakistan Indigenous Scholarship for M.S. and Ph.D. studies, and the Research Productivity Awards 2014 and 2015, respectively. He is the author of many research publications in SCIE journals, including the IEEE Access, the *Journal of Grid Computing*, *Parallel Computing*, *Parallel and Distributed Computing*, *The Journal of Supercomputing*, and *Sustainable Cities and Society*. His research interests include data-intensive real-time systems, resource allocation problems in HPC systems, and energy-efficient IoT.

AYAZ ALI KHAN received the Ph.D. degree from the Department of Computer Science, Abdul Wali Khan University Mardan (AWKUM), Pakistan, and the M.Phil. (M.S.) degree in computer science from the COMSATS Institute of Information Technology (CIIT), Islamabad, Pakistan. His area of research includes energy-aware and performance-efficient scheduling, resource allocation, placement and resource management, and datacenter level. Moreover, he has enough knowledge of distributed systems, optimization algorithms, machine learning, game theory, and computer programming.

MUHAMMAD ZAKARYA (Member, IEEE) received the Ph.D. degree in computer science from the University of Surrey, Guildford, U.K. He is currently a Lecturer with the Department of Computer Science, Abdul Wali Khan University Mardan, Pakistan. His research interests include cloud computing, mobile edge clouds, performance, energy efficiency, algorithms, and resource management. He has a deep understanding of theoretical computer science and data analysis. Furthermore, he also owns a deep understanding of various statistical techniques which are, largely, used in applied research. He is an Associate Editor for the IEEE Access Journal and has served as a TPC member in various international level conferences and workshops.

MUHAMMAD FAYAZ received the Ph.D. degree in computer engineering from Jeju National University (JNU), Jeju, South Korea, in 2019. During his studies, he worked at the Mobile Computing Lab, JNU. During his stay in South Korea, he also worked on several projects on the underground system and the SAR-based big data classification and analysis of disaster/damage type, funded by the Electronics and Telecommunication Research Institute, and Korean Aerospace Research Institute, respectively. He is currently an Assistant Professor in computer science with the UCA's School of Arts and Sciences. Throughout his academic career, he has published over 40 research papers in the reputed Web of Science and Scopus indexed international journals and conferences. His research interests include combinatorial optimization problems, machine learning, image processing, the Internet of Things (IoT), fuzzy inference systems, and other related areas.