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/TII.2022.3153508, IEEE Transactions on Industrial Informatics

Two-Phase Industrial Manufacturing Service Management for Energy Efficiency of Data Centers

Weizhe Zhang, *Senior Member, IEEE*, Rahul Yadav *Member, IEEE*, and Yu-Chu Tian *Senior Member, IEEE*, Sumarga K. Sah Tyagi, Ibrahim A. Elgendy, and Omprakash Kaiwartya

***Abstract*—Data-driven industrial manufacturing services are proliferating. They use large amounts of data generated from Industrial-Internet-of-Things (IIoT) devices for intelligent services to end-service-users. However, cloud data centers hosting these services consume a huge amount of energy, resulting in a high operational cost. To address this issue, an energy-efficient resource allocation framework is proposed in this paper for cloud services. It operates in two phases. Firstly, a multi-threshold-based host CPU utilization classification scheme is developed to classify hosts into four groups for improved CPU resource allocation. It is designed through analyzing CPU utilization data by using the least median squares regression technique. Thereby, the scheme limits search space, thus reducing time complexity. In the second phase, with a metaheuristic search, an energy- and thermal-aware resource allocation method is developed to find an energy-efficient host for allocating resources to services. From real data center workload traces, extensive experiments show that our framework outperforms existing baseline approaches with 6.9%, 33.75%, and 34.1% on average in terms of temperature, energy consumption, and service-level-agreement violation, respectively.**

***Index Terms*—Industrial manufacturing service, energy efficiency, resource allocation, data center.**

I. Introduction

Industry 4.0 refers to the process of the digitization of the manufacturing sector to create an ecosystem for industry with a focus on manufacturing and supply chain management. The Industrial-Internet-of-Things (IIoT) includes smart sensors, camera systems, smart meters, industrial robotics, and actuators to leverage the power of smart machines and real-time analytics in a cloud computing environment (CCE) [1]. The IIoT market is expected to grow from $64.00 billion in 2018 to $91.40 billion by 2023, at a compound annual growth rate of 7.39% [2] [3]. In Data-Driven industrial Manufacturing (D2M) services, IIoT devices have great potential in sustainable and green practices, supply chain traceability, quality control, and overall supply chain efficiency. Consequently, an enormous amount of data is being generated from IIoT devices and analyzed at cloud data centers for the provisioning of efficient D2M services. An extensive analysis of the big industrial data has been carried out at cloud data centers [4], [5].

The CCE provides manufacturing as a service (MaaS) to the manufacturing industry, enabling the industry to take benefits by minimizing operational and administrative costs. MaaS includes different types of D2M services, such as supply chain management, and asset tracking & optimization. These services require a massive amount of computing, networking, and storage resources for their delivery to end-service-users [6]. Therefore, CCE requires support of a large amount of energy, and such energy consumption is increasing with the rapid growth of the demand for D2M services. The main portion of energy consumption in CCE is in their hardware infrastructure including servers (hosts), storage, network devices, and cooling. As hardware devices still consume a large amount of energy when they are idle, such energy consumption leads to enormous energy wastage. It is reported that physical hosts use nearly 30% of their peak power consumption while sitting idle 70% of time [7]. So, a basic reason of energy waste in data centers' infrastructure is under-utilization [8]. In the United States, cloud data centers consume about 2% (70 billion kWh) of the total energy production. Therefore, improving the energy-efficiency of cloud data centers is desirable for a sustainable and cost-effective CCE.

A challenge in this context is how to reduce host energy consumption while ensuring the Service Level Agreements (SLAs) delivered by cloud manufacturing services providers. To address this challenge, improvement in the level of host resource utilization will help reduce the total energy consumption in a CCE. However, naively improving host utilization levels may affect the temperature of the host and the SLA delivered by cloud manufacturing services providers [9]. So, to reduce the SLAs violation, select D2M services from either a service-overloaded-host (SO-host) or a service-underloaded-host (SU-host) for re-allocation according to the current resource requirement. Broadly speaking, energy-efficient resource allocation can be performed by using three major tasks: (1) to detect CPU utilization level; (2) to select D2M services form SO-host; and (3) to find an energy-efficient host for allocating resources to D2M services [10].

Most existing investigations into resource allocation to services focus on traditional cloud environments without explicit considerations of the impact of host temperature in overall performance and SLA violation. Lin *et al.* [11] have proposed

W. Zhang, is with the Harbin Institute of Technology, Shenzhen and with Cyberspace Security Research Center, Peng Cheng Laboratory, Shenzhen 518055, China (e-mail: wzzhang@hit.edu.cn,).

R. Yadav is with Cyberspace Security Research Center, Peng Cheng Laboratory, Shenzhen 518055, China Email: rahulyd@pcl.ac.cn

Y-C Tuan is with the School of Computer Science, Queensland University of Technology, Brisbane QLD, Australia. Email: y.tian@qut.edu.au

S. K. S. Tyagi is with the School of Electronic and Information Engineering, Zhongyuan University of Technology, Zhengzhou, China. Email: sumarga@zut.edu.cn

I. A. Elgendy is with the Harbin Institute of Technology, and with Menoufia University, Shibin el Kom, Egypt Email: ibrahim.elgendy@hit.edu.cn

O. Kaiwartya is with the School of Science and Technology, Nottingham Trent University, Nottingham NG1 4FQ, U.K. Email: omprakash.kaiwartya@ntu.ac.uk

a resource-constraint project scheduling approach based on a genetic algorithm for solving resource allocation problems in cloud manufacturing environments. However, this proposal neglects the fact that the temperature of the hosts also plays a significant role in resource utilization and energy consumption. Thus, the allocation of resources to the services significantly affects the overall temperature in CCEs. Mikko *et al.* [12] have introduced a method to optimize the cost of data centers by analyzing the waste heat utilization from the perspectives of both data centers and district heating network operators. They have considered the timing of data center waste heat production from an existing data center load profile. Al-Qawasmeh *et al.* [13] have introduced a power- and thermal-aware approach for the consolidation of workload in heterogeneous cloud data centers. They have designed optimization methods to assign the performance state of CPU core at data center level. Mhedheb *et al.* [14] have presented an approach to identify critical hosts for Virtual Machine (VM) migrations according to a threshold. The threshold is computed by using the utilization and temperature of the host. However, all these approaches still lack SLA violation and communication cost mechanism. This leads to high costs for searching an energy-efficient host for resource allocation, resulting in inefficient energy management with possible SLA violation.

Motivated from the aforementioned work, this paper presents a two-phase framework for energy-efficient resource allocation in CCE. In the first phase, an adaptive multi-threshold-based host CPU utilization classification (MCU) scheme is proposed. It classifies the hosts into four groups: SU-hosts, service-allocate hosts (SA-hosts), normal-loaded hosts (NL-hosts), and SO-hosts. For this purpose, we use the least median square (LMS) regression technique to estimate three thresholds ($T_{lower}, T_{middle}, T_{upper}$) based on a statistical analysis of past CPU utilization. In the second phase, an energy- and thermal-aware resource allocation scheme is proposed. It is designed with a metaheuristic approach to search for an energy-efficient host for resource allocation to D2M services based on host temperature. The advantages of metaheuristics include their progressive optimization ability, incremental and step-wise searching manner, and a controllable search time. Often a near-optimal solution can be found within a specified period of time. Hence, a metaheuristic approach is suitable for applications with real-time constraints. The main contributions of this paper are summarized below:

- We optimize the energy consumption of the CCE during the peak time by exploring computation energy consumption, communication energy consumption, and cooling energy consumption. This work presents cloud computing architecture based on a data-engine and control engine for building an intelligent manufacturing system. It formulates the resource allocation problem as a constrained optimization problem to minimize the energy consumption under several constraints such as dynamically fluctuating resources and upper-temperature threshold.
- A two-phase framework is proposed. In the first phase, a multi-threshold-based host CPU classification (MCU) scheme is used to classify various hosts into four groups for efficient utilization of host resources with a reduced risk of overloading hosts. In the second phase, we introduce a metaheuristic energy- and thermal-aware resource allocation scheme (ETV) to search an energy-efficient host for processing D2M services. Proposed scheme reduces the risk of over-heating hosts, maximizes energy savings, and compresses time complexity in problem solving.

The paper is organized as follows: Section 2 presents the architecture of the CCE. Section 3 formulates the problem of energy efficiency. Section 4 presents our framework. Section 5 conducts simulations. Finally, Section 6 concludes the paper.

II. System Architecture and Model

The section outlines the system architecture of cloud computing with D2M services and formulates our system model.

A. Overview of System Architecture

CCE for D2M services is a new network manufacturing paradigm. It organizes smart-manufacturing resources (e.g., robots, sensors, smart cameras, and IoT) and computing services (e.g., data analytic, real-time production monitoring, and user emotion data estimation) over networks. This section discusses the cloud computing architecture, as shown in Fig. 1, which consists of four main layers as discussed below.

1) Terminal Layer: The end-service-users is a group that requests smart-manufacturing resources and computing services from cloud manufacturing services providers as per their requirements. The end-service-users can monitor their production line by using terminal devices.

2) Cloud Layer: An enormous amount of data being generated by IIoT devices is stored in the cloud data center and applies machine learning or deep learning algorithms for providing intelligent manufacturing services. Cloud layer also provides high-performance computation resources to D2M services and this layer is equipped with a data-engine and control engine.

- *Data-Engine*- A data-engine collects data being generated by IIoT devices and carries out comprehensive big data analysis through artificial intelligence algorithms, which effectively execute the D2M services.
- *Control Engine*- The control engine plays a crucial role in managing and allocating the communication and computing resources of the cloud data center (e.g., allocate resources to D2M services as per their requirements, network type, communication quality, data flow between services, and other dynamic parameters). It also sends real-time resources availability status to the data-engine, while gathering the significant data analytics results of the data-engine. Thus, the control engine can allocate cloud resources efficiently to D2M services.

3) Gateway Layer: This layer consists of the routers or switches, a bridge between the smart-manufacturing layer and the cloud layer. It plays a significant role in interchange data and information between these layers.

4) Smart-Manufacturing Layer: This layer consists of all manufacturing hardware resources, e.g., camera, Wi-Fi antenna, motor-rotation interface, temperature sensors, vibration frequency sensors, RFID tag, Li-battery, pressure sensor, and LED indicator light. These devices are assembled into different types of robots, each of which performs a particular task as per end-service-users requirements.

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/TII.2022.3153508, IEEE Transactions on Industrial Informatics

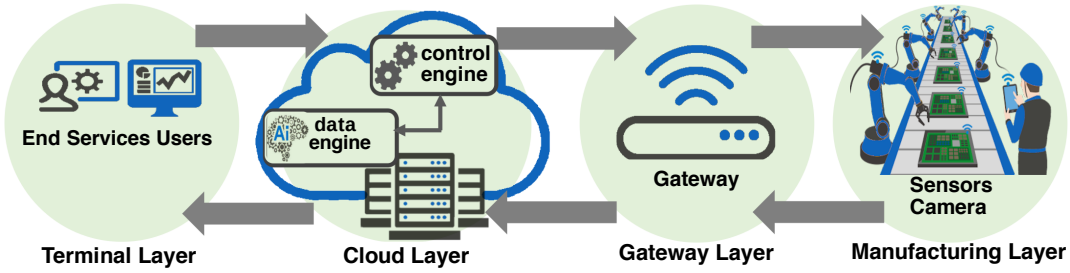

Fig. 1: The system architecture of industrial manufacturing services with cloud support.

B. System Models

The considered cloud computing system is presented in Fig. 1. The requests of services from users are considered as tasks. The total number of submitted tasks is denoted as n. We consider that the same number of VMs are required to execute these tasks [15]. The VMs are denoted by $V = (v_1, v_2, ..., v_j, ..., v_n)$. We model the workload in terms of the n services $S = (s_1, s_2, ..., s_j, ..., s_n)$ that need to be placed on m hosts $H = (h_1, h_2, ..., h_i, ..., h_m)$. We assume that each VM processes its associated D2M service, and the VM is terminated after the service is executed. The key attributes of the D2M service s_j, are denoted by $(D_j, C(s_j))$, where D_j represents the data size of D2M service, and $C(s_j)$ is the required computation resources. Here, we assume that the size of a VM is similar to its associated D2M service workload. Therefore, we only need to consider the workload of service instead of VM. Let G denote a gateway device. Since we are addressing resource allocation, we submit all tasks at the beginning of the experiment. The energy consumption of CCE comes from computing, communication, and cooling. An observation time slot is denoted by q with $q = 1, 2, \ldots, l$. The duration of each time slot is κ.

1) Computing Energy Consumption Model: In a CCE, the energy consumption of computing is determined by the energy consumed by CPU, storage, memory, and network. The CPU is the leading energy consumer. Thus, the energy consumption model mainly focuses on the energy consumed by the CPU.

A host consumes a *base energy* P^h_{base} at its idle state, and the maximum energy P^h_{max} at its full utilization. Its total capacity of hosting services is represented by $C(host)$. For the j^{th} D2M service, the total number of required CPUs is denoted by $C(s_j)$. We use a binary variable δ_j to indicate whether or not the jth D2M service is offloaded to get resources from a host (value "1" for yes and value "0" for no). Therefore, the overall energy P^{host} consumed by a host is given by Marcel *et al.* [16] as follows:

$$P^{host} = P^h_{base} + (P^h_{max} - P^h_{base}) \frac{\sum_{j=1}^{n} C(s_j) \times \delta_j}{C(host)} \quad (1)$$

Equation (1) states that the utilization of a host given by the ratio between the sum of the CPU cores requested by each D2M service and the total number of CPU cores of the host.

The overall computing energy consumption of the CCE within an observed period of time is expressed as an $m \times l$ matrix P with elements $P^{host}_{i,q}$ $(q = 1, 2, ..., l; i = 1, 2, ..., m)$.

TABLE I: Notations and symbols.

B	Channel bandwidth
$C(s_j)$	CPU requirement of j^{th} D2M service
$C(host)$	Resource capacity of a host
C	Heat capacity of host
c	Safety parameter
$d_{i,k}$	Effect of heat recirculation to k^{th} host from i^{th} host
D_j	Data size of j^{th} D2M service
$E^{computing}$	Total computing energy consumption
$E^{cooling}$	Total cooling energy consumption
E^G	Total communication energy consumption
h_j	Channel power gain
H	A set of m Hosts
i, k	Indices for hosts, $i, j = 1, \ldots, m$
j	Index for VMs or services, $j = 1, \ldots, n$
l, m, n	Total nos. of time slots, hosts, and VMs, respectively
N_0	Noise power
P^{host}	Energy consumption of a host
P^h_{base}	Ideal energy consumption of host
P^h_{max}	Maximum energy consumption of host
P_j	Transmission power of IIoT device
q	Index for time slots, $q = 1, \ldots, l$
R	Thermal resistance of host
S	A set of n D2M services
$T^{inlet}_{i,q}$	Inlet temperature at i^{th} host on observe time q
$T^{cpu}_{i,q}$	Temperature at i^{th} host on observe time q
T^{up}	Maximum threshold of CPU temperature
T^{supply}	Supply cold air temperature from CRAC
T_{upper}	Upper loaded CPU utilization threshold
T_{middle}	Normal loaded CPU utilization threshold
T_{lower}	Underloaded CPU utilization threshold
X, Y	Data points
α, β	Intercept and slop variables
κ	The duration of a time slot

The computing energy consumption model of the CCE is:

$$E_{computing} = \kappa \sum_{q=1}^{l} \sum_{i=1}^{m} P^{host}_{i,q} \quad (2)$$

2) Communication Energy Consumption Model: In an offloading mode, a j^{th} D2M service first transfers its data to the cloud via a wireless access point (gateway). We assume that all wireless links are symmetric, and data transmission between a gateway to CCE is through a wired network [17]. A free-space propagation path-loss-model captures the large-scale fading of the channel from an IIoT device to a gateway. If a j^{th} D2M service is offloaded to cloud via a gateway, the transmitting time T^G_j of an offloaded D2M service depends on the available bandwidth, D2M service data size D_j, $j \in S$, and transmission rate. Since the D2M service is completely offloaded to the cloud via a gateway, the transmission time

T_j^G required by s_j can be obtained as

$$T_j^G = \frac{D_j}{r_j},\ r_j = B\log_2\left(1+\frac{P_j h_j}{N_0}\right),\ j=1,\ldots,n \quad (3)$$

where r_j is the achievable rate, P_j denotes the transmission power of the IIoT device, which offloaded j^{th} D2M service to the cloud via a gateway, and h_j represents the channel power gained from IIoT device to wireless gateway due to path loss and shadowing attenuation during offloading j^{th} D2M service, B represents to a channel bandwidth, and N_0 is a power noise. As in existing studies [18], we also do not consider the transmitted time required to send back computing results because it is much smaller than input data.

The energy consumption of transmitting the j^{th} D2M service is the product of the transmission time and transmit power for each IIoT device [19]. We use an exiting model to calculate the total transmission energy consumption E_j^G as

$$E_j^G = T_j^G P_j,\ j=1,\ldots,n \quad (4)$$

3) Cooling Energy Consumption Model: Computer room air conditioning (CRAC) is used for cooling management in a cloud data center. The amount of heat generated by the hosts as well as the efficiency of CRAC negatively impact on overall energy consumption [20]. The pattern of airflow in a typical cloud data center is intricate. This leads to a heat recirculation (HR) phenomenon, where hot air from the host outlets recirculates in the data center and is mixed with the supplied cold air from the CRAC, causing the temperature at the host inlets to be higher than that of supplied air. We use HR matrix **d** to describe the relationship between each pair of hosts. Its element $d_{i,k}$ expresses the recirculation rate generated from the i^{th} host to the k^{th} host, where $i \neq k$ [21]. The HR in a data center causes the rise of inlet temperature $T_{i,q}^{inlet}$ of the i^{th} host in observed time slot q than supplied cold air with temperature T_q^{supply}. Thus, HR plays a crucial role in increased energy consumption in cloud data centers. The effect of the heat contribution of all hosts on the inlet temperature of i^{th} host in observed time slot q is described by:

$$T_{i,q}^{inlet} = T_q^{supply} + \sum_{k=1}^{m} d_{i,k}P_k^{host},\quad i=1,\ldots,m;\ q=1,\ldots,l \quad (5)$$

The CPU generates an extensive amount of heat that impacts the inside temperature of the host. The host CPU temperature is governed by not only its energy consumption but also thermal resistance and heat capacity along with the inlet temperature. Thus, the CPU $T_{i,q}^{cpu}$ dissipated by the i^{th} host during observed time q can be defined by an Resistance-Capacitance model (RC model) as [22]

$$T_{i,q}^{cpu} = P_{i,q}^{host}R + T_{i,q}^{inlet} + (T_{i,q}^{ini} - P_{i,q}^{host}R - T_{i,q}^{inlet})e^{-\frac{q}{RC}},\quad i=1,\ldots,m; q=1,\ldots,l \quad (6)$$

where R and C represent thermal resistance (K/W) and heat capacity (J/K) of the host, respectively; T^{ini} denotes the initial temperature of i^{th} host CPU.

To keep the reliability of all hosts within the cloud data center, all inlet temperature $max(T_{i,q}^{inlet}) \leq T^{up}$, where T^{up} is a warning temperature (nominal values is 70 °C [20]). Therefore, the setting of cold air temperature supplied from the CRAC by using equation (5) is

$$\begin{cases} T_q^{supply} + \max\limits_{i\in H}\sum\limits_{k=1}^{m} d_{i,k}P_k^{host} \leq T^{up},\ q=1,\ldots,l \\ T_q^{supply} \leq T^{up} - \max\limits_{i\in H}\sum\limits_{k=1}^{m} d_{i,k}P_k^{host},\ q=1,\ldots,l \end{cases} \quad (7)$$

The amount of heat generated by hosts is directly proportional to computing energy consumption. The efficiency of the CRAC unit is generally characterized by the Coefficient of Performance (CoP), which is described as the ratio of the amount of heat removed by a cooling device to the consumed energy by CRAC units. For example, the expression of CoP = 2 means that the CRAC units will consume 50 J energy for removing 100 J heat. The CoP typically is non-linear with the supplied cold air temperature (T^{supply}). We use HP utility data center CoP model: $\mathbf{CoP}(T^{supply}) = \left(0.0068(T^{supply})^2 + 0.0008T^{supply} + 0.458\right)$ [21], [23]. Therefore, the energy consumption $E^{cooling}$ of cooling can be described as follows:

$$E^{cooling} = E^{computing}/CoP(T^{supply}) \quad (8)$$

The total amount of energy (E^{total}) consumed by CCE in an observed period of time is calculated by integrating equations (2), (4), and (8) as follows:

$$\begin{aligned} E^{total} &= E^{computing} + E^{cooling} + E^G \\ &= E^{computing} + \frac{E^{computing}}{CoP(T^{supply})} + E^G \\ &= \kappa\sum_{q=1}^{l}\sum_{i=1}^{m} P_{i,q}^{host}\left(1+\frac{1}{CoP(T_q^{supply})}\right) + E^G \end{aligned} \quad (9)$$

The maximum allowable input temperature is maintained through the overall performance of computing and cooling facilities of the CCE. Thus, the total energy consumption (E^{total}) can be modified by using equations (9) and (7) as:

$$E^{total} = \kappa\sum_{q=1}^{l}\sum_{i=1}^{m} P_{i,q}^{host}\left(1+\frac{1}{CoP(T^{up} - max_i\sum_{k=1}^{m} d_{i,k}P_k^{host})}\right) + E^G \quad (10)$$

This equation shows that all entry d_i of HR matrix **d** [21] and other parameters (e.g., T^{up} and κ) are determined. Therefore, the total energy consumption from cooling, communication, and computing in the observed time is directly affected by P_i^{host}, where P_i^{host} is a set of all computing power of the hosts in the CCE.

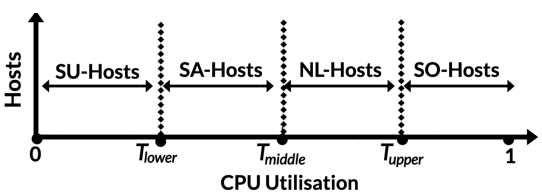

Fig. 2: Multi-threshold scheme for dividing hosts CPU utilization level in to four groups

III. Problem Formulation

This work aims to minimize the energy consumption of a CCE by leveraging the under-utilized computation resources of the hosts and shutting down idle hosts. The total energy consumption in a CCE can be split into energy consumption for computing ($E^{computing}$), energy consumption for communication (E^{G}), and energy consumption for cooling ($E^{cooling}$). Hence, we model the objective function as the total energy consumption of the CCE. We investigate how to allocate resources between D2M services and hosts such that the objective function is minimized.

Definition 1: The resource allocation decision between m D2M services and n hosts is defined as an $n \times m$ matrix Δ, where its $(i,j)^{th}$ element $\delta_{i,j}$ is defined as a binary value. $\delta_{i,j} = 1$ means the resources of the i^{th} host allocated to the j^{th} D2M service. Otherwise, $\delta_{i,j} = 0$. The problem is formulated as:

$$
\begin{aligned}
Y : \min_{\Delta} E^{total} &= E^{G} + \kappa \sum_{q=1}^{l} \sum_{i=1}^{m} \delta_{i,j} P_{i,q}^{host} \\
&\left(1 + \frac{1}{CoP(T^{up} - max_i \sum_{k=1}^{m} d_{i,k} P_k^{host})}\right) \qquad (11) \\
\text{s.t.} \quad C_1 &: \sum_{i \in H} \delta_{i,j} \leq 1, \quad \forall j \in S, \\
C_2 &: \sum_{j \in S} s_j(C(s_j), M) \leq h_i(C(host), M), \forall i \in H, \\
C_3 &: T_{i,q}^{cpu} \leq T^{up} \quad \forall i \in H, \forall q \in \{1, \ldots, l\}
\end{aligned}
$$

The objective function Y takes care of the holistic minimization of energy. The first constraint C_1 ensures that a D2M service is allocated to one and only one host. The second constraint C_2 ensures that the resources requirements of all D2M services allocated to one host are capped by the capacity of a host. The third constraint C_3 ensures that the thermal violation does not occur when more D2M services are executed on the host.

Because of the integer constraint $\delta_{i,j} \in \{0, 1\}$, the above optimization problem is an integer programming problem, which is generally non-convex and NP-hard. Thus, in the next section, an energy-efficient resources allocation framework is proposed based on a metaheuristic approach. It finds a near-optimal solution in a reasonable amount of time.

IV. A two-phase framework

The efficient energy consumption can be achieved by consolidating the computation load into a small number of hosts while setting idle hosts to an energy-saving mode. So, we have divided the energy consumption problem into three sub-problem. First, detecting overloaded host, second, selecting service from overloaded host for migration, third, allocating resources to migrated service.

Therefore, our resource management framework operates in the following two phases. In the first Phase, an *MCU* scheme is used to classify hosts into four groups: SU-hosts, SA-hosts, NL-hosts, and SO-hosts. *MCU* scheme help to detect overloaded host. After detecting an overloaded host, we need to select service from an overloaded host for migration. So, we use an existing approach for service selection, named Bandwidth-Aware Dynamic Selection Policy (Bw) [9]. In the second Phase, an *ETV* scheme is designed to find the energy-efficient host from the SA-host group for resource allocation to D2M services, which improves energy-efficiency, the temperature of the host, reduces the complexity, and reduces SLA violations. Our scheme proposed in this work is a general one applicable to the scenarios with or without GPUs, the only difference is the inclusion or exclusion of a GPU energy consumption model. For the simplicity and without loss of generality, we do not choose to include a GPU model for the demonstration of our scheme in this paper.

A. Multi-threshold scheme to classify host CPU utilization

The proliferation of D2M services demands an enormous amount of computing resources to provide these services efficiently. However, The dynamic property of a CCE is a big concern for cloud manufacturing service providers. A static threshold is not a reliable solution for the dynamic workload of the CCE. Therefore, we propose a multi-threshold based host CPU utilization classification for this dynamic nature. In this section, we solve the sub-problem of the proposed framework for scenarios when a host is considered to be service-underloaded, service-allocated for resource allocation to D2M services, normal loaded, or service-overloaded.

We classify all cloud data center hosts based on their current CPU utilization. As shown in Fig. 2, we set three host CPU utilization thresholds in the range between 0 and 1, i.e., $0 \leq T_{lower} < T_{middle} < T_{upper} \leq 1$. From this classification, hosts are grouped to SU-hosts, SA-hosts, NL-hosts, and SO-hosts. The process of the proposed *MCU* is discussed as follows:

- Firstly, obtain current CPU utilization H_i^u of the i^{th} host.
- If $H_i^u \geq T_{upper}$, this i^{th} host is categorized into SO-host group. The hosts in this group must migrate some D2M services to SA-host ($T_{lower} \leq H_i^u \leq T_{middle}$) group, thus minimizing SLA violation with high energy efficiency.
- Extravagant D2M services migration from one host to another in the cloud data center leads to inefficient energy consumption and high SLA violation [9]. Therefore, all D2M services of the host in NL-host ($T_{middle} \leq H_i^u \leq T_{upper}$) group are kept unchanged.
- If $H_i^u \leq T_{lower}$, the host is categorized into SU-host group. The hosts in this group must migrate all D2M services to the SA-host group. As a result, idle hosts are switched to low-energy-mode or shut-down for energy savings. The hosts are reactivated based on resource demand increment [10], [24].

For the proposed MCU scheme, how to obtain the thresholds T_{lower}, T_{middle} and T_{upper}? Assigning a constant value to these thresholds is not a reliable solution for unpredictable or dynamic workload of the CCE. Therefore, the least median square (*LMS*) regression technique [9] is used to adjust these thresholds automatically through a statistical analysis of the past host CPU utilization. An *LMS* estimator is more robust than other estimators such as variance, ordinary least squares, standard deviation, and median. Let c denote a safety parameter that defines how fast the system allocates resources to D2M services. The adaptive thresholds of the proposed MCU

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/TII.2022.3153508, IEEE Transactions on Industrial Informatics

approach are defined as follows:

$$\begin{cases} T_{lower} = a(1 - c \times LMS), \\ T_{middle} = b(1 - c \times LMS), \\ T_{upper} = (1 - c \times LMS) \end{cases} \tag{12}$$

where c is set as 0.5 from the existing work by Yadav *et al.* [9]. A larger value of c implies higher energy consumption, but lower SLA violation. A smaller value of c indicates lower energy consumption but higher SLA violation. We use two hyper-parameters a and b to divide upper threshold into 30%, and 90% for T_{lower} and T_{middle}, respectively. Thus, we set $a = 0.3$ and $b = 0.9$, respectively.

For calculating the *LMS*, we need to calculate the following Ordinary Least Squares (OLS) model:

$$Y_i = \alpha + \beta X_{\mathrm{i}} + \varepsilon_{\mathrm{i}} \tag{13}$$

$$\varepsilon_i = Y_i - (\alpha + \beta X_i) \tag{14}$$

where ε_i is an independent variable called residuals, and X and Y are data points. This model aims to minimize the value of residuals ε_i. If the value of all residuals ε_i reaches zero, then an optimal model is found with all given data points. $i \in H_u$, where H_u is a set of CPU utilization of all hosts in the data center. The goal is to estimate the parameters, α and β, which is usually called the intercept and slope of the fitted line in the given data set, respectively. In OLS, a line is fitted in a given dataset by estimating the value of α and β to minimize the sum of squared residuals (SR) as described below:

$$\min_{\alpha,\beta} SR = \sum_{i=1}^{m} \Big(Y_i - (\alpha + \beta X_i)\Big)^2 \tag{15}$$

For minimizing the values of α and β, partially differentiate Equation (15) with respect to α and β

$$\frac{\partial SR}{\partial \alpha} = -2 \sum_{i=1}^{m} \Big(Y_{\mathrm{i}} - (\alpha + \beta X_{\mathrm{i}})\Big) = 0 \tag{16}$$

$$\frac{\partial SR}{\partial \beta} = -2 \sum_{i=1}^{m} \Big(Y_{\mathrm{i}} - (\alpha + \beta X_{\mathrm{i}})\Big) X_{\mathrm{i}} = 0 \tag{17}$$

Simplifying Equation (16) gives

$$\sum_{i=1}^{m} Y_{\mathrm{i}} - \sum_{i=1}^{m} \alpha - \sum_{i=1}^{m} \beta X_{\mathrm{i}} = 0 \tag{18}$$

It follows that

$$\alpha = \bar{Y} - \beta \bar{X} \tag{19}$$

Similarly, through simplifying Equation (17) , we obtain:

$$\beta = \frac{\sum_{i=1}^{m} (X_i - \bar{X})(Y_i - \bar{Y})}{\sum_{i=1}^{m} (X_i - \bar{X})^2} \tag{20}$$

where $\bar{Y}$ and $\bar{X}$ are the means of variables Y_i and X_i, respectively. Formally, the least median of squares fit is determined by the median of the residuals after plotting the values of α and β as follows:

$$LMS = \underset{\forall i \in H}{\mathrm{med}} \Big(Y_i - (\alpha + \beta X_i)\Big)^2 \tag{21}$$

The selection of service is an immediate task after the host detected service-overloaded, and the decision needs to be made to select services for migration from SO-host to SA-host. The efficient selection of service is an essential task to minimize performance degradation and SLA violation. For service selection, we use an existing approach, named Bandwidth-Aware Dynamic Selection Policy (Bw) [9].

Algorithm 1: Hosts Status Detecting Algorithm

Input: *HostList*, *ServiceList*, T_{lower}, T_{middle}, T_{upper}
Output: Resources Allocation to D2M Services
Phase : Service Allocation
`SortDecreasingOrder.ServiceListUtil(`*ServiceList*`);`
foreach *(service:ServiceList)* **do**
  HostAllocated ← NULL;
  foreach *(Host:HostList)* **do**
    if *(* $s_j(C(s_j), M) \leq h_i(C(host), M)$*)* **then**
      CurrentHostCpuUtil ← `GetHostCpuUtil(`*Host, service*`);`
    if *(*$T_{lower} \leq$ *CurrentHostCpuUtil* $\leq T_{middle}$*)* **then**
      MapService ← `SearchHost.PlaceService();`
    if *(HostAllocated*$\neq$ *Null)* **then**
      Allocate Host Resources to Service;
    else
      (HostAllocated== Null)
    HostAllocated ← ActiveNewHostfromSleep-modeHostList() ;
Phase : SO-Host and SU-Host Detection
for *Host:HostList* **do**
  CurrentHostCpuUtil = `GetHostCpuUtil(`*Host*`);`
  if *(*$T_{upper} \leq$ *CurrentHostCpuUtil)* **then**
    SO-Host ← `GetOverloadedHost();`
    SelectService ← `SelectServiceForMigration(`*SO-Host*`);`
    MapService ← `SearchHost.PlaceService(`*SelectService*`);`
  else
    ($T_{lower} \geq$ CurrentHostCpuUtil);
  HostUnderloaded ← `GetUnderloadedHost();`
  SelectService ← `SelectServiceForMigration(`*SU-Host*`);`
  MapService ← `SearchHost.PlaceService(`*SelectService*`);`

B. Energy and thermal-aware resource allocation (ETV)

An integer programming solver for the optimization problem in Equation (11) has high complexity. It is not feasible to solve D2M resource allocation under dynamic resources constraints. Thus, a metaheuristic approach is proposed to solve the D2M resource allocation problem. The allocation of resources to D2M services plays a significant role in the efficient utilization of resources and energy consumption in

the cloud data center. In this paper, we balance the tradeoff between computing and cooling systems. The basic idea of this scheme is that the best suitable host for resource allocation to D2M services is directly dependent on the current temperature of the host. So, find a host whose thermal temperature is minimum than the other hosts in CCE for resource allocation.

In **Algorithm 1**, *"HostList"* and *"ServiceList"* represent the sets of hosts and services in the cloud data center, respectively. Our proposed framework works into two phases: firstly, allocate host resources for newly offloaded services; and secondly, detect SO-hosts and SU-hosts (for minimizing energy consumption and SLA violation) and migrate their services in SA-hosts group. First of all, this algorithm sorts all D2M services as per their Million Instructions Per Second (MIPS) requirements in a decreasing order (line 2) and then check that the resource requirements of D2M service are guaranteed by available hosts (line 5-6). In next step, the algorithm finds a host which is categorized in SA-host group by using T_{lower}, and T_{middle} (line 7) for resources allocation. The allocation of resources is process with help of Algorithm 2 (line 8). Next, if no host is fulfilling new D2M service requirements, then the shutdown-mode host will turn-on for allocating resources to services (line 11-13). In second phase, the detection of SO-host and SU-host is a crucial stage to minimize energy consumption, performance degradation, and SLA violation (line 14-21). After detecting SO-host, select a D2M service for migration (line 22). In the next stage, selected D2M services should be placed in the host according to their temperature. The next algorithm is to describe the host resources allocation workflow according to each host's temperature.

Algorithm 2 refers to the search for the best host for resource allocation. In each scheduling interval (five minutes in this case), the algorithm makes a local list (*HostSearchList*) of hosts as per their thermal cost, which is calculated with the help of Algorithm 3 (line 4). This local list (*HostSearchList*) limit the search space to find the best host for resource allocation, as a result, reduces the time complexity. Next, in each iteration, the algorithm compares the thermal cost of each host from a list (*HostSearchList*) and select a host whose thermal cost is minimum than other hosts for allocating its resource to D2M service (6-8).

Algorithm 3 refers to energy and thermal-based cost construction. It takes host list, service, T_{lower}, and T_{middle} as input and returns the best feasible hosts list ($BestFeasible\text{-}HostList$) with their cost calculated by equation (10). The first stage of an algorithm is making a feasible host list ($FeasibleHostList$) by using T_{lower}, and T_{middle} (line 5-6), which represents the finite search space to construct the best feasible host list, and it positively impacts on the reduction of time complexity. To the end, this procedure will return a $BestFeasibleHostList$ with the cost of each host belong in this list. In the next section, we will evaluate the effectiveness of the proposed framework.

Algorithm 2: Search Host for Resources Allocation to D2M Service

Input: *HostList*, *service*
Output: Best Host For Resources Allocation to D2M Service

```
1 for (q:T) do
2     GetBestHost ← True;
3     while (GetBestHost) do
4         HostSearchList ← CostConstruction(HostList, service) ;
5         BestHost ← Null ;
6         for (LocalBestHost:HostSearchList) do
7             if (Cost of (LocalBestHost) < Cost of (BestHost)) then
8                 BestHost ← LocalBestHost;
9 return BestHost
```

V. Simulation Setup & Results Analysis

A. Simulation Setup

The prototype of the proposed scheme is implemented on top of CloudSim [25] with real dataset traces, which models large datacenters provisioning computing infrastructures as services. CloudSim implements a view of infinite computing resources. This feature is important for us to evaluate the proposed scheme on a large virtualized data center infrastructure. For simplicity and without loss of generality, we choose the CloudSim simulator to demonstrate our scheme. In contrast, validation on a real Cloud infrastructure will be extremely difficult for performing different experiments in order to examine the full functionality of the implemented algorithm and the impact of the resources allocation strategies. Therefore, simulation is the best choice for evaluating the efficiency of proposed schemes.

In our simulation setup, we analyze the performance of host temperature, SLA violation metric [9] [26], ESV metric and energy consumption. These results are compared with existing baseline approaches. We install 800 heterogeneous hosts with their real configurations, and energy consumption at different workloads are summarized in TABLE II. The energy consumed by the gateway is 20W, and the network bandwidth of each host is 1 GB/s. We divide D2M services based on CPU and memory requirements. The limits of these services is based on the commercial amazon web service, e.g., HighCPU-intensive, LargeCPU-intensive, Small-CPU-intensive, and MicroCPU-intensive, as summarized in TABLE III.

Our experiments use real workload traces from real servers that are available publicly in the CloudSim simulator. The CPU utilization was simulated based on the CoMon project's data, a monitoring infrastructure for PlanetLab [27]. This dataset contains bandwidth, CPU utilization, and memory of more than 1000 hosts situated in 500 places around the world. However, for our experiments, we only need CPU utilization. The data has been collected over three days for every five minutes between 3rd March to 3rd April 2011, shown in TABLE IV. On 3rd March, 6th March, and 3rd April, there are 1052, 898, and 1464 CPU utilization data points. The CPU dataset's size for 3rd March is 1052 CPUs, 6th March is 898 CPUs, and 3rd April is 1464 CPUs. The data is interpolated to generate CPU utilization for every second. The data have some peak CPU utilization and very low off-peak CPU utilization levels, thus satisfying our simulation requirements. We set

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/TII.2022.3153508, IEEE Transactions on Industrial Informatics

Algorithm 3: Energy and thermal based cost construction

Input: *HostList*, *services*, T_{lower}, T_{middle}
Output: Best Feasible Host List

BestFeasibleHostList ← *Null*;
for *Host:HostList* **do**
  if *(Service resources requirements meets available host resources)* **then**
    CurrentHostCpuUtil ← `GetHostCpuUtil(Host, service);`
  if $(T_{lower} \leq$ *CurrentHostCpuUtil* $\leq T_{middle})$ **then**
    FeasibleHostList ← ∪ *Host*;
for *(FeasibleHost:FeasibleHostList)* **do**
  if *(Feasible Host meets service requirements)* **then**
    Cost(FeasibleHost) ← E^G + $P_i^{host}\left(1+\frac{1}{CoP(T^{up}-\max_i \sum_{k=1}^{m} d_{i,k}P_k^{host})}\right)$;
    BestFeasibleHostList ← ∪ *FeasibleHost*
return *BestFeasibleHostList*

TABLE II: Energy Consumption (W) and Characteristics of the Hosts

Server	Fujitsu M1	Fujitsu M3	Hitachi TS10	Hitachi SS10
0%	13.3	12.4	37	36
10%	18.3	16.7	39.9	38.8
20%	21.1	19.4	43.2	41.2
30%	23.4	21.4	48.8	46.3
40%	26.5	23.4	48.8	46.3
50%	29.6	26.1	52.8	49.4
60%	34.7	29.7	57.8	53.1
70%	40.7	34.8	65.1	58.8
80%	40.7	34.8	65.1	58.8
90%	46.8	41	73.8	64.2
100%	60	51.2	85.2	69.7
Server	Xeon 1230	Xeon 1230	Xeon 1280	Xeon 1280
Cores	4	4	4	4
CPU	2.7GHz	3.5 GHz	3.5 GHz	3.6 GHz
Memory	8GB	8GB	8GB	8GB

TABLE III: D2M Services

Feature	High CPU	Large CPU	Small CPU	Micro CPU
MIPS	2500	2000	1000	500
Memory	4050	3750	1700	613

TABLE IV: Workload dataset

Date	03-Mar	06-Mar	03-Apr
Workload	1052	898	1463
SD(%)	17.09	16.83	16.55

TABLE V: Parameter settings.

R	C	T^{ini}	T^{supply}	T^{up}	B	N_0	P_j
0.35 K/W	340 J/W	318 K	25°C	70°C	20 MHz	-113 dBm	31 dBm

other parameters that are useful in this simulation shown in TABLE V.

We also assume that hosts are arranged in a rack layout, and eight racks are arranged in a zone. Each zone laid 4×2 rows, and each rack has 10 hosts [21]. In each zone heat recirculation effect exists, which is negligible across the zone. Thus, we do not consider the heat recirculation effect in this paper. The heat distribution matrix **d** that represents the recirculation effect within the zone is adopted from work in [21].

B. Baseline Strategies

Four baseline strategies are considered: RANDOM, KMI-MR (K-means with a maximum ratio of CPU utilization to memory utilization), KMI-MP (K-mean with minimum the product of a CPU utilization and memory utilization), and PABFD (Power-aware Modified Best Fit Decreasing):
RANDOM: The D2M services are allocated to randomly selected hosts. This is the most intuitive approach. It does not consider the energy and thermal status of the host.
KMI-MR: The k-mean clustering method is used with a maximum ration of CPU utilization to memory utilization. It is only considered the energy consumption status of hosts [24].
KMI-MP: The k-mean clustering method is used with a Minimum the product of a CPU utilization and a memory utilization. [24].
PABFD: It is only considered CPU utilization of hosts for resource allocation to minimize energy consumption [28].

C. Cloud Efficiency Metrics

To evaluate the effectiveness of framework with schemes, results are compared with existing baseline approaches by using different metrics. The following subsection will discuss these metrics.

1) SLA Violation Metric: In the cloud data center, SLA defines quality attributes such as Quality of Service that cloud manufacturing services providers provide to cloud services users. SLA metrics are used to measure the performance characteristics of the service objects. The computing performance of SO-hosts are reduced over time, which increases SLA violation. The value of SLA violations is vital for the energy-aware algorithms, and this metric is proposed by Beloglazov et al. [7].

2) Performance Metric: The main objective of Cloud manufacturing-service providers is to minimize overall operational and cooling costs by consolidating VMs or services into a minimal number of servers. On the other hand, cloud service users focus on performance of services, which should not be affected by the consolidation process. Thus, cloud manufacturing-service providers seek to reduce energy consumption without violating SLA. Therefore, a combined energy consumption (E) and SLA violation (SLAV) metric called ESV metric is proposed by Beloglazov et al. [7] to analyze operational cost along with SLA. This metric is defined as *ESV = E * SLAV*.

3) Migration Metric: It is crucial to estimate the overhead of dynamic scheduling caused due to migration and workload consolidation. Therefore, we consider migration metrics to visualize performance degradation during the migration of services. Hence, the number of migrations should be minimized to reduce the overhead and SLA violations.

D. Analysis of Results

Fig. 3 illustrates the comparisons of total energy consumption among *KMI-MR*, *KMI-MP*, *PABFD*, *RANDOM*, and *MCU-ETV* schemes. It can be observed that the *MCU-ETV* scheme depicts the least energy consumption, and the *RANDOM* algorithm portrays the highest energy consumption. Moreover, the superiority of the *MCU-ETV* scheme gradually diminishes with the growing workloads of the data center.

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/TII.2022.3153508, IEEE Transactions on Industrial Informatics

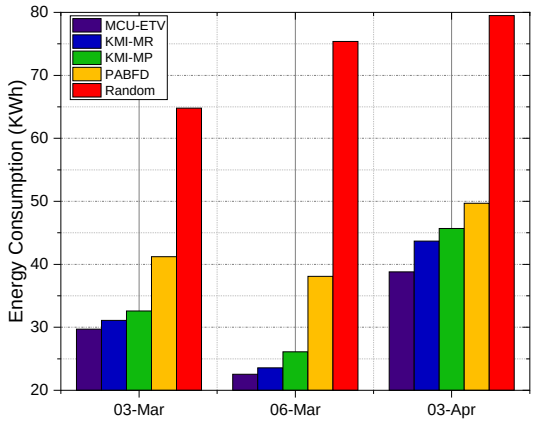

Fig. 3: The energy consumption of hosts compared with baseline strategies

The energy consumption of given workload trace by the approaches *KMI-MR*, *KMI-MP*, *PABFD*, *RANDOM*, and *MCU-ETV* are 43.2 kWh, 41.1 kWh, 49.3 kWh, 76.7 kWh, and 32.5 kWh respectively. In other words, the proposed scheme (*ETV*) consume 24%, 20%, 34%, and 57% less than *KMI-MR*, *KMI-MP*, *PABFD*, and *RANDOM*, respectively. If we estimate the average of energy consumption of all four baseline approaches then we can observe that the proposed scheme consume 33.75% less energy than the average energy consumption of all four baseline. The higher energy efficiency of the *MCU-ETV* scheme is attributed to: (1) minimizing the number of active hosts using the *MCU* scheme, and (2) the reduction of cooling energy consumption by efficiently allocating resources to D2M services, and the effects of heat recirculation using *ETV* scheme.

Considering both Figs. 4 and 5, we observe that the increasing value of a parameter c has a negative impact on energy consumption. However, increasing the value of a parameter c (safety parameter) has a positive impact on SLA violation. Hence, these two figures demonstrate that the value of c plays a significant role in balancing the tradeoff between energy and SLA.

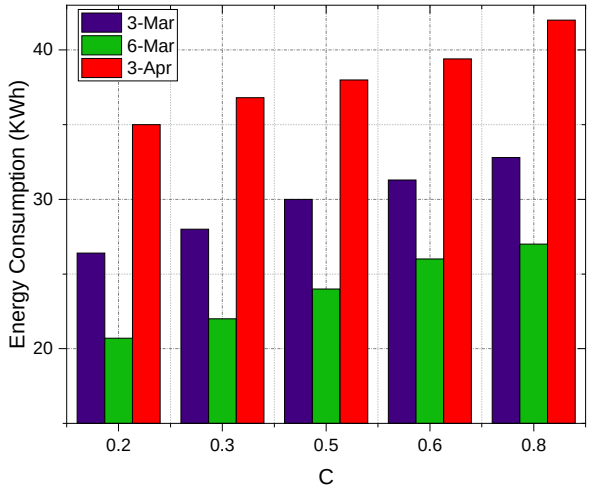

Fig. 4: Energy consumption of hosts by using the proposed scheme at different values of c compared with different workloads.

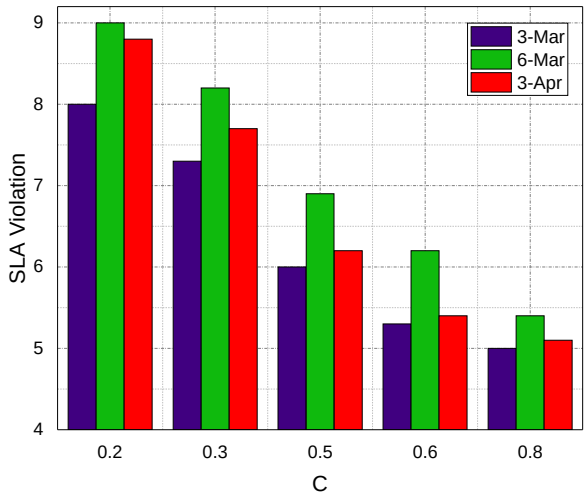

Fig. 5: SLA violation by using proposed scheme at different values of c compared with different workloads.

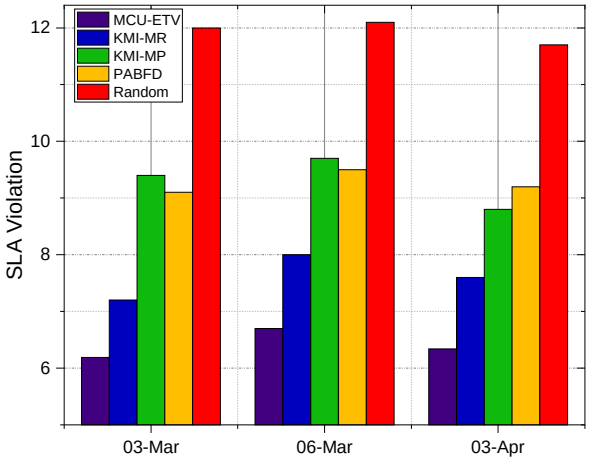

Fig. 6: SLA violation compared with baseline strategies

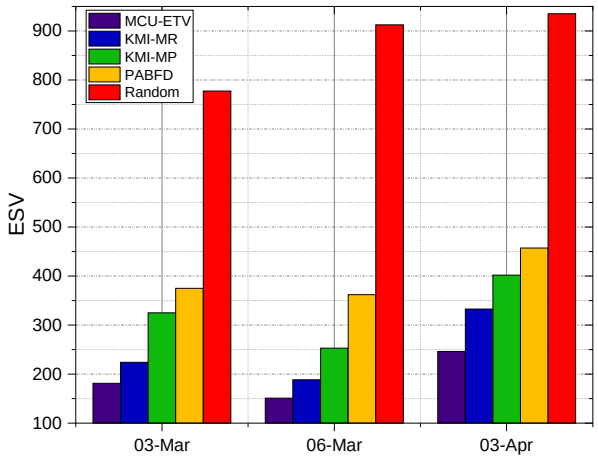

Fig. 7: The Performance metric compared with baseline strategies

The aggressive migration of D2M services leads to a high SLA violation in a cloud data center. Fig. 6 compares the percentage of SLA violation of the proposed scheme with *KMI-MR*, *KMI-MP*, *RANDOM*, and *PABFD*. It is observed

that the SLA violation using the proposed *MCU-ETV* scheme is 16%, 36%, 33.6%, and 51% less than *KMI-MR*, *KMI-MP*, *PABFD*, and *RANDOM*, respectively. If we estimate the average of SLA violation of all four baseline approaches, we observe that the proposed scheme violates 34.1% less SLA than the average SLA violation of all four baselines. Hence, the proposed scheme is capable of minimizing SLA violations with better performance due to its host classification technique.

Cloud manufacturing-services provider aims to maximize the overall performance, minimize energy consumption and SLA violation. Therefore, ESV is used to evaluate the operation cost of cloud data centers along with SLA. The lower, the better. Fig. 7 demonstrates the comparisons of ESV among *KMI-MR*, *KMI-MP*, *RANDOM*, *PABFD*, and *MCU-ETV* schemes using different workloads. We can find that the *MCU-ETV* scheme can achieve the lowest ESV, followed by *KMI-MR*, *KMI-MP*, *PABFD*, and *RANDOM*.

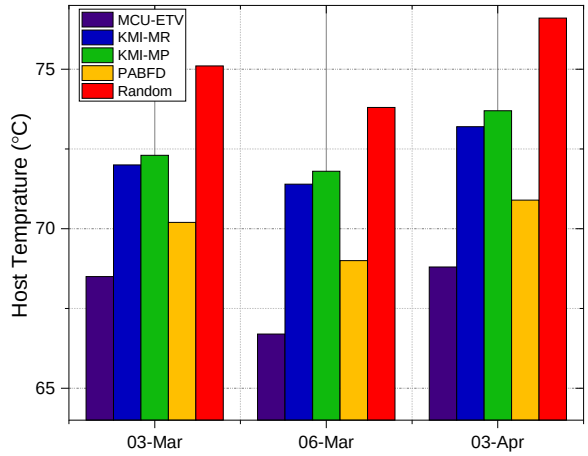

Fig. 8: The temperature of hosts compared with baseline strategies.

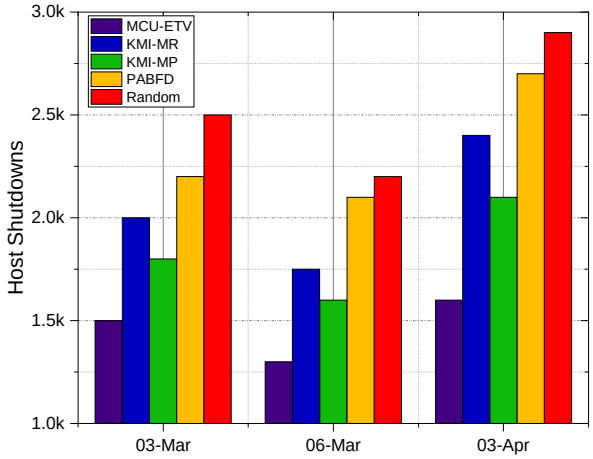

Fig. 9: The host shutdowns compared with baseline strategies.

Fig. 8 compares host temperature among *KMI-MR*, *KMI-MP*, *RANDOM*, *PABFD*, and *MCU-ETV* schemes. It can be observed that similar to the general trend of energy consumption, the host temperature also grows with the increasing workloads in the cloud data center. The temperature of given workload trace by the approaches *KMI-MR*, *KMI-MP*, *PABFD*, *RANDOM*, and *MCU-ETV* are 70.7 °C, 71 °C, 68.3 °C, and 65.8 °C, respectively. In other words, the obtained temperature using the proposed scheme (*MCU-ETV*) is 6.9%, 7.3%, 3.3%, and 9.9% less than *KMI-MR*, *KMI-MP*, *PABFD*, and *RANDOM*, respectively. If we estimate the average of temperature of all four baseline approaches then we can observe that the obtained temperature using proposed scheme 6.9% less than the average temperature obtained using all four baseline. The *MCU-ETV* scheme depicts the lowest temperature and never exceeds the upper temperature (T^{up}) due to its *ETV* resources allocation approach, as a result, maximizes the resource utilization and minimizes the cooling cost.

The hosts are reactivated for allocating new D2M services and shutdown when detected as idle. Fig. 9 shows that the *MCU-ETV* minimizes host reactivation 23%, 16%, 31%, and 40% compared to *KMI-MR*, *KMI-MP*, *PABFD*, and *RANDOM*, respectively. In our simulation, we use only 800 hosts. However, more than 800 hosts are shutdown due to host reactivation. We can find that the *MCU-ETV* reduces host reactivation more than baseline strategies. It indicates that the superiority of the *MCU-ETV* scheme is more evident with the increasing workload on each host; this is because of the advantages of the *MCU* approach.

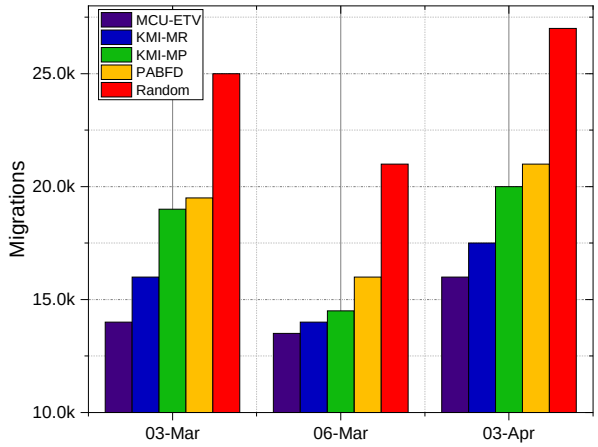

Fig. 10: The migration of services is compared with baseline strategies.

The migration metric is vital for the real environment of cloud computing technology, which involves a considerable bandwidth cost for live migrations of VMs or services. The high number of VM or service migrations directly influences the SLAs and the performance of cloud services. Simulation result of the number of migrations using the proposed scheme is obtained. Fig. 10 shows that the *MCU-ETV* minimizes the total number of migrations compared to *KMI-MR*, *KMI-MP*, *PABFD*, and *RANDOM* algorithms.

VI. Conclusion

The rapidly growing demand for D2M services in CCE has led to enormous energy consumption. This paper has investigated how to improve energy efficiency through energy-efficient resource allocation for CCE. This has been mathematically described as a constrained optimization problem. Due to the significant complexity, the optimization problem has been solved using a metaheuristic approach to improve energy efficiency, thus reducing SLA violation and complexity.

A two-phase framework has been presented with algorithm implementation for a solution to the optimization problem. In the first phase, the *MCU* scheme has been developed to classify hosts into four groups by using the least median square regression technique. In the second phase, an *ETV* scheme has been developed to search an energy-efficient host for resource allocation to D2M service through a metaheuristic search procedure. Simulations have been conducted with the CloudSim simulator. Experiments with real-world data-traces have shown that the proposed framework substantially minimizes energy consumption, host temperature, and SLA violation as compared with existing baseline approaches. Future research can expand the scope of investigation of GPU model and the Energy Reuse Effectiveness (ERE) metric in the cloud data center. We will embed this model into our framework for energy-efficient resource allocation.

VII. Acknowledgement

This work was supported in part by the Shenzhen Science and Technology Research and Development Foundation (JCYJ20190806143418198), the Fundamental Research Funds for the Central Universities (Grant No. HIT.OCEF.2021007) and in part by the Australian Research Council under the Discovery Projects Scheme (Grant No. DP170103305). Rahul Yadav is the Corresponding Author.

References

[1] S. Yin, X. Li, H. Gao, and O. Kaynak, "Data-based techniques focused on modern industry: An overview," *IEEE Transactions on Industrial Electronics*, vol. 62, no. 1, pp. 657–667, 2014.

[2] "Industrial iot market (iiot) by device and technology: Global forecast to 2023," https://www.marketsandmarkets.com/Market-Reports/industrial-internet-of-things-market-129733727.html, 2018.

[3] R. Chaudhary, G. S. Aujla, N. Kumar, and J. J. Rodrigues, "Optimized big data management across multi-cloud data centers: Software-defined-network-based analysis," *IEEE Communications Magazine*, vol. 56, no. 2, pp. 118–126, 2018.

[4] K. Kaur, T. Dhand, N. Kumar, and S. Zeadally, "Container-as-a-service at the edge: Trade-off between energy efficiency and service availability at fog nano data centers," *IEEE wireless communications*, vol. 24, no. 3, pp. 48–56, 2017.

[5] H. Li, K. C. Chan, M. Liang, and X. Luo, "Composition of resource-service chain for cloud manufacturing," *IEEE Transactions on industrial informatics*, vol. 12, no. 1, pp. 211–219, 2016.

[6] B. Lin, F. Zhu, J. Zhang, J. Chen, X. Chen, N. Xiong, and J. Lloret, "A time-driven data placement strategy for a scientific workflow combining edge computing and cloud computing," *IEEE Transactions on Industrial Informatics*, 2019.

[7] 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," *Concurrency and Computation: Practice and Experience*, vol. 24, no. 13, pp. 1397–1420, 2012.

[8] R. Yadav, W. Zhang, H. Chen, and T. Guo, "Mums: Energy-aware vm selection scheme for cloud data center," in *Database and Expert Systems Applications (DEXA), 2017 28th International Workshop on*, pp. 132–136. IEEE, 2017.

[9] R. Yadav, W. Zhang, K. Li, C. Liu, M. Shafiq, and N. K. Karn, "An adaptive heuristic for managing energy consumption and overloaded hosts in a cloud data center," *Wireless Networks*, pp. 1–15, 2018.

[12] M. Wahlroos, M. Pärssinen, J. Manner, and S. Syri, "Utilizing data center waste heat in district heating–impacts on energy efficiency and prospects for low-temperature district heating networks," *Energy*, vol. 140, pp. 1228–1238, 2017.

[10] R. Yadav and W. Zhang, "Mereg: Managing energy-sla tradeoff for green mobile cloud computing," *Wireless Communications and Mobile Computing*, vol. 2017, 2017.

[11] Y.-K. Lin and C. S. Chong, "Fast ga-based project scheduling for computing resources allocation in a cloud manufacturing system," *Journal of Intelligent Manufacturing*, vol. 28, no. 5, pp. 1189–1201, 2017.

[13] A. M. Al-Qawasmeh, S. Pasricha, A. A. Maciejewski, and H. J. Siegel, "Power and thermal-aware workload allocation in heterogeneous data centers," *IEEE Transactions on Computers*, vol. 64, no. 2, pp. 477–491, 2015.

[14] Y. Mhedheb, F. Jrad, J. Tao, J. Zhao, J. Kołodziej, and A. Streit, "Load and thermal-aware vm scheduling on the cloud," in *International Conference on Algorithms and Architectures for Parallel Processing*, pp. 101–114. Springer, 2013.

[15] R. Yadav, W. Zhang, K. Li, C. Liu, and A. A. Laghari, "Managing overloaded hosts for energy-efficiency in cloud data centers," *Cluster Computing*, pp. 1–15, 2021.

[16] A. Marcel, P. Cristian, P. Eugen, P. Claudia, T. Cioara, I. Anghel, and S. Ioan, "Thermal aware workload consolidation in cloud data centers," in *2016 IEEE 12th international conference on intelligent computer communication and processing (ICCP)*, pp. 377–384. IEEE, 2016.

[17] R. Yadav, W. Zhang, O. Kaiwartya, H. Song, and S. Yu, "Energy-latency tradeoff for dynamic computation offloading in vehicular fog computing," *IEEE Transactions on Vehicular Technology*, 2020.

[18] Z. Hong, W. Chen, H. Huang, S. Guo, and Z. Zheng, "Multi-hop cooperative computation offloading for industrial iot-edge-cloud computing environments," *IEEE Transactions on Parallel and Distributed Systems*, 2019.

[19] S. Guo, J. Liu, Y. Yang, B. Xiao, and Z. Li, "Energy-efficient dynamic computation offloading and cooperative task scheduling in mobile cloud computing," *IEEE Transactions on Mobile Computing*, vol. 18, no. 2, pp. 319–333, 2018.

[20] S. Ilager, K. Ramamohanarao, and R. Buyya, "Etas: Energy and thermal-aware dynamic virtual machine consolidation in cloud data center with proactive hotspot mitigation," *Concurrency and Computation: Practice and Experience*, vol. 31, no. 17, p. e5221, 2019.

[21] Q. Tang, S. K. S. Gupta, and G. Varsamopoulos, "Energy-efficient thermal-aware task scheduling for homogeneous high-performance computing data centers: A cyber-physical approach," *IEEE Transactions on Parallel and Distributed Systems*, vol. 19, no. 11, pp. 1458–1472, 2008.

[22] S. Zhang and K. S. Chatha, "Approximation algorithm for the temperature-aware scheduling problem," in *2007 IEEE/ACM International Conference on Computer-Aided Design*, pp. 281–288. IEEE, 2007.

[23] J. D. Moore, J. S. Chase, P. Ranganathan, and R. K. Sharma, "Making scheduling" cool": Temperature-aware workload placement in data centers." in *USENIX annual technical conference, General Track*, pp. 61–75, 2005.

[24] Z. Zhou, J. Abawajy, M. Chowdhury, Z. Hu, K. Li, H. Cheng, A. A. Alelaiwi, and F. Li, "Minimizing sla violation and power consumption in cloud data centers using adaptive energy-aware algorithms," *Future Generation Computer Systems*, 2017.

[25] 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*, vol. 41, no. 1, pp. 23–50, 2011.

[26] R. Yadav, W. Zhang, O. Kaiwartya, P. R. Singh, I. A. Elgendy, and Y.-C. Tian, "Adaptive energy-aware algorithms for minimizing energy consumption and sla violation in cloud computing," *IEEE Access*, vol. 6, pp. 55 923–55 936, 2018.

[27] K. Park and V. S. Pai, "Comon: a mostly-scalable monitoring system for planetlab," *ACM SIGOPS Operating Systems Review*, vol. 40, no. 1, pp. 65–74, 2006.

[28] A. Beloglazov, J. Abawajy, and R. Buyya, "Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing," *Future generation computer systems*, vol. 28, no. 5, pp. 755–768, 2012.