Classification-based and Energy-Efficient Dynamic Task Scheduling Scheme for Virtualized Cloud Data Center

Avinab Marahatta[a,b], Sandeep Pirbhulal[c], Fa Zhang[b], Reza M. Parizi[d], Kim-Kwang Raymond Choo[e,*], Zhiyong Liu[b,*]

Abstract—The size and number of cloud data centers (CDCs) have grown rapidly with the increasing popularity of cloud computing and high-performance computing. This has the unintended consequences of creating new challenges due to inefficient use of resources and high energy consumption. Hence, this necessitates the need to maximize resource utilization and ensure energy efficiency in CDCs. One viable approach to achieve energy efficiency and resource utilization in CDC is task scheduling. While several task scheduling approaches have been proposed in the literature, there appears to be a lack of classification-based merging concept for real-time tasks in these existing approaches. Thus, an energy-efficient dynamic scheduling scheme (EDS) of real-time tasks for virtualized CDC is presented in this paper. In the scheduling scheme, the heterogeneous tasks and virtual machines are first classified based on a historical scheduling record. Then, similar type of tasks are merged and scheduled to maximally utilize an operational state of the host. In addition, energy efficiencies and optimal operating frequencies of heterogeneous physical hosts are employed to attain energy preservation while creating and deleting the virtual machines. Experimental results show that, in comparison with existing techniques, EDS significantly improves overall scheduling performance, achieves a higher CDC resource utilization, increases task guarantee ratio, minimizes the mean response time, and reduces energy consumption.

Index Terms—Cloud data center, virtualization, energy efficiency, task scheduling, task merging, virtual machine.

1 Introduction

Cloud data centers (CDCs) are a collection of interconnected and virtualized heterogeneous resources (including hosts, storages, applications, networks, and services), which facilitate ubiquitous, on-request network access to (shared) computing resources [1], [2], [3]. CDCs generally host large number of physical hosts, for example hundreds of thousands of physical hosts. The scale of such operations, such as those related to the hosts and equipments, also incurs significant energy costs [4], [5]. This clearly has environmental implications. For example, CDCs are estimated to contribute 2% of the overall carbon emission [6]. There are also additional energy-related costs [7], which can have an implication on the reliability of hosts' performance [8]. In other words, we are facing an environmentally unsustainable situation [6]. This necessitates the design of energy consumption reduction or "green" CDCs.

Authors Affiliations:
[a] *University of Chinese Academy of Sciences, China*
[b] *High Performance Computer Research Center, Institute of Computing Technology, Chinese Academy of Sciences, China*
[c] *CAS Key Laboratory of Human-Machine Intelligence -Synergy Systems, Shenzhen Institute of Advanced Technology, Chinese Academy of Sciences, China*
[d] *Department of Software Engineering and Game Development, Kennesaw State University, Marietta, GA 30060, USA*
[e] *Department of Information Systems and Cyber Security, University of Texas at San Antonio, San Antonio, TX 78249, USA*
Authors E-mail: avinab.marahatta@ict.ac.cn, sandeep@siat.ac.cn, zhangfa@ict.ac.cn, rparizi1@kennesaw.edu, raymond.choo@fulbrightmail.org, zyliu@ict.ac.cn
[*] *Corresponding authors.*

In parallel to the excessive energy consumption, CDCs also face other challenges such as (very) low resource utilization. For example, it was estimated that completely idle hosts consume approximately 50% additional energy in comparison to fully utilized hosts [9]. In other words, lower utilization of computing resources (estimated to be between 10% and 50%) indicates the massive waste of energy [10], [11]. Therefore, improving resource utilization by decreasing the number of on-run hosts, for example via dynamic arrangement of virtual machines (VMs) on a host, is one potentially viable strategy to minimize energy consumption [12]. In practice, most applications in CDCs are heterogeneous submitted dynamically by customers, with defined deadlines. In other words, we need to have sufficient number of active hosts in the CDC at any time to execute incoming or upcoming tasks successfully prior to their deadlines. Failing would affect energy consumption and violate the Quality of Service (QoS), which has significant financial and reputation implications.

In this paper, we design an energy-efficient dynamic scheduling scheme (EDS), to achieve energy-efficiency, and optimize task guarantee ratio, mean response time, and resource utilization for CDCs. More precisely in our EDS, tasks are classified using task classification method and then mapped to the most suitable VMs based on the task's requirement and the host's capacity. EDS uses task classification based on Bayes classifier and historical scheduling record (HSR), which allow one to find both task type and VM type. The task merging strategy is developed to merge similar type of tasks and schedule the most suitable VM until the resource condition is satisfied. Therefore, the sched-

uler can dynamically scale up and consolidate the resources of CDC, according to resource requirement.

A summary of our contributions in this paper is as follows:

- The design of a dynamic task scheduling scheme with resource provisioning in the context of virtualized CDC.
- The design of a task classification method to classify tasks and map them to the most suitable VM, in order to minimize mean response time.
- The design of a task merging strategy to merge similar type of tasks, when are then scheduled in the same physical host to maximize utilization ratio.
- The design of policies to create, migrate and delete VMs dynamically to adjust the scale of CDC and satisfy real-time requirements.

In the next two sections, we will present related literature and relevant mathematical background materials. In Sections 4 and 5, we present our proposed approach and describe our evaluation of the proposed approached. Specifically, the proposed approach is simulated using CloudSim toolkit, and we benchmark the performance of the proposed approach with existing schemes, in terms of task guarantee ratio, resource utilization, total energy consumption, and mean response time. Section 6 presents the conclusion.

2 Related Work

Topics relating to minimizing/ optimizing energy consumption and scheduling for CDCs are active research areas, since the late 2000's [13], [14], [15]. For example, there have been extensive research efforts on designing energy efficient scheduling approaches, such as virtualization-based energy-efficient approaches and dynamic voltage and frequency scaling (DVFS) enabled approaches, for both homogeneous and heterogeneous environments [16], [17], [18]. The survey of energy efficient strategies for data centers presented in [17], for example, classified existing research efforts into those focusing on operating system, hardware, data center and virtualization. A multi-dimensional partition model based on VM placement technique was proposed in [19]. In a similar work, a greedy algorithm was proposed to deal with quadratic assignment and bin packing problems to improve utilization of resources by reducing the number of active hosts [20].

Energy-aware Resource Allocation Method (EnReal), proposed in [21], focuses on VMs' dynamic deployment for scientific workflow executions. The authors proposed an energy consumption model to study cloud application requirements, and an energy-aware resource allocation method for VM allocation; thus, supporting scientific workflow execution based on their energy consumption model.

A balanced VM Scheduling method for energy-performance trade-offs in cyber-physical cloud system is presented in [22]. In this particular work, a joint optimization model for energy consumption and performance degradation of VM migration for cyber-physical cloud systems was formulated. Using the model, the authors designed a corresponding VM scheduling method for trade-offs between energy and performance to achieve both energy saving and performance degradation mitigation.

Several algorithms have been proposed for VM dynamic consolidation in CDCs, in order to optimize the number of hosts by consolidating VMs dynamically and powering off idle hosts to decrease additional energy consumption [23], [24], [25]. The main objective of these algorithms is mainly to improve the utilization of computing resources and reduction of energy consumption under existing service level agreement (SLA) constraints. Overloading host detection, power and SLA-aware VM selection, and two-phase VM placement algorithms are developed using an iterative weighted linear regression method to determine two utilization thresholds and avoid performance degradation [24]. These algorithms can be utilized for the effective placement of new VMs, where the selected VMs can be further processed for consolidation. Similarly, an enhancing energy-efficient and QoS dynamic virtual machine consolidation (EQVC) method is proposed in [25], which consists of four algorithms that correspond to different stages in VM consolidation. Based on their method, redundant VMs from the hosts are selected, and then they are migrated to other hosts before they get overloaded. In this work, the host-model with adaptive reserved resources were further introduced to prevent re-overload of hosts, guaranteeing QoS requirements.

A parameter-based VM consolidation solution is proposed in [26], which aims to mitigate issues with reservation-based and demand-based solutions. This parameter-based VM consolidation exploits the range between the demand-based and reservation-based ways of finding VM to host allocations that strike a delicate balance according to cloud providers' goals. In another work, a self-adaptive approach, called SAVE, is proposed by [27]. This approach makes the decision of the assignment and migration of VMs by using probabilistic processes that exclusively use local information.

A workload prediction-based reconfiguration algorithm is proposed to allocate VMs and hosts dynamically [28]. Similarly, a scheduling algorithm is proposed in [29] to select the most suitable physical host to a VM, based on the future power consumption's forecast. An online scheduling system is also developed by [30] for distributed computing platforms to minimize the energy consumption.

The brownout enabled system considering application components is presented in [31], which are either mandatory or optional. The authors consider component-level control in the system that can also be applied to container or micro services architecture. In this work, they proposed an algorithm to determine brownout time and reduced utilization of host. Their results showed that the application utilization and total energy consumption could be reduced.

In another work, a utilization-based migration algorithm is proposed to minimize the energy consumption of host by decreasing the migrations [32]. The authors developed a performance function for utilization-based migration scheme to optimize the VM placement, which, in turn, consolidates VMs to surpass the energy efficiency and guarantee the QoS. Experimental results showed about 10% of hosts has low utilizations.

A multi-objective dynamic scheduling system is proposed to minimize the energy consumption [30]. A real-time system that combines execution time and energy consump-

tion to execute task-based applications on cloud computing platforms based on energy performance important factor. A prototype of scheduler has been implemented and tested with real task-based COMPSs applications. The system is evaluated with different kind of DAG and size instances which provides a better solution.

Energy-aware resource provisioning mechanism is proposed in [33] for cloud data centers, which are capable of serving real-time periodic tasks following the Software as Service model. The three-tier energy-aware hierarchical scheduling approach is presented, where the scheduler of the guest OS constitutes the first tier, the placer component of the hypervisor located on top of servers forms the second tier, and the third tier is a server-level scheduler that allocates the VCPUs onto physical CPUs (PCPUs)within each multi-core server. Similarly, an online energy-aware resource provisioning algorithm (EAICA) for real-time cloud services is proposed in [34] to reduce the deadline miss ratio for real-time cloud services. The proposed provisioning framework not only considers the energy consumption of servers but it also takes the energy consumption of the communication network into account, to provide a holistic solution. EAICA is inspired from a swarm intelligence technique based on the Imperialist Competitive Algorithm (ICA).

A migration technique is used to move VMs from underutilized hosts to utilized hosts and makes under-utilized hosts idle; then idle servers get powered off [35], [36], [37]. From a critical point of view, migration techniques could bring delay with extra energy consumption. Maximum (rate) Utilization (MaxUtil) has compared two consolidation approaches which added migration techniques to former approaches aiming at maximizing resource utilization [38]. In MaxUtil's system, neither adaptive nor predictive techniques have been proposed. A power utility based energy-aware scheduling (EAS) algorithm is proposed in [39] to reduce the energy consume by workflow. The proposed EAS algorithm covered two sub-algorithms named as task merging algorithm and task mapping algorithm. First, all tasks (merged and non-merged task) are mapped to an optimal VM-type based on the power utility concept; then second, two tasks from task sequence are merged as one.

An algorithm based on Dynamic Voltage and Frequency Scaling (DVFS) technology has been proposed to reduce the amount of energy consumed by parallel jobs [40]. This algorithm widens the non-critical task's execution time and reduces energy cost by lowering the frequency and voltage of processor. However, the given algorithm does not examine parallel scheduling operation for other resource demands in addition to the CPU. An energy-aware scheduling for real-time tasks based on cooperative two-tier strategy was also proposed by [41], which aimed to be beneficial to both cloud users and their service providers. Similarly, a new task scheduler for cloud computing has been proposed in [42] by focusing on dependency between tasks and aiming to achieve energy savings. A cloud-based energy consumption model was presented in [43] to improve the energy efficiency based on a statistical method.

3 Problem Formulation and Modeling

This section formulates a complete scheduling model for CDC to define the system model, resource model, task model, energy consumption model, optimization problems and constraints, task and virtual machine mark model, task merging model and migration model. Table 1 shows the key terms and their descriptions, which have been used in the rest of the paper.

3.1 System Model

The scheduler consists of task observer, historical scheduling record (HSR), VMs data, resource monitor, and resource allocator. When the user submit tasks, the tasks join the queue of the entire system in descending order according to deadlines [30]. Upon the arrival of a task in scheduling process, the task observer classifies the task and determines the types of VM. The complete task classification process is given in Algorithm 1 (Section 4), which is based on Bayes classifier. A merging queue is generated when the task observer finalizes the task types and VM types from the HSR. In case of no similarity, new task types and VM types are created. If the VMs show any inability to satisfy scheduling requirements, such as deadline, resource requirements (CPU, RAM, Network Bandwidth, and Disk Storage), the resource allocator will scale up resources. The resource monitor checks the status of hosts in terms of capability to see whether it can accommodate the first and second task of the same type or not. The task merger prefers the completion of merging similar types of tasks together based on satisfactory conditions including deadline, resource requirements, and energy optimization until the same type of tasks are all processed in the merging queue. Furthermore, the hosts report their execution status of scheduling a task and its resource utilization to the scheduler directly. The resource allocator and resource monitor manage the status of all the physical hosts in CDC. In addition, if the host is underloaded, it may lead to having some VMs stay idle for a long period of time. In this scenario, the resource allocator decides which VMs should be migrated to other hosts to boost resource utilization by scaling down resources. The whole system diagram is shown in Fig. 1.

3.2 Resource Model

A CDC consists of unlimited set of hosts $H = \{H_1, H_2, \cdot \cdot \cdot, H_i\}$, providing the physical infrastructure for creating virtualized resources to satisfy user's requirements. An active host set $H^{active} = \{H_1^{active}, H_2^{active}, \cdot \cdot \cdot, H_i^{active}\}; H_i^{active} \subseteq H_i$ is characterized by $H_i = ((f_i, \vartheta i), c_i, m_i)$, where$(f_i, \vartheta i) = \{(f_i^1, \vartheta i^1), \cdot \cdot \cdot, (f_i^{max}, \vartheta i^{max})\}$ is discrete pairs of frequency and voltage of H_i, c_i represents the CPU capability, m_i represents the memory capacity, which are computed. A set of VM on H_i can be modeled as $V_{ij} = \{V_j^\tau, V_j^\varphi\}$, where V_j^τ represents VM types defined in Section 3.6, and V_j^φ is VM requirement which is modeled as $V_j^\varphi = \{V_j^p, V_j^m, V_j^n, V_j^s\}$, where V_j^p, V_j^m, V_j^n and V_j^s represent the parameters of processor, memory, network bandwidth and disk storage of VM, respectively.

TABLE 1: Key terms and descriptions

Terms	Descriptions
H	The set of hosts $H = \{H_1, H_2, \cdots, H_i\}$
H^{active}	The set of active hosts $H^{active} = \{H_1^{active}, H_2^{active}, \cdots, H_i^{active}\}$
H^{off}	The set of powered off hosts $H^{off} = \{H_1^{off}, H_2^{off}, \cdots, H_i^{off}\}$
H^{Idle}	The set of idle hosts $H^{Idle} = \{H_1^{Idle}, H_2^{Idle}, \cdots, H_i^{Idle}\}$
V	The set of virtual machine $V = \{V_1, V_2, \cdots, V_j\}$
V_j^{φ}	Virtual machine resource requirement $V_j^{\varphi} = \{V_j^p, V_j^m, V_j^n, V_j^s\}$
$\mathbb{T}$	The set of tasks $\mathbb{T} = \{\mathbb{T}_1, \mathbb{T}_2, \mathbb{T}_3, \cdots, \mathbb{T}_k\}$
$\mathbb{T}_k^{\varphi}$	Resource requirement for task $\mathbb{T}_k$ to execute successfully $\mathbb{T}_k^{\varphi} = \{\mathbb{T}_k^p, \mathbb{T}_k^m, \mathbb{T}_k^n, \mathbb{T}_k^s\}$
t_k^a, t_k^l, t_k^d	Task arrival time, task work volume, task deadline
P_{total}	Total energy consume by the host
ϑ_i^d	Supply voltage of hosts H_i
f_i^d	Frequency of H_i
P_{static}	Static power
$P_{dynamic}$	Dynamic power
E_i^{total}	Total energy consumption
S_i^t	Status of running host H_i at time t
t_i^s, t_i^f	Start time, Finish time of host H_i
Υ_i	Power ratio for the idle host H_i^{Idle}
f_i^{max}	H_i's maximal frequency
P_i^{max}	Maximum power consumed by H_i
α_i	The proportionality coefficient of H_i
t_k^e	Execution time of task $\mathbb{T}_k$ at time t
ξ	Computation size of the task $\mathbb{T}_k$ having f_i^d of H_i
f_i^o	Optimal frequency of H_i
R_f^P	Ratio of power frequency
$H_{\mathbb{T}}$	Historical task set that is extracted from HSR
$H_{\mathbb{T}}^{\zeta}$	Historical task type count
$P(H_{\mathbb{T}}^{\zeta})$	Probability of task type count
$\mathbb{T}_k^{\tau}$	Task $\mathbb{T}_k$ of type τ
V_j^{τ}	Virtual machine V_j of type τ
m_n	Distance from one task to another task of same type.
t^{Idle}	Threshold time for idle virtual machines
$t_{V_{ij}}^{Idle}$	Idle time of virtual machine V_j running on host H_i

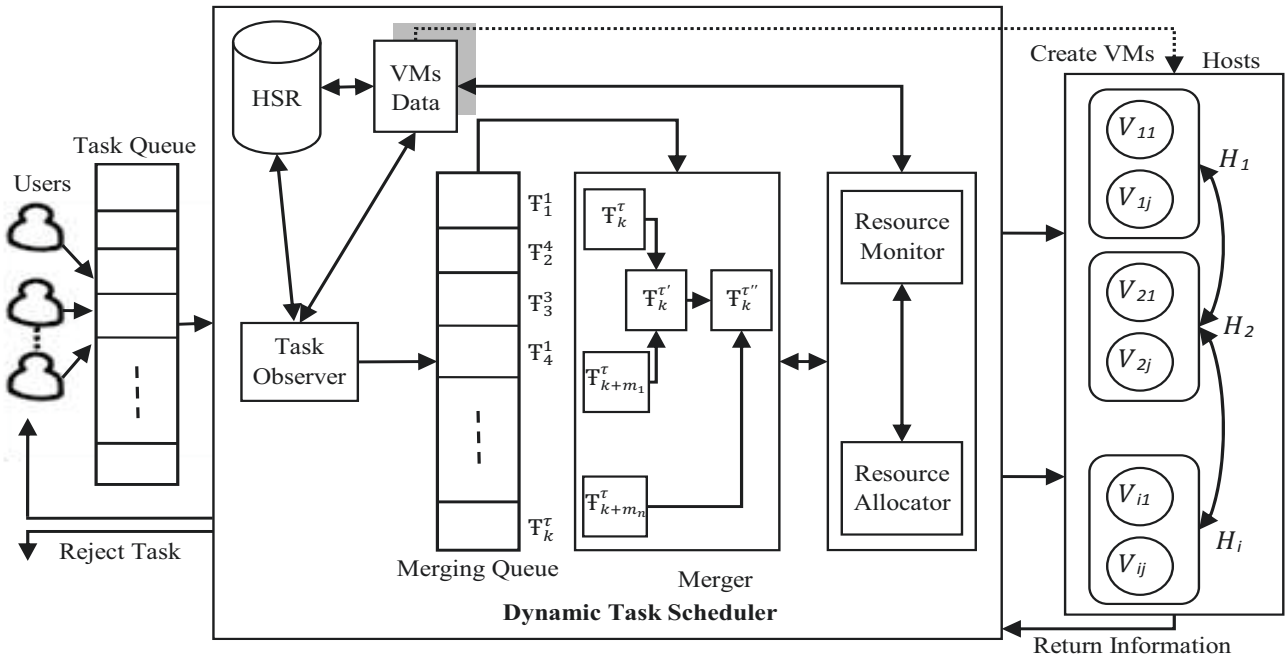

Fig. 1: System diagram

3.3 Task Model

The task is a medium or container that expresses or holds requirements of users. It can be observed that there are primarily two types of task. The first one is a computing task which requires high CPU and low memory, and the second task that needs low CPU and high memory. Here, we have defined a set of task as $\mathbb{T} = \{\mathbb{T}_1, \mathbb{T}_2, \mathbb{T}_3, \cdots, \mathbb{T}_k\}$ which includes a set of parameter $\mathbb{T}_k^{\varphi} = \{\mathbb{T}_k^p, \mathbb{T}_k^m, \mathbb{T}_k^n, \mathbb{T}_k^s\}$, where $\mathbb{T}_k^p$, $\mathbb{T}_k^m$, $\mathbb{T}_k^n$ and $\mathbb{T}_k^s$ represent the processing capacity, primary memory, network bandwidth and disk storage, respectively, and they are needed to execute a given task set. Similarly, $H_{\mathbb{T}}$, $\mathbb{T}_k^{\tau}$, $\mathbb{T}_k^{\zeta}$ represent the historical task record, task type and type count, respectively. In CDC environments, service providers get independent real-time tasks submitted by end-user in random time frame, and then the system accepts the service without understanding the complexity of computing infrastructure. An independent real-time task can be modeled as $\mathbb{T}_k = (t_k^a, t_k^l, t_k^d)$, where t_k^a, t_k^l and t_k^d represent the task arrival time, task work volume (i.e., total computing units), and task deadline, respectively. Cloud providers specify task parameters when users submit their tasks to them.

3.4 Energy Consumption Model

Generally, there are several components that consume energy, including hosts, electrical and network components, cooling system, storage devices. The main cause of energy consumption in a CDC is hosts [23], which mainly are determined by CPU, RAM, and disk storage. The energy consumed by H_i is categorized into static power (P_{static}) and dynamic power $(P_{dynamic})$, where idle host consumes approximately 70% of the energy consumed by the host running at the full CPU speed [23]. The total consumed power is the summation of both static and dynamic powers as shown below.

$$P_{total} = P_{static} + P_{dynamic} \tag{1}$$

This paper mainly focuses on the dynamic power, $P_{dynamic}$ of CPU like [8], which is directly proportional to supply voltage squared $(\vartheta_i^d)^2$ and its frequency f_i^d. Let α_i be a coefficient of proportionality, $P_{dynamic}$ of H_i which can be presented as follows, where ϑ_i^d and f_i^d is proportional to the $P_{dynamic}$ [23].

$$P_{dynamic} = \alpha_i \cdot (\vartheta_i^d)^2 \cdot f_i^d = \alpha_i \cdot (f_i^d)^3 \tag{2}$$

Let, Υ_i be the power ratio for the idle host H_i^{Idle}, f_i^{max} is H_i 's maximal frequency and P_i^{max} is the maximum power consumed by H_i. The proportionality coefficient α_i of H_i can be computed as

$$\alpha_i = \frac{(1-\Upsilon_i) \cdot P_i^{max}}{(f_i^{max})^3} \tag{3}$$

The dynamic power $P_{dynamic}$ can be computed from equations (2) and (3) as follows:

$$P_{dynamic} = \frac{(1-\Upsilon_i) \cdot P_i^{max}}{(f_i^{max})^3} \cdot (f_i^d)^3 \tag{4}$$

So, the power of H_i can be expressed as

$$P_i = P_{static} + P_{dynamic} = \Upsilon_i \cdot P_i^{max} \cdot S_i^t + \frac{(1-\Upsilon_i) \cdot P_i^{max}}{(f_i^{max})^3} \cdot (f_i^d)^3 \tag{5}$$

where, $S_i^t \subseteq (1, 0)$ indicates the status of running H_i at t, where S_i^t is 1 if H_i is active, and is 0, otherwise at time t. The total energy consumption E_i^{total} of H_i from start time t_i^s to finish time t_i^f can be approximated as follows.

$$E_i^{total} = \int_{t_i^s}^{t_i^f} (\Upsilon_i \cdot P_i^{max} \cdot S_i^t + \frac{(1-\Upsilon_i) \cdot P_i^{max}}{(f_i^{max})^3} \cdot (f_i^d)^3) dt \tag{6}$$

Therefore, the total energy consumption of physical hosts can be determined as follows.

$$E^{total} = \sum_{i=1}^{n} E_i^{total} \tag{7}$$

3.5 Optimization Problems and Constraints

Let t_{kij}^s and t_{kij}^f of $\mathbb{T}_k$ be the start time and finish time on V_{ij} of H_i, and t_k^e be the $\mathbb{T}_k$ execution time, then the finish time t_{kij}^f of $\mathbb{T}_k$ on V_{ijk} of H_i can be calculated as follows.

$$t_{kij}^f = t_{kij}^s + t_k^e \tag{8}$$

The parameter X_{kij} is used to represent the mapping relation between $\mathbb{T}_k$ and V_j on H_i. The assignment variable X_{kij} is 1 when $\mathbb{T}_k$ is scheduled to V_j, otherwise, X_{kij} is 0. This is formally represented as follows.

$$X_{kij} = \begin{cases} 1, & if\ \mathbb{T}_k\ assigned\ to\ V_{ij} \\ 0, & Otherwise. \end{cases}$$

The optimization problem can be formulated with aforementioned analysis as:

1) Maximum Guarantee Ratio = $\sum_{k=1}^{m} \sum_{i=1}^{n} \sum_{j=1}^{|V_{ij}|} \frac{X_{kij}}{m}$

2) Minimum Total energy Consumption= $\sum_{i=1}^{n} \int_{t_i^s}^{t_i^f} (\Upsilon_i \cdot P_i^{max} \cdot S_i^t + \frac{(1-\Upsilon_i) \cdot P_i^{max}}{(f_i^{max})^3} \cdot (f_i^d)^3) dt$

3) Minimum Response Time= (Waiting Time + Execution Time + Migration Time)

4) Maximum Resource Utilization = $\left(\frac{\sum_{k=1}^{|\mathbb{T}|} \sum_{i=1}^{|H_i^{active}|} \sum_{j=1}^{|V_j|} t_k^l \cdot X_{kij}}{\sum_{i=1}^{|H_i^{active}|} C_i \cdot (t_i^f) - t_i^s}\right)$ (9)

3.5.1 Constraints

(a) $(f_i^d, \vartheta i^d) \subseteq (f_i, \vartheta i), \forall\ H_i \subseteq H$
(b) $\forall \mathbb{T}_k^{\tau} \subseteq \mathbb{T}_k, \forall \mathbb{T}_k \subseteq \mathbb{T}$
(c) $X_{kij} = 1, 0, \forall V_{ij\Upsilon_i} \subseteq V_j, \forall H_i \subseteq H$
(d) $\sum_{k=1}^{m} \sum_{i=1}^{n} \sum_{j=1}^{|V_{ij}|} X_{kij} \leq 1, \forall \mathbb{T}_k \subseteq \mathbb{T}$
(e) $t_{kij}^f \leq t_k^d$

Here, we consider four optimization objectives to minimize the total energy consumption and the mean response time, while maximizing the task guarantee ratio and the utilization ratio.

3.5.2 Energy Optimization

Suppose, ξ is the computation size of the task $\mathbb{T}_k$ having utilization of frequency f_i^d of the host H_i. Then, the execution time t_k^e can be calculated as the ratio of ξ and f_i^d, i.e., $t_k^e = \xi / f_i^d$. Thus, E_i^{total} of H_i can be expressed as follow.

$$E_i^{total} = \Upsilon_i \cdot P_i^{max} \cdot \frac{\xi}{f_i^d} + \frac{(1-\Upsilon_i) \cdot P_i^{max}}{(f_i^{max})^3} \cdot (f_i^d)^3 \cdot \frac{\xi}{f_i^d} \tag{10}$$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2019.2918226, IEEE Transactions on Cloud Computing

For optimization,

$$-\Upsilon_i \cdot P_i^{max} \cdot \frac{\xi}{(f_i^d)^2} + 2 \cdot \frac{(1-\Upsilon_i) \cdot P_i^{max}}{(f_i^{max})^3} \cdot f_i^d \cdot \xi = 0 \quad (11)$$

Based on the above equations, the optimal frequency of H_i can be calculated as

$$f_i^d = \sqrt[3]{\frac{\Upsilon_i}{2.(1-\Upsilon_i)}(f_i^{max})^3} \quad (12)$$

Optimal frequency (f_i^o) of H_i can execute in set of discrete frequency, i.e., $f_i^d \subseteq f_i = \{f_1^1, f_2^2, \cdots, f_i^{max}\}$, which can be defined with maximum power P_i^{max} in the following manner.
$f_i^o = f_i^1$, if $f_i^d \leq f_i^1$,
$f_i^o = |f_i^d|$, if $f_i^1 \leq f_i^d \leq f_i^{max}$, where $|f_i^d|$ represents the frequency that is more than f_i^o and nearest of f_i^{max},
$f_i^o = f_i^{max}$, if $f_i^d \geq f_i^{max}$.
The ratio of power frequency (R_f^P) can be computed as below.

$$R_f^P = \frac{f_i^o}{E_i} = \frac{f_i^o}{\Upsilon_i \cdot P_i^{max} + \frac{((1-\Upsilon_i) \cdot P_i^{max})}{(f_i^{max})^3} \cdot (f_i^o)^3} \quad (13)$$

The observed scenario indicates that the larger R_f^P of host denotes the higher energy efficiency in which case the system schedules to execute the real-time heterogeneous tasks when its workload decreases. This, in turn, results in powering off the host with lower R_f^P firstly.

3.6 Task and Virtual Machine Mark Model

The task and virtual machine mark model has two operations, i.e., categorizing randomly arriving tasks to define the best VM types for mapping with the most suitable VMs [44] [45], and obtaining historical data set from HSR to classify the tasks, and generating VM types. Let $H_{\mathbb{T}}$ be the historical task set that is extracted from HSR in CDC. Let $H_{\mathbb{T}}^{\zeta}$ represents the historical task type count, then the ratio $P(H_{\mathbb{T}}^{\zeta})$ is given as

$$P(H_{\mathbb{T}}^{\zeta}) = \frac{|H_{\mathbb{T}}^{\zeta}|}{|H_{\mathbb{T}}|} \quad (14)$$

Based on our resource model and task model, the matching degree $P(\mathbb{T}_k^{\varphi}|V_j^{\varphi})$ is calculated as

$$P(\mathbb{T}_k^{\varphi}|V_j^{\varphi}) = \begin{cases} (V_j^{\varphi}/\mathbb{T}_k^{\varphi})^2, If(\mathbb{T}_k^{\varphi} > V_j^{\varphi}) \\ (V_{max}^{\varphi} - V_j^{\varphi} + \mathbb{T}_k^{\varphi})/V_{max}^{\varphi}, Otherwise \end{cases} \quad (15)$$

where $V_{max}^{\varphi} = \cdot_{\tau \subseteq U}^{max} V_{\tau}^{\varphi}$.

For us suppose that $V_j^1 = 12$, $V_{\tau}^1 = 8$, $V_{max}^1 = 100$, and $\mathbb{T}_k^{\varphi} = 10$.

According to equation 15,
$P(\mathbb{T}_k^1|V_j^1) = (V_{max}^1 - V_j^1 + \mathbb{T}_k^1)/V_{max}^1 = 0.98$
$P(\mathbb{T}_k^1|V_{\tau}^1) = (V_{\tau}^1/V_{max}^1)^2 = 0.64$
$P(\mathbb{T}_k^1|V_{max}^1) = (V_{max}^1 - V_{max}^1 + \mathbb{T}_k^1)/V_{max}^1 = 0.1$

We can see in the above outcome, $0.98 > 0.64 > 0.1$. Note that if $P(\mathbb{T}_k^{\varphi}|V_j^{\varphi}) = 1$, it means we have a perfect match. Thus, the result is that $\mathbb{T}_k^1$ has a better match with V_j^1, rather than V_{τ}^1 and V_{max}^1. By using the Bayes Classifier [46], we can calculate the probability of $\mathbb{T}_k$ belonging to $\mathbb{T}_k^{\tau}$,

$$P(\mathbb{T}_k^{\varphi'}|V_j^{\varphi'}) = P(\mathbb{T}_k^p|V_j^p)P(\mathbb{T}_k^m|V_j^m) \quad (16)$$

Then, the decision function of $\mathbb{T}_k$ is

$$\{(j', k') = arg\,max\ P(\mathbb{T}_k^{\tau'}|P(\mathbb{T}_k^{\varphi'})\} \quad (17)$$

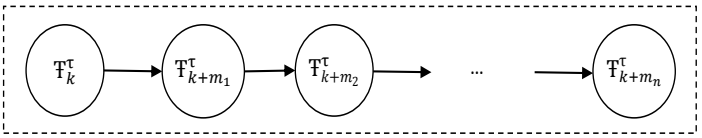

Fig. 2: Task series of same task types

3.7 Task Merging Model

We define two tasks sequence, the same type $\mathbb{T}_k^{\tau}$, and $\mathbb{T}_{k+m_n}^{\tau}$ the same VM types. Here, k represents the sequence number of task, and $m = \{m_1, m_2, \cdots, m_n\}$ represents the queue length from one task to another task of the same type, which is not necessarily has to be equal. The task series of the same task type is given as $\mathbb{T}_k^{\tau}$, $\mathbb{T}_{k+m_1}^{\tau}$, $\mathbb{T}_{k+m_2}^{\tau}$, $\cdots$, $\mathbb{T}_{k+m_n}^{\tau}$, as shown in Fig. 2.

The system keeps all tasks in merging queue which were generated by the task observer. The task observer categories the task based on HSR, then it maps them with the most suitable VM. The merging queue is generated by the task observer with task types and VM types. Merging queue is sorted with the decreasing order of deadline.

Let t_k^e, $t_{k+m_1}^e$, $t_{k+m_2}^e$, $\cdots$ be the execution time of task series of tasks of the same type, as shown in Fig. 3. When two tasks $\mathbb{T}_k^1$ and $\mathbb{T}_{k+m_1}^1$ get merged, then the execution time of merged tasks $\mathbb{T}_k^{'}$ is given as $\mathbb{T}_k^e = \mathbb{T}_{k+m_1}^e$. The deadline of task $\mathbb{T}_k^{'}$ will be t_k^d because tasks $\mathbb{T}_k^1$ must be finished before the deadline t_k^d.

Example. Majority of the applications in cloud environments are dynamically submitted by end users and are specified with deadlines to ensure their timeliness requirements in many domains, including scientific visualization, real-time signal processing and so forth . Suppose, $\mathbb{T}_k^1$, $\mathbb{T}_{k+m_1}^1$, $\mathbb{T}_{k+m_2}^1$, $\cdots$ are the task series of task type 1 from Merging Queue (MQ) and V_j^1, $V_{j+n_1}^1$, $V_{j+n_2}^1$, $\cdots$ are the VM series of the same type. The deadline of $\mathbb{T}_k^d$ is very near because MQ is sorted with decreasing order of deadline. First, if $\mathbb{T}_k^1$ can be schedule to V_j^1 with the deadline t_k^d, it checks the same task type successor $\mathbb{T}_{k+m_1}^1$. If $\mathbb{T}_k^1$ + $\mathbb{T}_{k+m_1}^1$ can finish its execution before t_k^d, it checks for another successor of the same type $\mathbb{T}_{k+m_2}^1$. If $\mathbb{T}_k^1$ + $\mathbb{T}_{k+m_1}^1$ + $\mathbb{T}_{k+m_2}^1$ is possible to be scheduled within t_k^d, the process continues until the same task type finishes. If $\mathbb{T}_k^1$ + $\mathbb{T}_{k+m_1}^1$ cannot be finished before t_k^d, it scales up resources and schedule. If $\mathbb{T}_k^1$ can not be scheduled within t_k^d, it scales up resources and check whether the scheduling is possible or not, if it is possible, it performs the scheduling, otherwise, it rejects the tasks.

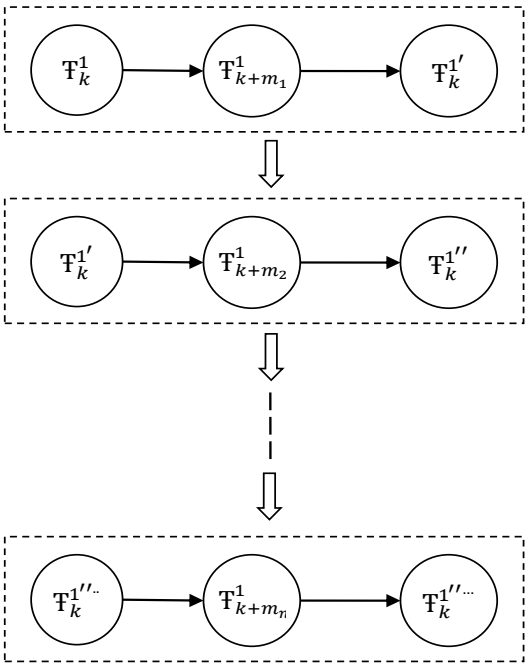

Fig. 3: The process of task merging

3.8 Migration Model

Migration is widely used in CDCs to optimize resource utilization at the host level for cloud resource management [45]. Basically there are two types of migrations, *offline migration*: it refers to moving a suspended VM from one host, and *live migration*: it refers to moving a running VM from one host to another.

In this paper, the live migration is solely considered due to two main reasons: firstly, it is more efficient from a performance point of view as it is able to transfer a VM between hosts with a close to zero downtime [47]. Secondly, it is the most adopted VM migration type in modern VM managers. The length of a live migration depends on the total CPU size, memory size, disk storage used by VM, and the available network bandwidth. During live migration, the CPU state context of the VMs are switched from the source host to destination host. As a result, there will be small data to be transferred, and represents the lowest limit for minimizing the migration time. Similarly, the VMs memory state also needs to be transferred to the destination host. This information may be greater than the CPU state, which includes the memory state of guest OS, and all the running processes within the VM.

Let f_i^{th} be the threshold frequency of hosts. If the host's resources are utilized to a very low extend (frequency of host (f_i^d) is less than or equal to threshold frequency (f_i^{th}) and greater than optimum frequency f_i^o, i.e., $f_i^o < f_i^d \leq f_i^{th}$), then a migration mechanism can be used to migrate running virtual machine from less utilized hosts to high utilized hosts (frequency of host (f_i^d) is greater than or equal to threshold frequency (f_i^{th}) and less then or equal to maximum frequency (f_i^{max}), i.e., $f_i^{th} < f_i^d \leq f_i^{max}$).

The VM migration time is very important factor for energy dissipation. Therefore, time must be considered before taking any migration-related decision. The total migration time, depends on the number of VMs to be migrated, and the size of VMs. However, the total number of VMs to be migrated, and the size of each VM cannot be known in advance except in a probabilistic sense [48]. The time cost of migrating of a single VM can be expressed in terms of the statistics of the CPU content size, RAM size, and storage size of this VM, as well as the available bandwidth at the time of migration. It is important to stress that the RAM size is the main parameter that affects the VM migration time. The migration time is given in equations (18) and (19).

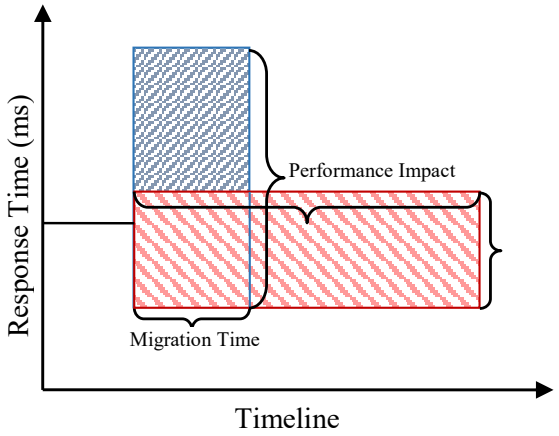

Fig. 4: The migration cost model

The VM Migration Time ($t_{V_j}^{Mig}$) can be computed as

$$t_{V_j}^{Mig} = (V_j^p + V_j^m)/V_j^n \tag{18}$$

where, V_j^n is the available bandwidth for VM V_j, in Bytes/Seconds between the source and destination hosts. The total VM Migration Time can be computed as

$$t_V^{Mig} = \sum_{j=1}^{|V^{Mig}|} t_{V_j}^{Mig} \tag{19}$$

In evaluating the total VM migration time t_V^{Mig}, the values of $t_{V_j}^{Mig}$ are statistically independent for all VMs, where V^{Mig} denotes the number of migrated VMs.

The energy consumed for VM migration can be computed by

$$E_{V_j}^{Mig} = P^{Mig} * t_{V_j}^{Mig} = P^{Mig} * (V_j^p + V_j^m)/V_j^n \tag{20}$$

where, P^{Mig} denotes a unit power consumption for migrating a VM.

Let $Cost^E$ be the energy cost per unit, $cost^P$ is the SLA violation fixed penalty value, ω_{V_j} is the portion of k task's served by H_i^{target} and λ_{H_i} is the average request rate for H_i^{target}. Fig. 4 indicates the migration cost model for the proposed system. Then, the migration cost, $Cost^{Mig}$, can be calculated using

$$Cost^{Mig} = Cost^E * P_{dynamic_i} \sum_{j=1}^{|V^{Mig}|} \phi_j + \sum_{j=1}^{|V^{Mig}|} cost_{ij}^P * \omega_{V_j} * \lambda_i * e^{-(((V_j^p+V_j^m)/V_j^n)*\phi_j*\mu_i-\lambda_i)} * t_{V_{ij}{}^{contract}} \tag{21}$$

4 Algorithm Design

4.1 Overview

Task scheduling and resource management are the striking means to optimize energy consumption for CDCs to ensure the timing requirements of tasks. The scheduling strategies are used to schedule real-time tasks dynamically. Scheduling process deals with various disruptions during system operation, including the arrival of new tasks or just-completed tasks. In such events, the workload of CDC becomes heavy and will lead to a scale-up situation to consolidate the available physical and virtual resources based on the requirements of tasks.

The main method energy-efficient dynamic scheduling EDS () is presented in Algorithm 2, which has four methods. MARKVM () is used to calculate various task and virtual machine types. Similarly, TASKMERGE () is used to merge same type of tasks. The SCALEUPRESOURCES() and CONSOLIDATION() methods are called multiple times when system dynamically scale up and scale down computing resources, respectively.

4.2 Description

When users submit their tasks $\mathbb{T}_k$, EDS gathers the information in Algorithm 2, which includes the arrival time t_k^a and deadline t_k^d. Then, these tasks join the Task Queue of the entire system according to the decreasing order of deadlines, and MARKVM () method (Algorithm 1) will be called. Algorithm 1 is based on historical scheduling records and Bayes classifier. It mainly consists of two parts. The first part is classifying the upcoming tasks, and second part is marking the virtual machines. Within Algorithm 1, lines 3-4 gather the information for tasks from historical scheduling records, and then count the number of task types. Task observer checks whether the task type for a given task is available or not. If the task type was found, it would mark the task type and VM types (lines 9-11), otherwise it would mark $\mathbb{T}_k$ as new $\mathbb{T}_k^\tau$ and V_j^τ, and update HSR (Line 13).

Algorithm 1 Task and virtual machine types

1: **procedure** MARKVM()
2: Input: Task $\mathbb{T}_k$, HSR
3: $H_{\mathbf{T}} \leftarrow$ Historical Task Set
4: $\zeta \leftarrow$ Task type count
5: **For** each task T_k in ζ
6: Compute $P(H_{\mathbf{T}_k})$
7: **End for**
8: $a \leftarrow$ Choose High $P(H_{\mathbf{T}_k})$
9: **For** each VM V_j in a
10: **If** V_j^τ found of $\mathbb{T}_k^\tau$ type **then**
11: $V_j^\tau \leftarrow$ VM types based on $P(H_{\mathbf{T}_k})$
12: **Else**
13: Create New $V_j\{V_j^p, V_j^m, V_j^n, V_j^s\}$, Create New V_j^τ, Update HSR
14: **End If**
15: **End For**
16: **end procedure**

In Algorithm 2, Line 6, the virtual machines are sorted based on the ratio of power frequency, R_f^P. If $V_j^\varphi > \mathbb{T}_k^\varphi$ is satisfied, all the tasks of $\mathbb{T}_k^\varphi$ type will be sequencing to Merging Queue (MQ)(Lines 8-11). Following this step, method TASKMERGING() will be invoked in Line 13.

Algorithm 2 Energy-efficient dynamic scheduling

1: **procedure** EDS()
2: Input: Task $\mathbb{T}_k$, a set of VM V_j, and set of host H_i
3: $V_j^\tau \leftarrow$ MARKVM()
4: $\tau \leftarrow$ Choose $\mathbb{T}_k^\tau$
5: Set $\delta \leftarrow 0$
6: Order VMs of V_j^τ types with descending order of R_f^P
7: **For** V_j^τ in list
8: **If** $V_j^{\varphi'} > \mathbb{T}_k^\varphi$ **then**
9: $P(\mathbb{T}_k^{\tau'}|\mathbb{T}_k^{\varphi'}) = (\mathbb{T}_k^p|V_j^p)(\mathbb{T}_k^m|V_j^m)$
10: $\delta \leftarrow 1$
11: MQ $\leftarrow$ List all the task of τ types
12: $\mathbb{T}_k^{\tau'} = \mathbb{T}_k^\tau, t_k^{e'} = t_k^e, t_k^{d'} = t_k^d$
13: TASKMERGING()
14: Schedule $\mathbb{T}_k^{\tau'}$ to V_{ij}, Update HSR, Remove all merged tasks from merging queue
15: **End If**
16: **Else**
17: SCALEUPRESOURCES()
18: **If** $V_j^\varphi > \mathbb{T}_k^\varphi$ **then**
19: Go to line 9
20: **End If**
21: **End If**
22: **End For**
23: **If** $\delta = 0$ **then**
24: $V_j^\tau \leftarrow$ VM types [Create new VM types V_j^τ], Update HSR
25: **If** $t_k^e \leq t_k^d$ **then**
26: TASKMERGING()
27: **Else**
28: Close Created VM
29: Reject $\mathbb{T}_k$
30: **End If**
31: **End If**
32: CONSOLIDATION()
33: **end procedure**

In the task merging method, as shown in Algorithm 3, similar type of tasks can be merged and scheduled to VM of the same VM type. We merge as many the same typed tasks as possible and assign them to a VM that can satisfy the resource requirement. If the execution time of tasks $\mathbb{T}_k^\tau$ and $\mathbb{T}_{k+m_1}^\tau$ can be scheduled within the deadline t_k^d and resource requirements are satisfied, two tasks $\mathbb{T}_k^\tau$ and $\mathbb{T}_{k+m_1}^\tau$ will be merged and scheduled to the VM, as shown in Fig. 3. Then, the new execution time and deadline for a merged task $\mathbb{T}_k^{\tau'}$ becomes $t_k^{e'} = t_k^e + t_{k+m_1}^e$, $t_k^{d'} = t_k^d$, respectively (Lines 4-7). Otherwise, the resources will be scaled up, and accordingly scheduled (Lines 9-11).

In Algorithm 4, while scaling up the resources, all the active hosts are sorted based on R_f^P (Lines 3-6). If the requirements of the task can be accommodated by active hosts, V_{ij} of V_j^τ is created and HSR is updated (Lines 6-8). Otherwise, a new host will be powered on from H_{List}^{off} (power off host list) based on the decreasing order of R_f^P,

Algorithm 3 Task merging

1: **procedure** TASKMERGING()
2: Input: List all the task of τ types $\mathbb{T}_k^{\tau'} = \mathbb{T}_k^{\tau}, t_k^{e'} = t_k^e$
3: **For** each $\mathbb{T}_k^{\tau}$ in MQ
4: Calculate the execution time of task $\mathbb{T}_{k+m_n}^{\tau}$
5: **If** $(t_k^{e'} + t_{k+m_n}^e) \leq t_k^d$ AND Satisfy resource requirements **then**
6: $\mathbb{T}_k^{\tau'} = \mathbb{T}_k^{\tau'} + \mathbb{T}_{k+m_n}^{\tau}, t_k^{e'} = t_k^{e'} + t_{k+m_n}^e$
7: **Else**
8: SCALEUPRESOURCES()
9: **If** $(t_k^{e'} + t_{k+m_n}^e) \leq t_k^d$ **then**
10: $\mathbb{T}_k^{\tau'} = \mathbb{T}_k^{\tau'} + \mathbb{T}_{k+m_n}^{\tau}, t_k^{e'} = t_k^{e'} + t_{k+m_n}^e$
11: **Else**
12: **Break**
13: **End If**
14: **End If**
15: **End For**
16: **end procedure**

Algorithm 4 Scale up resources

1: **procedure** SCALEUPRESOURCES()
2: Select a V_{ij} that can finish task $\mathbb{T}_k$ before task deadline
3: **If** V_{ij} != NULL **then**
4: $H_{List}^{active} \longleftarrow$ Active hosts
5: Sort H_{List}^{active} by R_f^P in decreasing order
6: **For** each H_i in H_{List}^{active}
7: **If** H_i can accommodate V_{ij} **then**
8: Create V_{ij} of V_j^{τ} on H_i^{target}, Update HSR
9: **Break**
10: **End If**
11: **End For**
12: **End If**
13: Select a VM V_{ij} that satisfy $V_j^{\varphi'} \geq \mathbb{T}_k^{\varphi'}$ and $t_k^e \leq t_k^d$
14: **If** V_{ij} != NULL, **then**
15: $H_{List}^{off} \leftarrow$ all the host being powered off
16: Sort H_{List}^{off} by R_f^P in decreasing order
17: Power on H_i from H_{List}^{off}, and Create new VM, $H_{List}^{active} \leftarrow H_i$, Update HSR
18: **End If**
19: **end procedure**

Algorithm 5 Scale down resources

1: **procedure** CONSOLIDATION()
2: $V_j^{active} \longleftarrow$ all the active VMs
3: **For** V_j in V_j^{active} list
4: **If** $t_{V_{ij}}^{Idle} \leq t^{Idle}$
5: Delete V_{ij}
6: **End If**
7: **End For**
8: $H_{List}^{active} \longleftarrow$ All the active hosts
9: List H_{List}^{active} by R_f^P in decreasing order
10: $a \longleftarrow$ Select all H_{List}^{active} having $f_i^d \leq f_i^{th}$
11: **For** each host H_i in a
12: **If** H_i is idle **then**
13: Power off H_i
14: **Else**
15: $b \longleftarrow$ Select all the VMs from H_i
16: **For** VM V_j in b
17: $H_i^{target} \longleftarrow$ Select all the active hosts having $f_i^d > f_i^{th}$ AND $f_i^d \leq f_i^{max}$
18: **For** each in H_i^{target}
19: **If** H_i^{target} can accommodate V_j **then**
20: Migrate V_j from H_i to H_i^{target}
21: **End If**
22: Break
23: **End For**
24: Remove V_j from H_i
25: **End For**
26: Power off H_i
27: **End For**
28: **end procedure**

then new V_j of V_j^{τ} will be created and HSR will be updated as well (Lines 14-17).

The workload of CDC naturally decreases if the arrival ratio of real-time tasks from end-users declines. When the idle time of VMs $t_{V_{ij}}^{Idle}$ are higher than the t^{Idle}, the consolidation method will apply to minimize the running VMs and hosts, as shown in Algorithm 5. If $t_{V_{ij}}^{Idle}$ of V_{ij} exceeds t^{Idle}, V_{ij} will be deleted (Lines 2-7), and if H_i^{active} for deleted V_{ij} becomes idle, H_i will be powered off (Lines 11-13). In addition, the virtual machines can be migrated from less utilized hosts (hosts having $f_i^d \geq f_i^o$ AND $f_i^d \leq f_i^{th}$) to high utilized hosts (hosts having $f_i^d > f_i^{th}$ AND $f_i^d \leq f_i^{max}$), and idle hosts H_i^{Idle} (hosts having $f_i^d < f_i^o$) are powered off to reduce energy consumed by hosts in the CDC (Lines 8-27).

Example. Suppose users submit their tasks, $\mathbb{T}_k$, to the cloud, for which these tasks have various deadline and heterogeneity. Scheduling scheme sequences these tasks to the task queue based on decreasing order of deadlines. A task set in the task queue is $\mathbb{T} = \{\mathbb{T}_1, \mathbb{T}_2, \mathbb{T}_3, \cdots, \mathbb{T}_k\}$. The task observer checks similar information in historical scheduling record and VMs data, such as task arrival ratio, deadline, execution time, virtual machine information (CPU, RAM, Network Bandwidth, Disk Storage), and resource requirements (CPU, RAM, Network Bandwidth, Disk Storage). As mentioned earlier, the task classification is performed based on Bayes classifier. If the historical scheduling record is found, the decision will be to sequence these tasks in the merging queue with task type. If the task observer is unable to find any historical scheduling information which matched with tasks, the decision will be to create a new virtual machine, mark it as new task type and virtual machine types, and update HSR. Assume $\mathbb{T}_1$ has task type 1 and VM type 1, $\mathbb{T}_2$ has task type 3 and VM type 3, $\mathbb{T}_3$ has task type 1 and VM type 1, $\mathbb{T}_4$ has task type 1 and VM type 1, as so on. Then, scheduler takes $\mathbb{T}_1^1$ ($\mathbb{T}_1$ with type 1) for the merge process. The scheduler sorts all the virtual machine of type 1 with descending order of R_f^P. Then, it checks whether V_1^1 resources can accommodate and schedule $\mathbb{T}_1^1$ within the deadline or not. If V_1^1 can accommodate $\mathbb{T}_3^1$, scheduler sees the next task of the same task type 1. It also checks again whether V_1^1 can accommodate both tasks or not. If both tasks can be

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2019.2918226, IEEE Transactions on Cloud Computing

accommodated, it will merge these two tasks and the deadline for merged task $\mathbb{T}_1^{1'}$ of $\mathbb{T}_1^1$ and $\mathbb{T}_3^1$ becomes the deadline of $\mathbb{T}_1^1$(shortest deadline). If both tasks cannot be accommodated by V_1^1, then the scheduler scales up resources and checks again whether V_1^1 can accommodate both tasks or not. If yes, the merging will take place based on the schedule of $\mathbb{T}_1^1$ and $\mathbb{T}_3^1$ to V_1^1. Otherwise, it just schedules only $\mathbb{T}_1^1$ to V_1^1.

4.3 Time Complexity Analysis

There are four methods running at different times during the scheduling scheme of Algorithm 2, i.e., the classification method described by Algorithm 1, the task merge method described by Algorithm 3, the Scale-up method that when some new host needs to be powered on in order to host new tasks described by Algorithm 4, and the consolidation method described by Algorithm 5.

Algorithm 1 has time complexity of $O(n)$, where n is the number of tasks, since each task needs to be checked and given a task type and a VM type according to its resource requirements. Let m denote the number of the active hosts, and the complexity of Algorithm 3 is $O(n * mlogm)$. This is because Algorithm 3 may call upon Algorithm 4 for each task in the merging queue, and the complexity of Algorithm 4 is $O(mlogm)$, since it needs to sort the active hosts in line 5 and the powered off hosts in line 16 of Algorithm 4.

After running for a period of time, consolidation needs to be performed, using Algorithm 5 Scale Down Resources, such that energy efficiency can be achieved. The hosts can be considered in three states. The first part is idle hosts, where they have nothing to do while still powered on; the second part is the less utilized hosts (i.e., they have very few tasks to execute, and the tasks can be migrated to the third part of hosts); and the third part of the host is the one that can accept newly migrated tasks. Let b be the number of VMs that need to be migrated from part two to part three. Since we need to find a suitable host in the third part of hosts for each VM in the second part of the hosts to migrate, the complexity of Algorithm 5 is $O(m * b)$.

The complexity of ETVMC [49] is $O(n * m * b)$, where n is the number of tasks, m is the number of hosts, and b is the number of VMs. Similarly, the complexity of CEVP [50] and UMA [32] are $O(n^2)$ and $O(n^2)$, respectively.

5 Experiments

This section describes the overall experimental setup, performance metrics, results, and analysis to evaluate the EDS. We compared the performance of EDS with ETVMC [49], CEVP [50], and UMA [32] algorithms as all of them are designed for energy optimization.

5.1 Environment Setup

Experimental environment includes CPU (Intel(R) Core(TM) i5-3230 M, 2.60 GHz), Memory (8.0 GB), Hard Disk (750 GB), Windows 10 operating system, NetBeansIDE, JDK 8.0 and CloudSim. Powered by CloudSim toolkit [51], our simulation carried out the dynamic scheduling process, the mechanism of merging similar type of tasks and energy-aware techniques.

TABLE 2: Simulated host specification

Host Types	CPU (MIPS)	Host Count
Type 1	1000	3360
Type 2	1500	2815
Type 3	2000	2325

TABLE 3: Task types with parameter

Task Types	CPU Demand (GHz)	Memory Demand (GB)
Type 1	0.01-0.09	0.003-0.016
Type 2	0.02-0.06	0.05-0.08
Type 3	0.05-0.09	0.003-0.009
Type 4	0.01-0.05	0.008-0.012
Type 5	0.15-0.17	0.03-0.10
Type 6	0.03-0.08	0.31-0.0.37
Type 7	0.01-0.05	0.006-0.009
Type 8	0.23-0.29	0.05-0.09

5.2 Performance Metrics

To simulate task's heterogeneity and dynamic nature in the CDC environment, we have used a total of 898,766 continuous tasks, and 8500 hosts. In the simulation, each host is modeled to have only one CPU core with having 1,000 MIPS, 1,500 MIPS, and 2,000 MIPS CPU performance. The host specifications are presented in Table 2. Based on our simulation setup, we have identified four performance metrics which can be computed by formula (9). These metrics include

1) Task guarantee ratio: The proportion of the number of tasks finished before their deadlines to the total number of tasks $\times$ 100%
2) Total energy Consumption: It represents the total energy consumed by the physical hosts while scheduling tasks.
3) Mean Response Time: Optimal time taken to response to user.
4) Resource Utilization: It represents the total utilization of resources while scheduling tasks.

5.3 Experimental Results and Analysis

Table 3 describes the CPU and memory demands for each type of tasks, and Table 4 describes the CPU and memory capacity of each type of VMs. It is important to note that, in the fiber optic network as well as hard disk provision in CDCs, the network bandwidth and the hard disk capacity are not critical elements, and hence they are typically treated in a way that they will have the perfect match with all the tasks. For this reason, the hard disk and the network bandwidth were not considered in our analysis.

TABLE 4: VM types with parameter

VM Types	CPU (GHz)	Memory (GB)
Type 1	0.09	0.016
Type 2	0.06	0.08
Type 3	0.09	0.009
Type 4	0.05	0.012
Type 5	0.17	0.10
Type 6	0.08	0.0.37
Type 7	0.05	0.009
Type 8	0.29	0.09

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2019.2918226, IEEE Transactions on Cloud Computing

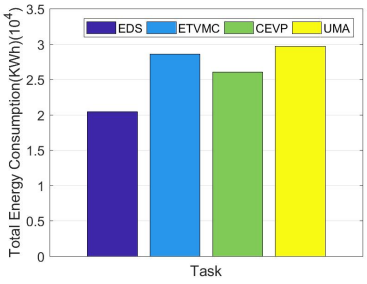

Fig. 5: Total energy consumption

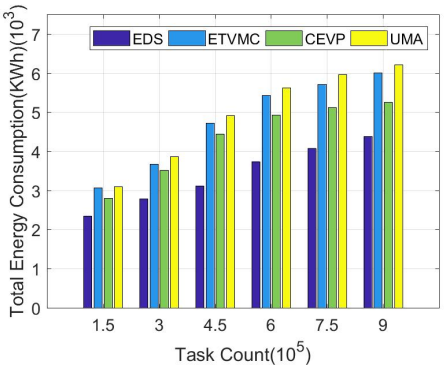

Fig. 6: Total energy consumption with task count

As mentioned earlier in Section 2, once again, the existing approaches for energy-efficient task scheduling in CDCs do not investigate the problems of combining different tasks of the same type in their underlying techniques. These approaches mainly focus on either a particular task type, or simply do not take into consideration a different kind of application that assumes a homogeneous task.

The metrics under study were measured and experiment results are shown in Fig. 5 to Fig. 15. A comparative summary of the total energy consumption of the proposed EDS and three peer algorithms is presented in Fig. 5, which shows that EDS has an optimal total energy consumption followed by CEVP, ETVMC, and UMA. Specifically, EDS saves about 28.49%, 21.48%, and 31.04% of energy compared to ETVMC, CEVP, and UMA, respectively. This performance is achieved because EDS uses a consolidation technique that performs simultaneous optimizations on the active number of hosts and the operating frequencies to minimize energy consumption.

Fig. 6 exhibits the total energy consumption of the four algorithms with respect to task count. As shown in the figure, the computation length of task required to be processed is linear to the number of tasks. Similarly, the total energy consumption of the CDC is nearly linear to the computation length of the system.

In every task count, EDS achieves better energy optimization. For example, as shown in Fig. 7, type 1 consumes the lowest energy while task count reaches 1.5×10^5. Similarly, type 1 and type 8 consume the highest energy while task count reaches 9×10^5.

Task guarantee ratios of the physical and virtual re-

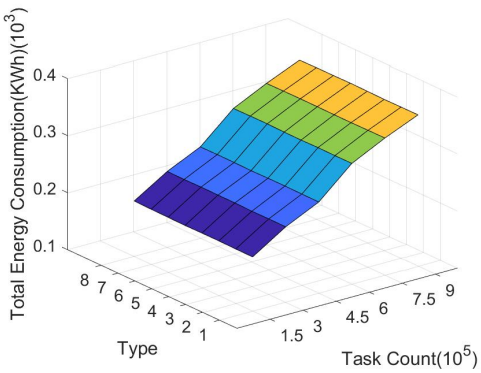

Fig. 7: Total energy consumption of task type with task count

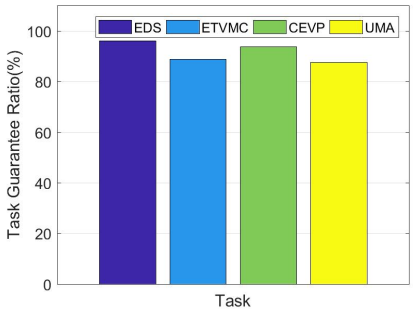

Fig. 8: Task guarantee ratio

sources of the four algorithms are shown in Fig. 8. The guarantee ratio for the EDS, ETVMC, CEVP, and UMA are stable in 96.12, 88.79, 93.78 and 87.57% with respect to the rise of task counts. Fig. 9 shows the task guarantee ratio with the task count of the four algorithms where the EDS shows to have a higher task guarantee ratio followed by CEVP, ETVMC and UMA. Similarly, Fig. 10 indicates the task guarantee ratio with the task count, in which type 6 has the highest guarantee ratio when task count reaches about 9×10^5.

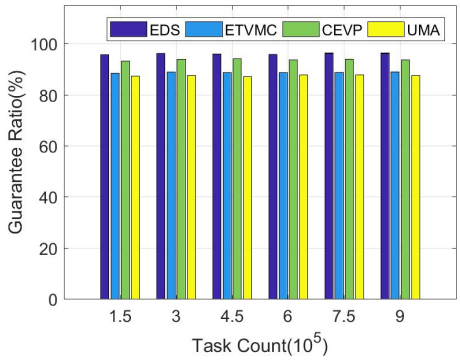

Fig. 9: Task guarantee ratio with task count

This particular observation can be associated to the fact that there are adequate computing resources in CDC. Hence, if the available resources can not accommodate the requirements of a task, accordingly more VMs will be created

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2019.2918226, IEEE Transactions on Cloud Computing

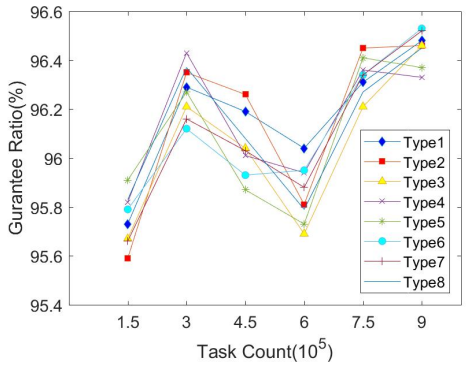

Fig. 10: Task guarantee ratio of task types with task count

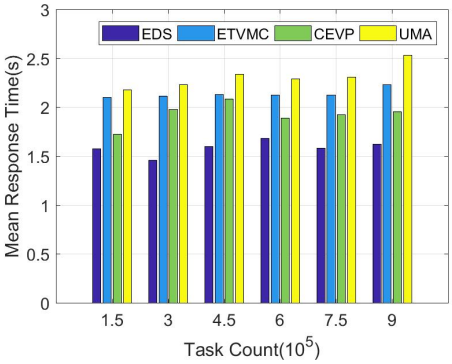

Fig. 12: Total mean response time with task count

and then added to run the workload in excess with extra hosts powered-on as necessary. On the other hand, the task guarantee ratio is not perfect as 100%. This phenomenon can be attributed to the delay of scaling up and consolidating resources.

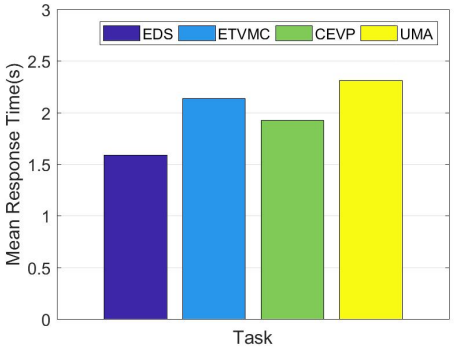

Fig. 11: Total mean response time

Fig. 11 indicates the mean response time of the four algorithms where the EDS apparently showed to have the optimal mean response time. The reason for this outperformance is that EDS benefits from a classification method to minimize iterative communication time for selecting the most suitable physical hosts and VMs. The total mean response time with the task count are shown in Fig. 12. EDS has the lowest mean response time when task count reaches 3×10^5.

Fig. 13 shows the mean response time of task types with different task counts. The mean response time of type 3 is lowest when the task count reaches 3×10^5. Similarly, the mean response time peaks when type 7 is at task count 6×10^5.

We also observed that the resource utilization of the four algorithms ascend accordingly. Fig. 14. shows that the EDS has a higher resource utilization compared to its peers. This may be due to the fact that EDS employs task merging strategies to allocate similar type of tasks to most suitable physical hosts and VMs, and it uses a consolidation technique to enhance host operating frequencies and the number of active hosts simultaneously that will result in reducing unnecessary resource wastage.

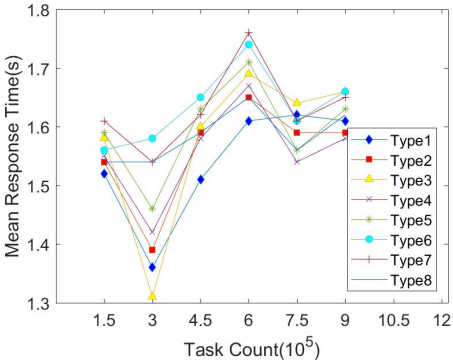

Fig. 13: Total mean response time of task types with task count

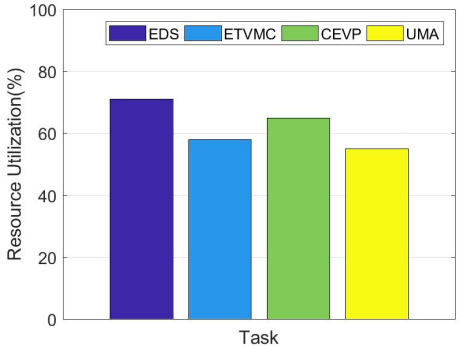

Fig. 14: Resource utilization

Fig. 15 illustrates the resource utilization percentages with the task count. As observed from the figure, the resource utilization is at a higher level when the task count reaches 7.5×10^5 and 9×10^5.

We will now provide a comparative summary of the waiting time of the proposed scheduling scheme and other existing algorithms. As shown in Fig. 16, EDS outperforms the other schemes, as their waiting times are linear to the number of tasks. The waiting time of EDS is lowest when the task count reaches 1.5×10^5 and 7.5×10^5. Similarly, the waiting time is at a higher level when the task count reaches 9×10^5.

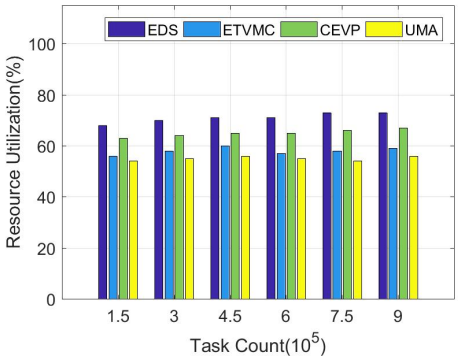

Fig. 15: Resource utilization with task count

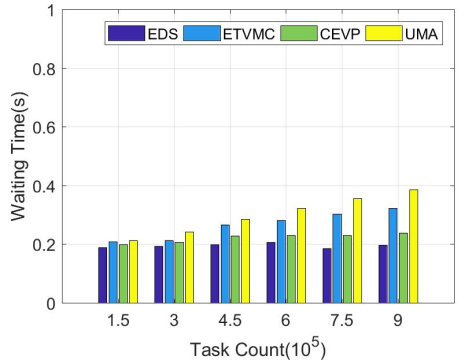

Fig. 16: Waiting time with task count

6 Conclusion and Future Work

Energy consumption and optimization are ongoing (operational) challenges faced by CDCs. In this research, the proposed EDS aims to optimize resource utilization and energy consumption in CDCs. In the EDS, the dynamic scheduling scheme makes use of HSR, classifies incoming tasks, creates VMs with different features, and assigns tasks in accordance with the matching features of the tasks and VMs; hence, the resource utilization ratio can be improved. The merging mechanism of same task types minimizes the mean response time and reduces total energy consumption. In addition, the scheme distributes and migrates tasks in accordance to the energy consumption function and the states of the computing resources (active or idle). The experimental results indicated that the EDS has a higher task guarantee ratio, shorter response time, higher resource utilization ratio, and a minimal energy consumption, in comparison to existing schemes.

In the future, we intend to investigate how task failure prediction based on machine learning approach, and task scheduling can be jointly optimized for better performance in the context of fault-tolerant mechanism.

Acknowledgments

This work is partially supported by the National Natural Science Foundation of China (grant numbers 61520106005, 61761136014), and the National Key Research and Development Program of China (grant number 2017YFB1010001). The first author gratefully acknowledges CAS-TWAS President's Fellowship for funding his Ph.D. at Chinese Academy of Sciences, Beijing, China.

References

[1] P. Mell and T. Grance, "The nist definition of cloud computing (draft)," 2011. [Online]. Available: http://csrc.nist.gov/publications/PubsSPs.html#800-145

[2] A. V. Dastjerdi, S. G. H. Tabatabaei, and R. Buyya, "A Dependency-Aware Ontology-Based Approach for Deploying Service Level Agreement Monitoring Services in Cloud," *Softw. - Pract. Exp.*, vol. 42, no. 7, pp. 501–518, 2011.

[3] Z. Shen, S. Subbiah, X. Gu, and J. Wilkes, "Cloudscale:elastic resource scaling for multi-tenant cloud systems," pp. 1–14, 2011. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2038916.2038921

[4] J. Koomey, "Growth in data center electricity use 2005 to 2010," *Oakland. CA Anal. Press*, 2011. [Online]. Available: https://www.missioncriticalmagazine.com/articles/84451-growth-in-data-center-electricity-use-2005-to-2010?v=preview

[5] K. Zhou, S. Yang, and Z. Shao, "Energy internet: The business perspective," *Appl. Energy*, vol. 178, pp. 212–222, 2016. [Online]. Available: http://dx.doi.org/10.1016/j.apenergy.2016.06.052

[6] Christy Pettey, "Gartner estimates ICT industry accounts for 2 percent of global CO2 emissions. Gartner Press Release," *Gart. Press Release*, 2007. [Online]. Available: http://www.gartner.com/it/page.jsp?id=503867

[7] Q. Zhang, Zhani, R. Boutaba, and J. L. Hellerstein, "Dynamic heterogeneity-aware resource provisioning in the cloud," *IEEE Trans. Cloud Comput.*, vol. 2, no. 1, pp. 14–28, 2015.

[8] X. Zhu, L. T. Yang, H. Chen, J. Wang, S. Yin, and X. Liu, "Real-time tasks oriented energy-aware scheduling in virtualized clouds," *IEEE Trans. Cloud Comput.*, vol. 2, no. 2, pp. 168–180, 2014. [Online]. Available: http://ieeexplore.ieee.org/document/6803043/

[9] R. Buyya, A. Beloglazov, and J. Abawajy, "Energy-efficient management of data center resources for cloud computing - a vision, architectural elements, and open challenges," in *Parallel Distrib. Process. Tech. Appl.*, 2010, pp. 6–17.

[10] M. Armbrust, I. Stoica, M. Zaharia, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, and A. Rabkin, "A view of cloud computing," *Commun. ACM*, vol. 53, no. 4, p. 50, 2010. [Online]. Available: http://portal.acm.org/citation.cfm?doid=1721654.1721672

[11] L. A. Barroso and U. Hölzle, *The Datacenter as a Computer*, 2013, vol. 24. [Online]. Available: http://www.valleytalk.org/wp-content/uploads/2013/10/WSC_2.4_Final-Draft.pdf

[12] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield, "Xen and the art of virtualization," *Proc. Ninet. ACM Symp. Oper. Syst. Princ. - SOSP '03*, 2003. [Online]. Available: http://portal.acm.org/citation.cfm?doid=945445.945462

[13] L. Liu, H. Wang, X. Liu, X. Jin, W. B. He, Q. B. Wang, and Y. Chen, "Greencloud: A new architecture for green data center," *Proc. 6th Int. Conf. Ind. Sess. Auton. Comput. Commun. Ind. Sess.*, pp. 29–38, 2009. [Online]. Available: http://doi.acm.org/10.1145/1555312.1555319

[14] Í. Goiri, J. L. Berral, J. O. Fitó, F. Juli, R. Nou, J. Guitart, R. Gavaldà, and J. Torres, "Energy-efficient and multifaceted resource management for profit-driven virtualized data centers," *Futur. Gener. Comput. Syst.*, vol. 28, no. 5, pp. 718–731, 2012. [Online]. Available: http://dx.doi.org/10.1016/j.future.2011.12.002

[15] X. Wang, Z. Du, and Y. Chen, "An adaptive model-free resource and power management approach for multi-tier cloud environments," *J. Syst. Softw.*, vol. 85, no. 5, pp. 1135–1146, 2012. [Online]. Available: http://dx.doi.org/10.1016/j.jss.2011.12.043

[16] Z. Xiao, S. Member, W. Song, and Q. Chen, "Machines for cloud computing environment," vol. 24, no. 6, pp. 1107–1117, 2013. [Online]. Available: http://ieeexplore.ieee.org/document/6311403/

[17] S. Bazarbayev, M. Hiltunen, K. Joshi, W. H. Sanders, and R. Schlichting, "Content-based scheduling of virtual machines (vms) in the cloud," *Proc. - Int. Conf. Distrib. Comput. Syst.*, no. July, pp. 93–101, 2013.

[18] I. Hwang and M. Pedram, "Hierarchical virtual machine consolidation in a cloud computing system," *IEEE Int. Conf. Cloud Comput. CLOUD*, pp. 196–203, 2013.

[19] X. Li, Z. Qian, S. Lu, and J. Wu, "Energy efficient virtual machine placement algorithm with balanced and improved resource utilization in a data center," *Math. Comput. Model.*, vol. 58, no. 5-6, pp. 1222–1235, 2013. [Online]. Available: http://dx.doi.org/10.1016/j.mcm.2013.02.003

[20] J. Dong, X. Jin, H. Wang, Y. Li, P. Zhang, and S. Cheng, "Energy-Saving virtual machine placement in cloud data centers," *Proc. - 13th IEEE/ACM Int. Symp. Clust. Cloud, Grid Comput. CCGrid 2013*, pp. 618–624, 2013.

[21] X. Xu, W. Dou, X. Zhang, and J. Chen, "Enreal: An energy-aware resource allocation method for scientific workflow executions in cloud environment," *IEEE Transactions on Cloud Computing*, vol. 4, no. 2, pp. 162–179, 2016.

[22] X. Xu, X. Zhang, M. Khan, W. Dou, and S. Xue, "A balanced virtual machine scheduling method for energy-performance tradeoffs in cyber-physical cloud systems," *Future Generation Computer Systems*, 2017.

[23] A. Beloglazov, J. Abawajy, and R. Buyya, "Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing," *Futur. Gener. Comput. Syst.*, vol. 28, no. 5, pp. 755–768, 2012. [Online]. Available: http://dx.doi.org/10.1016/j.future.2011.04.017

[24] M. A. Khoshkholghi, M. N. Derahman, A. Abdullah, S. Subramaniam, and M. Othman, "Energy-efficient algorithms for dynamic virtual machine consolidation in cloud data centers," *IEEE Access*, vol. 5, pp. 10 709–10 722, 2017.

[25] Y. Liu, X. Sun, W. Wei, and W. Jing, "Enhancing energy-efficient and qos dynamic virtual machine consolidation method in cloud environment," *IEEE Access*, vol. 6, pp. 31 224 – 31 235, 2018.

[26] A. Mosa and R. Sakellariou, "Virtual machine consolidation for cloud data centers using parameter-based adaptive allocation," *Proceedings of the Fifth European Conference on the Engineering of Computer-Based Systems (ECBS)*, 2017.

[27] W. Guo, X. Ren, W. Tian, and S. Venugopal, "Self-adaptive consolidation of virtual machines for energy-efficiency in the cloud," *Proceedings of the 2017 VI International Conference on Network, Communication and Computing (ICNCC)*, 2017.

[28] Q. Liang, J. Zhang, Y. H. Zhang, and J. M. Liang, "The placement method of resources and applications based on request prediction in cloud data center," *Inf. Sci. (Ny).*, vol. 279, pp. 735–745, 2014. [Online]. Available: http://dx.doi.org/10.1016/j.ins.2014.04.026

[29] Z. Tang, Y. Mo, K. Li, and K. Li, "Dynamic forecast scheduling algorithm for virtual machine placement in cloud computing environment," *J. Supercomput.*, vol. 70, no. 3, pp. 1279–1296, 2014.

[30] F. Juarez, J. Ejarque, and R. M. Badia, "Dynamic energy-aware scheduling for parallel task-based application in cloud computing," *Futur. Gener. Comput. Syst.*, vol. 78, pp. 257–271, 2018. [Online]. Available: http://dx.doi.org/10.1016/j.future.2016.06.029

[31] M. Xu, A. V. Dastjerdi, and R. Buyya, "Energy efficient scheduling of cloud application components with brownout," *IEEE Transaction on Sustainable Computing*, vol. 1, pp. 40–53, 2016.

[32] Q. Chen, J. Chen, B. Zheng, J. Cui, and Y. Qian, "Utilization-based vm consolidation scheme for power efficiency in cloud data centers," *IEEE ICC 2015-Workshop on Cloud Computing Systems, Networks, and Applications (CCSNA)*, pp. 1928–1933, 2015.

[33] H. R. Faragardi, S. Dehnavi, T. Nolte, M. Kargahi, and T. Fahringer, "An energy-aware resource provisioning scheme for real-time applications in a cloud data center," *Journal of Software: Practice and Example*, pp. 1–24, 2018.

[34] H. R. Faragardi, A. Rajabi, K. Sandström, and T. Nolte, "Eaica: An energy-aware resource provisioning algorithm for real-time cloud," *2016 IEEE 21st International Conference on Emerging Technologies and Factory Automation (ETFA)*, 2016.

[35] D. Gmach, J. Rolia, L. Cherkasova, and A. Kemper, "Resource pool management: Reactive versus proactive or let's be friends," *Comput. Networks*, vol. 53, no. 17, pp. 2905–2922, 2009. [Online]. Available: http://dx.doi.org/10.1016/j.comnet.2009.08.011

[36] J. Cao, Y. Wu, and M. Li, "Advances in grid and pervasive computing," vol. 3947, 2006. [Online]. Available: http://link.springer.com/10.1007/11745693

[37] A. Beloglazov and R. Buyya, "Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in cloud data centers," *Concurr. Comput. Pract. Exp.*, vol. 24, no. 13, pp. 1397–1420, 2012.

[38] G. L. Valentini and B. Khan, Samee U.and Pascal, "Energy-efficient resource utilization in cloud computing," *Large Scale Network-Centric Distributed System*, 2012. [Online]. Available: http://onlinelibrary.wiley.com/doi/10.1002/9781118640708.ch16/summary

[39] H. Li, H. Zhu, G. Ren, H. Wang, H. Zhang, and L. Chen, "Energy-aware scheduling of workflow in cloud center with deadline constraint," *2016 12th Int. Conf. Comput. Intell. Secur.*, pp. 415–418, 2016. [Online]. Available: http://ieeexplore.ieee.org/document/7820492/

[40] L. Wang, S. U. Khan, D. Chen, J. Kodziej, R. Ranjan, C. Z. Xu, and A. Zomaya, "Energy-aware parallel task scheduling in a cluster," *Futur. Gener. Comput. Syst.*, vol. 29, no. 7, pp. 1661–1670, 2013. [Online]. Available: http://dx.doi.org/10.1016/j.future.2013.02.010

[41] S. Hosseinimotlagh, F. Khunjush, and S. Hosseinimotlagh, "A cooperative two-tier energy-aware scheduling for real-time tasks in computing clouds," *Proc. - 2014 22nd Euromicro Int. Conf. Parallel, Distrib. Network-Based Process. PDP 2014*, pp. 178–182, 2014.

[42] E. N. Watanabe, P. P. V. Campos, K. R. Braghetto, and D. Macedo Batista, "Energy saving algorithms for workflow scheduling in cloud computing," *Comput. Networks Distrib. Syst. (SBRC), 2014 Brazilian Symp.*, pp. 9–16, 2014. [Online]. Available: http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=6927114

[43] K. Wu, R. Du, L. Chen, and S. Yan, "An energy-saving virtual-machine scheduling algorithm of cloud computing system," *2013 Int. Conf. Inf. Sci. Cloud Comput. Companion*, no. 1, pp. 219–224, 2013. [Online]. Available: http://ieeexplore.ieee.org/document/6973595/

[44] P. Zhang and M. Zhou, "Dynamic cloud task scheduling based on a two-stage strategy," *IEEE Transactions on Automation Science and Engineering*, pp. 1–12, 2017. [Online]. Available: https://doi.org/10.1109/TASE.2017.2693688

[45] A. Marahatta, Y.-S. Wang, F. Zhang, A. K. Sangaiah, S. K. Sah Tyagi, and Z. Liu, "Energy-aware fault-tolerant dynamic task scheduling scheme for virtualized cloud data centers," *Mobile Networks and Applications*, 2018.

[46] "Bayes Classifier." [Online]. Available: https://en.wikipedia.org/wiki/Bayes_classifier

[47] Y. Mansouri, A. N. Toosi, and B. Rajkumar, "Cost optimization for dynamic replication and migration of data in cloud data centers," *IEEE Transactions on Cloud Computing*, 2017.

[48] W. Dargie, "Estimation of the cost of vm migration," *23rd International Conference on Computer Communication and Networks (ICCCN)*, 2014.

[49] S. K. Mishra, D. Puthal, B. Sahoo, P. P. Jayaraman, S. Jun, A. Y. Zomaya, and R. Ranjan, "Energy-efficient vm-placement in cloud data center," *Sustainable Computing: Informatics and Systems*, 2018.

[50] X. Chen, Y. Chen, A. Y. Zomaya, R. Ranjan, and S. Hu, "Cevp: Cross entropy based virtual machine placement for energy optimization in clouds," *The Journal of Supercomputing*, vol. 72, no. 8, pp. 3194–3209, 2016.

[51] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. De Rose, and R. Buyya, "Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," *Software: Practice and Experience (SPE), Wiley Press*, vol. 41, no. 1, pp. 23–50, 2011.