A Self-Adaptive Approach for Managing Applications and Harnessing Renewable Energy for Sustainable Cloud Computing

Minxian Xu, *Member, IEEE*, Adel Nadjaran Toosi, *Member, IEEE*, and Rajkumar Buyya, *Fellow, IEEE*

Abstract—Rapid adoption of Cloud computing for hosting services and its success is primarily attributed to its attractive features such as elasticity, availability and pay-as-you-go pricing model. However, the huge amount of energy consumed by cloud data centers makes it to be one of the fastest growing sources of carbon emissions. Approaches for improving the energy efficiency include enhancing the resource utilization to reduce resource wastage and applying the renewable energy as the energy supply. This work aims to reduce the carbon footprint of the data centers by reducing the usage of brown energy and maximizing the usage of renewable energy. Taking advantage of microservices and renewable energy, we propose a self-adaptive approach for the resource management of interactive workloads and batch workloads. To ensure the quality of service of workloads, a brownout-based algorithm for interactive workloads and a deferring algorithm for batch workloads are proposed. We have implemented the proposed approach in a prototype system and evaluated it with web services under real traces. The results illustrate our approach can reduce the brown energy usage by 21 percent and improve the renewable energy usage by 10 percent.

Index Terms—Cloud data centers, renewable energy efficiency, QoS, microservices, brownout

1 Introduction

Today's society and its organizations are becoming ever-increasingly dependent upon information and communication technologies (ICT) with software systems, especially web systems, largely hosted on cloud data centers. Clouds offer an exciting benefit to enterprises by removing the need for building own Information Technology (IT) infrastructures and shifting the focus from the IT and infrastructure issues to core business competence. Apart from the infrastructure, elasticity, availability, and pay-as-you-go pricing model are among many other reasons which led to the rise of cloud computing [1]. This massive growth in cloud solutions demanded the establishment of huge number of data centers around the world owned by enterprises and large cloud service providers such as Amazon, Microsoft, and Google to offer their services [2].

However, data centers hosting cloud services consume a large amount of electricity leading to high operational costs and high carbon footprint on the environment [3]. ICT sector nowadays consumes approximately 7 percent of the global electricity, and it is predicted that the share will increase up to 13 percent by 2030 [4]. Among this, the operation of data centers accounts for one of the fastest growing sources of carbon dioxide emissions [5]. In 2013, U.S. data centers solely consumed an estimated 91 billion kWh of electricity (equivalent to the two-year power consumption of all households in New York City) and this is projected to reach 140 billion kWh by 2020 [6].

One of the main sources of energy inefficiencies in data centers is represented by servers, which are often utilized between 10 to 50 percent of their peak load [7]. This issue is amplified by the fact that server machines in data centers do not exhibit complete energy proportionality, that is, servers do not consume electricity in proportion to their load [8]. Even though, cloud providers use techniques such as dynamic consolidation of virtual machines (VMs) [9] to achieve energy saving and avoid underutilized servers, energy is still being wasted if cloud consumers hold many idle or under-utilized virtual machines up and running. RightScale[1] states that the cloud consumers waste between 30-45 percent of their total cloud consumption [10].

In this regards, microservice architectures and technologies such as containers [11], that are steadily gaining adoption in industrial practice, provide a leap towards more efficient utilization of cloud resources. Containerization allows for higher resource utilization and reduction of cost by running multiple services on the same VM and providing a fine-grained control on resources. Traditional web applications are considered to be migrated from monolithic structure to microservices architecture [12]. In this paper,

- *Minxian Xu is with Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, Shenzhen 518055, China. E-mail: mx.xu@siat.ac.cn.*
- *Adel Nadjaran Toosi is with the Faculty of IT, Monash University, Clayton, VIC 3800, Australia. E-mail: adel.n.toosi@monash.edu.*
- *Rajkumar Buyya is with Cloud Computing and Distributed Systems (CLOUDS) Lab, School of Computing and Information Systems, University of Melbourne, Parkville VIC 3010, Australia. E-mail: rbuyya@unimelb.edu.au.*

*Manuscript received 17 Jan. 2020; revised 4 July 2020; accepted 4 Aug. 2020. Date of publication 7 Aug. 2020; date of current version 8 Dec. 2021. (Corresponding author: Minxian Xu.)
Recommended for acceptance by D. Gizopoulos.
Digital Object Identifier no. 10.1109/TSUSC.2020.3014943*

1. https://www.rightscale.com/

we take advantage of container technology to reduce the energy consumption of the system.

Brownout [13] is a self-adaptive approach to manage resources and applications in cloud computing systems. With brownout, the optional parts of applications can be dynamically deactivated/activated according to the system status. A control knob, called dimmer, is applied to represent the degree that brownout should be performed on the application's optional parts. Brownout can also be utilized for the management of microservices to improve resource usage in data centers. However, generally, there are always trade-offs that should be balanced, e.g., balancing energy consumption and system performance.

Apart from self-contained microservices, renewable energy is another solution gaining momentum to address energy consumption concerns (i.e., the carbon footprint) of cloud computing. In response to the climate change concerns and economic stimulus, many research initiatives have been launched to promote renewable energy use to power cloud data centers in recent years [14], [15], [16]. Many cloud providers also work on this goal by generating their own renewable energy or drawing power from a nearby renewable power plant. For example, in January 2018, AWS achieved 50 percent renewable energy usage by investing in clean energy activities including a commercial-scale wind farm in North Carolina.[2]

Renewable energy systems are shown to be extremely effective in reducing dependence on finite fossil fuels and decreasing environmental impacts. Currently, all modern inverters have interfaces to select the source of power for either grid or batteries. However, power generation using photovoltaic (PV) solar energy can only be done during the daytime and the amount of produced power depends on the weather and geographical location of the data center. A large solar power system with a sufficiently large battery setup to fully support workload is not economical. Therefore, we are looking into approaches to match energy consumption with the availability of renewable energy. In these approaches, cloud resource management systems need to support methods that allocate resources and schedule applications execution by preferring to finish them during the time when renewable energy is available while at the same time need to make sure that user QoS requirements are honored.

A fundamental problem of powering data centers with renewable energy sources is how to optimize the use of renewable energy. Powering data centers with renewable energy sources such as solar or wind is challenging as these sources of energy are non-dispatchable and are not always available due to their fluctuating nature. Thus, in this work, we aim to address this challenge through an optimization problem of maximizing renewable energy usage while minimizing the usage of brown energy.

In this work, we address the research problem as: by predicting the amount of renewable energy, determining when to use brownout for interactive workloads and when to defer batch workloads, when to consolidate VMs to fewer hosts and scale hosts to maximize the usage of renewable energy while ensuring the QoS of workloads. Based on the detailed survey [13], this research problem has not been addressed by previous work.

2. https://aws.amazon.com/about-aws/sustainability/

The key *contributions* of the paper are:

- Provide a perspective model for multi-level adaptive resource scheduling to manage workloads and renewable energy;
- Propose a self-adaptive approach for interactive workloads and batch workloads to ensure their QoS by considering the predicted renewable energy at Denver city;
- Implement a prototype system derived from the perspective model and the proposed approach on a small-scale testbed;
- Evaluate the performance of the self-adaptive approach in the proposed prototype system for web services.

The rest of the paper is organized as follows: Section 2 discusses the related work for managing energy in the cloud computing environment. Section 3 depicts the system model of our proposed approach, followed by modeling and problem statement in Section 4. The scheduling algorithm with renewable energy is introduced in Section 5. Section 6 provides the detailed information about the implementation of our prototype system, and Section 7 shows the evaluation results of our proposed approach under our prototype system. Finally, conclusions along with the future directions are given in Section 8.

2 Related Work

In this section, we discuss related research in the context of the dominant energy efficient approaches based on resource scheduling, brownout approaches, cooling-aware data center energy management and resource scheduling with renewable energy.

2.1 DVFS and VM Consolidation

A large body of research on the energy efficiency of data centers has been dedicated to the optimization techniques to reduce the energy consumption of servers within a data center using technologies such as dynamic voltage and frequency scaling (DVFS) and VM consolidation [17], [18]. Liu *et al.* [19] proposed a heuristic algorithm for big data task scheduling based on thermal-aware and DVFS-enabled techniques to minimize the total energy consumption of data centers. Kim *et al.* [18] modeled real-time service as VM requests and proposed several DVFS algorithms to reduce energy consumption for the DVFS-enabled cluster. Cheng *et al.* [27] proposed a heterogeneity-aware task assignment approach to improve the overall energy consumption in a heterogeneous Hadoop cluster without sacrificing job performance. Teng *et al.* [20] presented a set of heuristic algorithms by taking advantage of DVFS and VM consolidation together for batch-oriented scenarios. Nguyen *et al.* [21] introduced a virtual machine consolidation algorithm with multiple usage prediction to improve the energy efficiency of cloud data centers.

Our work differs from these efforts in several perspectives: (1) none of these DVFS-based and VM consolidation approaches can function well if the whole system is overloaded; (2) none of them put the efforts to schedule the mixed type of workloads; (3) none of them applied the renewable energy to power their systems.

2.2 Brownout

Xu *et al*. [13] proposed a survey and taxonomy on brownout-based approaches, which summarized the application of brownout in cloud computing systems for different optimization objectives. Tomas *et al*. [28] applied brownout to address the load balancing issues in clouds. Shahrad *et al*. [29] introduced a practical pricing model for brownout system and aimed to increase the utilization of the cloud infrastructure by incentivizing users to dampen their usage fluctuations. Trade-offs exist when it comes to applying brownout and deactivating microservices. For example, the trade-off between energy and revenue is investigated in [30]. In contrast, our optimization objective is managing the energy usage in cloud data centers.

Xu *et al*. [22], [31] presented brownout-based approaches to manage microservices and resources from the energy perspective. Hasan *et al*. [23] investigated the green energy and user experience trade-off in interactive cloud applications and proposed a controller to provide guarantees of keeping response time within the Service Level Agreement (SLA) range in the presence of green energy based on a brownout-enabled architecture. In contrast, this work advances the previous ones by: (1) managing the energy in a holistic way by considering multiple layers resource management, cooling power and mixed type of workloads; (2) incorporating renewable energy usage to reduce brown energy usage.

2.3 Holistic Management With Cooling

Due to the complexity of thermal modeling of data center operation, traditional approaches ignored the impacts of resource management techniques on the cooling power system of data centers. Recently, the holistic management of resources in which both computing and cooling energy are considered in the minimization of the overall consumption of energy has gained considerable attention from the community. Li *et al*. [24], for example, provided models capturing thermal features of computer room air conditioning (CRAC) unit of the data center and accordingly propose a VM scheduling algorithm to reduce data center energy consumption while it maintains the SLA violation in an acceptable range. In their work, resource scheduling happens on VM level and the workload type is batch. Al-Qawasmeh *et al*. [32] presented power and thermal-aware workload allocation in the heterogeneous cloud. They developed optimization techniques to assign the performance states of CPU cores (P-states) at the data center level to optimize the power consumption while ensuring performance constraints. Tang *et al*. [33] investigated the thermal-aware task scheduling for homogeneous HPC data center, which aims to minimize peak inlet temperature through task assignment, thus reducing the cooling power.

Unlike these efforts, our work considers the management of virtualized resources to optimize the resource usage. In addition, we consider the multiple layers resource scheduling and mixed types of workloads.

2.4 Renewable Energy

Studies have been investigated in the literature that focused on the optimization of on-site renewable energy use in data centers. Goiri *et al*. [16] presented a prototype of a green data center powered with solar panels, a battery bank, and a grid-tie which they have built as a research platform. They also describe their method, called GreenSwitch, for dynamically scheduling the interactive and batch workload and selecting the source of energy to use. GreenSwitch aims to minimize the overall cost of electricity while respecting the characteristics of the workload and battery lifetime constraints. In contrast, while their work focused on the resource scheduling at the application level, our work is a multiple layers scheduling approach that considers the application, VMs, and hosts. We also model the applications to be fitted into the brownout feature. In addition, our renewable energy prediction model based on support vector machine differs significantly from their model in which solar energy was predicted based on the last epoch.

Liu *et al*. [25] also focused on shifting workloads and matching renewable energy supply and demand in the data center. They schedule non-critical IT workload and allocates IT resources within a data center according to the availability of renewable power supply and the efficiency of the cooling system. They formulated the problem as a constrained convex optimization and aim to minimize the overall cost within the data center. Different from the optimization of the overall costs, we aim to optimize the energy perspective. Another difference is that we can optimize the power consumption by scheduling both interactive workloads and batch workloads, while [25] only optimizes the scheduling of batch workloads without optimizing interactive workloads.

To optimize onsite renewable energy use, in our previous work [26], we focused on microservices management problem as finite Markov Decision Processes (MDP). Similar to this work, the proposed method dynamically switches off non-mandatory microservices of the application to strike a balance between the workload execution and brown energy consumption. Following a greenness property value, it also suggests in what capacity the battery power should be consumed in each time slot. Different from that work, we consider joint management of both interactive workloads and batch workloads. We also cover the entire stack of resource scheduling including microservices, VMs and physical hosts. In addition, we test our system in a real testbed, while [26] only conducts simulation.

The current paper contributes to the growing body of work in the related area. Table 1 summarizes the comparison among the related work based on the key techniques, energy model, workload types and resource scheduling layers. Given the contributions of existing work, it is important to highlight the key difference between our proposed work and prior work. To be best of our knowledge, our work is the first to jointly manage interactive workloads based on brownout and batch workloads based on deferral approach. Prior work have only included one of these features. We also consider multiple layers resource scheduling based on microservices management, VM scheduling and host scaling with real testbed, which enables to form a truly integrated resource management.

3 System Model

In this section, we propose a system model for adaptive resource scheduling as shown in Fig. 1. We consider both

TABLE 1
Comparison for Related Work

Approach	Technique				Energy Model			Workloads Type		Resource Scheduling Layer	
	DVFS	VM Consolidation	Host Scaling	Brownout	Brown Energy	Renewable Enegy	Cooling Energy	Single	Mixed	Single Layer	Multiple Layer
Beloglazov et al. [17]		√	√		√			√			√
Kim et al. [18]	√				√			√		√	
Liu et al. [19]	√				√		√	√		√	
Teng et al. [20]	√	√	√		√			√			√
Nguyen et al. [21]		√	√		√			√			√
Xu et al. [22]		√	√	√	√			√			√
Hasan et al. [23]			√	√	√	√		√			√
Li et al. [24]		√			√		√	√			√
Beloglazov et al. [9]		√	√		√			√			√
Goiri et al. [16]			√		√	√	√		√	√	
Liu et al. [25]			√		√	√	√		√	√	
Xu et al. [26]				√	√	√		√		√	
Our Approach		√	√	√	√	√	√		√		√

interactive workloads and batch workloads in the application layer and consider green and brown energy together in the energy supply layer.

In the users layers, users submit their service requests to the system. The users can define QoS constraints for the submitted requests, such as budget and deadline. The submitted requests are forwarded to the application layer. From the service providers' viewpoint, these workloads generated by users are processed by the applications hosted in the cloud infrastructure.

We consider two types of applications: the interactive application (such as web application), and batch application. The interactive application should be executed as soon as possible to ensure the QoS. We consider the interactive application to support brownout, thus the microservices of the interactive applications can be identified as optional or mandatory. The optional ones can be deactivated to save resource usage, if deemed necessary. For the batch application, the workloads can be deferred for execution if their deadline is ensured.

Applications provided by service providers to offer services for users are managed by the application hosting engines, such as Docker [34] or Apache Tomcat. Applications can be deployed on either virtualized platform (virtual resources) or cloud infrastructure (physical resources). The host application engine can be container-based management platforms, e.g., Docker Swarm [35], Kubernetes [36] or Mesos [37], which provide management of container-based applications. The application containing multiple microservices can be deployed on multiple VMs. The virtualized platform manages the virtualized resources, for example, the Virtual Machines managed by VMware [38]. As for the resource allocation and provisioning in cloud infrastructure, they can be managed by infrastructure management platform, e.g., OpenStack [39]. Multiple VMs can be deployed on multiple hosts.

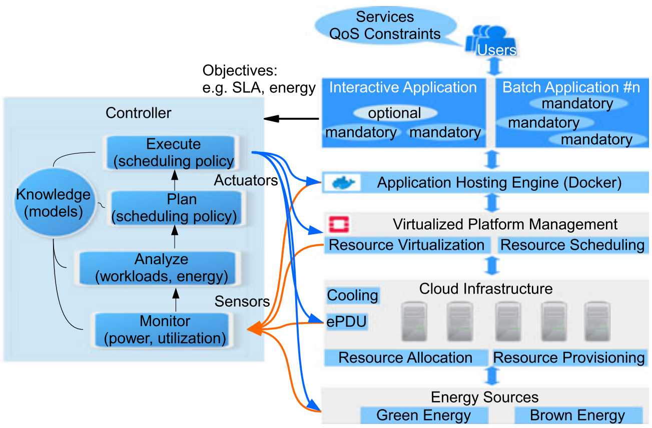


Fig. 1. Perspective model.

The bottom layer is the energy supply layer, which provides the mixed of energy sources for powering the system. The brown energy comes from a coal-based thermal power plant, which has high carbon footprint. The green energy comes from the renewable energy, such as solar power.

To support the resource provision, monitor and allocation in the system, a controller is required based on Monitor, Analyze, Plan, Execute and Knowledge (MAPE-K) architecture model and fits into the feedback loop of the MAPE-K process, which has modules including *Monitor*, *Analyze*, *Plan* and *Execute* to achieve the adaptation process in cloud computing system [40], [41]. Sensors and Actuators are used to establish interactions with the system. Sensors gather the information from different levels in the system, including application hosting engine, virtualized platform, cloud infrastructure, and energy usage. The sensors can be the devices attached to hardware, e.g., power meter. The collected information is provided to the Monitor module.

The Analyze module analyzes the received information from the Monitor module, and the Plan module makes decisions for applying scheduling policies, in which the scheduling policies are implemented. According to the decisions, the Execute module schedules resources via actuators on the application hosting engine and the virtualized platform to enable/disable optional microservices in interactive applications or defer the workloads of batch applications to be supplied by renewable energy. These operations can be fulfilled via the Application Programming Interfaces (APIs) provided by the application hosting engine or the virtualized platform.

The Knowledge pool in MAPE-K model is applied to store the predefined objectives (energy efficiency or SLA constraints) and trade-offs (e.g., trade-offs between energy and SLA). The rules in Knowledge pool, such as SLA rules, can be updated according to resource scheduling algorithms. The Knowledge pool also contains models like predicting the supplied amount of renewable energy, which can be used by scheduling algorithms. .

In the following sections, we will introduce our proposed approach and the prototype system that is derived from this perspective model.

4 Problem Modeling

In this section, we will discuss our modeling and optimization problem, including the power consumption model, workloads model, and optimization objectives.

4.1 Power Consumption

4.1.1 Server Power Consumption

The server power model is derived from [42], which is based on the average CPU utilization.[3] As we consider multiple layers scheduling, the utilization of hosts, VM and microservices are modeled

$$P_i^s = \begin{cases} P_i^{idle} + \theta_t \sum_{j=1}^{w_i} U_{i,j}^{vm} \times P_i^{dynamic} & , w_i > 0 \\ 0 & , w_i = 0 \end{cases}, \tag{1}$$

where P_i^s is the power consumption of the host i in the data center, which is composed of two parts: the idle power P_i^{idle} and dynamic power $P_i^{dynamic}$. The dynamic power part is related to the VM utilization on the host. If there is no VM running on the host, it means the host can be switched into the low power mode (S-State) and consume low energy. The w_i represents the number of VMs deployed on host i. $U_{i,j}^{vm}$ represents the utilization of jth VM on host i. The θ_t is the dimmer value of brownout at time interval t, which represents the percentage of resource utilization provisioned to the active microservices. For instance, if $\theta_t = 0.8$, it means 20 percent utilization will be reduced by deactivating microservices. The dimmer value is calculated based on the number of overloaded hosts and more details will be given in Section 5.2.

The utilization of VM is the sum of microservices utilization running on the VM, which is modeled as

$$U_{i,j}^{vm} = \sum_{k=1}^{A_j} U_{j,k}^{ms}, \tag{2}$$

where the ms is the id of microservice and A_j is the number of microservices. Since CPU computation is main power consumption component of servers, in our server model, we mainly focus on the power draw by the CPU utilization.

4.1.2 Cooling Power Consumption

We consider the data center thermal control is managed by Computer Room Air Condition (CRAC) system. The system contains multiple CRAC units, which transfer cold air to the hosts to reduce hotspots. Based on server power consumption and cooling efficiency, we can calculate the power consumed P_i^c by cooling equipment for host i as

$$P_i^c = \frac{P_i^s}{CoP(T_{sup})}. \tag{3}$$

There are some complex cooling models [43], while they are beyond the scope of this paper. We use the model from HP lab data center [44] as follows:

$$CoP(T_{sup}) = 0.0068T_{sup}^2 + 0.0008T_{sup} + 0.458. \tag{4}$$

The CoP is a function to estimate the cooling efficiency of cold air supply temperature T_{sup} provided by cooling equipment, which is related to the target temperature that room is aimed to be maintained. The total power draw by the server part and the cooling part can be represented as

$$P_i = P_i^s + P_i^c. \tag{5}$$

The total power of data center with k servers

$$P_t = \sum_{i=1}^{k} P_i. \tag{6}$$

4.2 Workloads Model

In this work, we consider two types of workloads: (1) interactive workloads and (2) batch workloads. The interactive workloads are response time sensitive, thus these workloads should be executed immediately with the response time specified in the SLA, while the batch workloads can be deferred for execution as long as the deadline is satisfied.

Based on the different characteristics of these workloads, we assume that there are M interactive workloads, and the amount of interactive workload m at time t is denoted as $a_m(t)$, and consumes resource u_m. This is a general model and the amount of workloads can be derived from analytical models (e.g., M/M/k) or realistic traces. Thus, the demanded resources of interactive workload m at time t are $a_m(t) \times u_m$. Based on M/GI/1/PS model, the demanded resources can have a relationship with the target response time rt_m as $\frac{1}{\mu_m - \lambda_m(t)/(a_m(t) \times u_m)} \leq rt_m$, where $\lambda_m(t)$ is the arrival rate and μ_m is the mean service rate. Thus, the minimum resources for interactive workloads to satisfy rt_m is: $a_m(t) \times u_m = \frac{\lambda_m(t)}{\mu_m - 1/rt_m}$.

We also assume that there are N types of batch workloads, type n batch workloads have total demand B_n. Let $b_n(t)$ denote the amount of type n batch workloads at time interval t, with start time S_n, execution time E_n, and deadline D_n, consuming resource amount u_n. We use $b_n(t)^{'}$ to denote the original amount of batch workloads, the actual amount of workloads should add the workloads deferred from the previous time slot and minus the workloads that will be deferred to the next time slot. We use γ_n^{t-1} and γ_n^t to present the percentage of deferred workloads at time interval t and $t-1$ for type n batch workloads. Then we have $b_n(t) = \gamma_n^{t-1} \times b_n(t-1) + b_n(t)^{'} - \gamma_n^t \times b_n(t)^{'}$.

Therefore, the total CPU resource demands at t are

$$d(t) = \sum_m a_m(t) \times u_m + \sum_n b_n(t) \times u_n. \tag{7}$$

The value of $d(t)$ should be $0 \leq d(t) \leq D$, in which D is the maximum CPU resource capability of the system. To be noted, rather than constant, $d(t)$ is varied based on our proposed scheduling policy, for instance, CPU resource provisioned to interactive workloads can be adjusted by brownout mechanism (Equation (1)) and the batch workloads can be deferred based on system status (Equation (7)).

4.3 Optimization Objectives

We assume the scheduling period as T, and the time interval at which we schedule resources is denoted as t. We assume the available renewable energy at time t is R_t, which can be predicted by prediction approaches, e.g., machine algorithms. As the server power and cooling power is related to workloads, we use $d(t)^{'}$ to denote the power consumption resulted from the workload execution on servers, and $c(d(t)^{'})$ represents the cooling power resulted from the

3. In this work, we consider the CPU utilization as the main source of resource consumption.

workloads, thus $P_t = d(t)^{'} + c(d(t)^{'})$. Our optimization objective is modeled as

$$\begin{aligned} & min \sum_t (max(P_t - R_t, 0)) \\ & s.t. \quad 0 \leq d(t) \leq D,\ \forall t \\ & \qquad\quad 0 \leq \theta_t \leq 1,\ \forall t \\ & b_n(t) = \gamma_n^{t-1} \times b_n(t-1) + b_n(t)^{'} - \gamma_n^t \times b_n(t)^{'},\ \forall t \\ & \sum_t b_n(t) \leq B_n,\ \forall n, \end{aligned} \tag{8}$$

where aims to minimize the usage of brown energy by maximizing the usage of renewable energy. Our proposed solution makes schedule decisions of each time interval slot for interactive workloads and batch workloads based on the availability of renewable energy. Meanwhile, the constraints including maximum capacity D in the system, dimmer value θ_t, maximum total demand of batch workloads should be satisfied. As this optimization objective is convex in $d(t)$, so it can be solved efficiently.

Algorithm 1. Green-Aware Scheduling Algorithm

Input: host utilization U_i^t, utilization thresholds TU_{up}, TU_{low}
Output: brown energy usage $\sum_t (max(P_t - R_t, 0))$
1: $n_o^t = \sum_k (U_i^t > TU_{up})$
2: **for** $t \leftarrow 0$ *to* T **do**
3:   **if** $n_o^t > 0$ **then**
    `// take actions to minimize` $\sum_t (max(P_t - R_t, 0))$
4:     $S_i \leftarrow$ brownout algorithm for interactive workloads
5:     $T_j^D \leftarrow$ deferring algorithm for batch workloads
6:   **else if** $U_{avg}^t < TU_{low}$ **then**
    `// take actions to minimize` n_a
7:     run VM consolidation algorithm
8:     run host scaling algorithm
9:   **else**
10:     process iteractive workloads in normal mode
11:     porcess batch workloads in normal mode
12:   **end**
13: **end**

5 Scheduling With Renewable Energy

In this section, based on the problem modelling, we introduce our proposed scheduling algorithm with renewable energy for both interactive workloads and batch workloads.

5.1 Green-Aware Scheduling Algorithm

To schedule the interactive and batch workloads in an energy efficient manner by considering renewable energy, we propose a Green-aware scheduling algorithm, which is shown in Algorithm 1. During the observation period T, at each time interval t, the algorithm will first identify the number of overloaded hosts (line 1). If the overloading situation exists, the algorithm will manage the interactive workloads and batch workloads with different algorithms: brownout algorithm for interactive workloads (Algorithm 2) and deferring algorithm for batch workloads (Algorithm 3) to minimize brown energy usage (lines 4-5). These two algorithms will do the actions including deactivating microservices and deferring workloads to achieve the brown energy usage objective. Here we assume the interactive workloads utilize more CPU utilization than the batch workloads, so the interactive workloads are processed earlier to achieve better scheduling effects. If the system is not overloaded and the average utilization is below the underutilized threshold (line 6), the algorithm will apply VM consolidation algorithm derived from [17] (line 7) that consolidates VMs to the hosts that produce the minimum incrementation of energy consumption, and apply host scaling algorithm (Algorithm 4) to change the number of active hosts n_a. The motivation is that the idle servers will be switched into the low power mode to save energy (lines 7-8). If the system is running at the normal status, then the workloads will be executed in the normal fashion.

Algorithm 2. Brownout Algorithm for Interactive Workloads

Input: time interval t, the number of overloaded hosts n_o^t, the percentage of utilization from batch workloads ϵ, and the starting time and end time of the available renewable energy t_r^s, t_r^e
Output: deactivated microservices S_i
1: **for** *host i in the host list* **do**
2:   **if** $t < t_r^s \,||\, t > t_r^e$ **then**
3:     $\theta_t = \sqrt{\frac{n_o^t}{n}}$
4:     $U_i^r = \theta_t \times U_i^t$
5:     **if** $U_i^t > TU_{up}$ **then**
      `// take actions to minimize` $|U_i^r - U(S_i)|$
6:       find deactivated microservices S_i on host i
7:       deactivate the microservices
8:     **end**
9:   **else**
10:     **if** $R_t < P_t$ **then**
11:       $\theta_t = \frac{1}{1-\epsilon} \times \sqrt{\frac{R_t}{P_t}}$
12:       $U_i^r = \theta_t \times U_i^t$
      `// take actions to minimize` $|U_i^r - U(S_i)|$
13:       find deactivated microservices S_i on host i
14:       deactivate the microservices
15:     **end**
16:   **end**
17: **end**

5.2 Brownout Algorithm for Interactive Workloads

The pseudocode of the brownout algorithm for interactive workloads is shown in Algorithm 2. The algorithm schedules resources differently according to whether the renewable is available or not. The starting time and end time of available renewable energy are denoted as t_r^s and t_r^e respectively. 1) During the time when renewable energy is not available (line 2), the brownout is triggered, and the dimmer value is generated. The dimmer value θ_t is computed based on the severity of overloads in the system (line 3). With the dimmer value, the expected utilization reduction U_i^r on host i is computed (line 4). Then the algorithm selects a set of microservices S_i to deactivate, thus the utilization is reduced. The difference between the expected utilization reduction U_i^r and the sum of utilization of selected microservices $U(S_i)$ is minimized (lines 6-7). In lines 6-7, to minimize the difference, the microservices selection process sorts the microservices according to their utilization in a list, and finds the sublist which has the utilization that is closest to U_i^r. To be noted, the selection process will only search for the optional microservices, which means if there are not enough or no optional microservices for deactivation, only the available or no microservices will be selected. 2) When the renewable energy is available but less than the total required energy, the brownout is also triggered (line 10). The dimmer value is calculated based on the renewable energy R_t and required energy P_t as noted in line 11. Then the remaining steps are the same as in the first part of Algorithm 2, which finds the microservices and deactivates them (same as in lines 6-7). 3) When sufficient renewable energy is available, brownout will not be triggered.

5.3 Deferring Algorithm for Batch Workloads

Algorithm 3 shows pseudocode for processing the batch workloads. The batch workloads are executed when their start time S_j is coming (line 1). The workloads are processed based on the time period that the workloads are in. For the workloads which have the start time before the renewable energy start time t_r^s, the objective is to defer their execution to the time when the renewable energy is available while ensuring their deadlines (lines 2-12). 1). If the deadline is before t_r^s, it means the workload cannot be deferred to be processed by renewable energy, so the workload can be executed at t (lines 3-4). If the workload can be deferred, the algorithm defers its time with T_n^d, then the algorithm updates the workloads at time td, which equals to $t + T_n^d$. The deferred time T_n^d should satisfy the constraint, e.g., not failing the deadline, and the renewable energy is enough at td. If the constraints are satisfied, the algorithm updates the predicted power consumption at td. 2). When the start time of the workload is during the time when renewable energy is available and sufficient, the workload is executed; otherwise, the workload will be deferred (lines 13-23). Similar to the first part of Algorithm 3, the deferred time td also needs to satisfy the constraints in Equation (8). 3) When the time is after t_r^e, it means the renewable energy is not available any more, therefore, the workloads are executed as soon as possible to comply with the deadlines (line 25).

Algorithm 3. Deferring Algorithm for Batch Workloads

Input: batch workload $b_n(t)$ with start time T_n, execution time E_n, deadline D_n, and the starting time and end time of the available renewable energy t_r^s, t_r^e
Output: deferred time T_n^d

1: **for** $t = T_n$ *in* $b_n(t)$ **do**
2:   **if** $0 < t < t_r^s$ **then**
3:     **if** $D_n < t_r^s$ **then**
4:       execute $b_n(t)$
5:     **else**
6:       defer T_n^d time for execution
7:       $td = t + T_n^d, \forall\, td \leq D_n - E_n, td > t_r^s$
8:       $d(td)' = \sum_m a_m(td)' + \sum_n b_n(td)'$
9:       $R_{td}' > P_{td}'$
10:       $T_n' = td$
11:       update P_{td}'
12:     **end**
13:   **else if** $t_r^s \leq t \leq t_r^e$ **then**
14:     **if** $R_t > P_t$ **then**
15:       execute $b_n(t)$
16:     **else**
17:       defer T_n^d time for execution
18:       $td = t + T_n^d, \forall\, td \leq D_n - E_n$
19:       $d(td)' = \sum_m a_m(td)' + \sum_n b_n(td)'$
20:       $R_{td}' > P_{td}'$
21:       $T_n' = td$
22:       update P_{td}'
23:     **end**
24:   **else**
25:     execute $b_n(t)$
26:   **end**
27: **end**

5.4 Host Scaling

We use a modified host scaling algorithm from [45] by considering renewable energy as shown in Algorithm 4. With profiling data, we configure the threshold of requests that leads to overloads, in which the average response time violates the predefined constraints. The predicted number is calculated based on the number of requests in recent time slots derived from the prediction approach in [31]. The algorithm calculates the difference n' between the number of required servers and actual servers. 1). When more servers are needed, then it adds n' servers into the system (lines 3-4). If the renewable energy is still enough, then it tries to scale more servers into the system to improve the QoS (lines 5-8). 2). If servers are already enough, then remove $|n'|$ servers from system to reduce energy. 3). If n' is 0, then it means no host scaling is required.

Algorithm 4. Hosts Scaling Algorithm

Input: number of hosts n in data center, number of requests when host is overloaded num_{thr}, predicted number of requests $n\hat{u}m(t)$ at time t.
Output: number of active hosts n_a

1: $n_a \leftarrow \lceil n\hat{u}m(t) \div num_{thr} \rceil$
2: $n' \leftarrow n_a - n$
3: **if** $n' > 0$ **then**
4:   Add n' hosts
5:   **while** $P_t \leq R_t$ **do**
6:     Add another host
7:     update P_t
8:   **end**
9: **else if** $n' < 0$ **then**
10:   Remove $|n'|$ hosts $//|n'|$ is the absolute value of n'
11: **else**
12:   no host scaling
13: **end**
14: **return** n_a

5.5 Renewable Energy Prediction

In practice, green data centers with onsite solar installations or wind farms are powered by electricity generated from the renewable source while they are backed up with the Grid (they use inverters that automatically switch energy source based on the availability). In some other potential

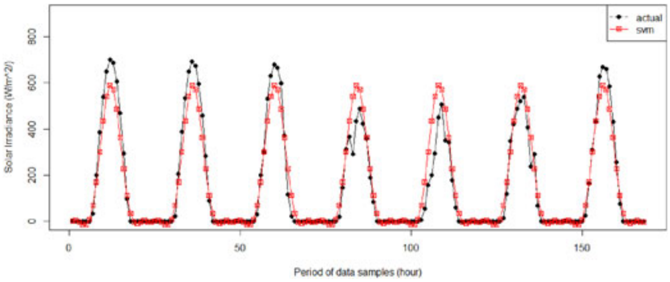


Fig. 2. Denver solar radiation.

scenarios, they are just connected to the Grid and pay for electricity generated from renewable energy sources. In both scenarios, physical servers (electricity consumers) are oblivious to energy sources. Therefore, in this work, we assumed that servers use electricity generated from renewable energy sources without compromising reality and precision. As per our assumption, if renewable energy is insufficient or not available, Grid (Brown) electricity will be used.

We focus on the solar energy as it is one of the most common sources of renewable energy. We use Support Vector Machine (SVM) to predict the solar irradiation or PV power output for the availability of renewable energy, which is a machine learning approach that has been applied to data analysis successfully. In the studies related to solar irradiation prediction [46], [47], SVM has been used to forecast and train the solar radiance model.

Since we do not have the access to the hourly solar irradiance at Melbourne City, in this paper, we use the historical data from NREL Solar Radiation Research Laboratory.[4] The solar panels of the Laboratory are located at Denver, Colorado, US (Latitude 39.742^o North, Longitude 105.18^o West), which has a similar weather to Melbourne instead. We use the hourly-based solar irradiance data from September 1 2018 to November 1 2018. SVM prediction approach has two phases: the training phase and testing phase. 80 percent data is used for the training phase, and 20 percent data is used for the testing phase. Once the process is finished, the test data and prediction results are compared to calculate the error rate. We use the SVM R toolbox for our purpose.

The obtained results are shown in Fig. 2. It shows that the SVM-based approach can achieve the values close to actual ones. In the testing phase, the coefficient of determination (R^2) is 0.763 and correlation coefficient (r) is 0.873. The selected parameters for SVM are regularization parameter $C = 4$ and Kernel bandwidth $\epsilon = 0$. In the experiments section, we applied this trained model to predict solar irradiance. The solar irradiance can be easily calculated with the conversion efficiency of solar panels, e.g., 20 percent. We assume all solar irradiance will be available as power energy. We use this trained model to predict the available renewable energy in advance for the observation period T, e.g., one day.

6 Prototype System Implementation

To realize our system model in Section 3 and evaluate our proposed approach, we configure our testbed to develop a prototype system. Fig. 3 shows the implemented architecture

4. https://midcdmz.nrel.gov/srrl_bms/

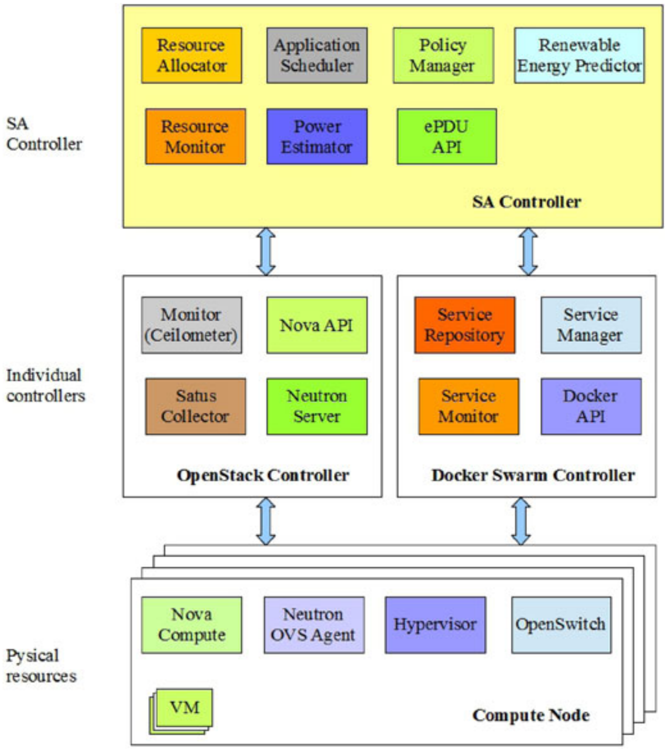


Fig. 3. System architecture and underlying software components of prototype system.

of our prototype system. Cloud resource management platform and microservices management platform have been developed and widely used for years, thus, in this work, we design and implement our prototype based on these mature platforms.

Cloud IaaS resource management platform, OpenStack, is responsible for managing cloud resources, including CPU, memory, and bandwidth. The monitored data of resources is collected by status collector and can be used for resource provisioning and optimization. The microservice management platform, Docker Swarm, is responsible for managing service images, monitoring service resource utilization and managing the service life cycles. Other Docker APIs can also be used to run operations on services. These two platforms are mapped to the Virtualized Platform Management layer and Application Hosting Engine layer respectively in the system model in Fig. 1.

Based on the two management platforms for cloud resources and services, Self-Adaptive (SA) controller is designed to manage and monitor both of them to achieve the multiple level resource scheduling, which is mapped to the Controller component in Fig. 1. When requests are submitted to the system, like interactive workloads or batch workloads, the resource allocator in SA controller manages cloud resource management platform and service management platform simultaneously to accept and process requests by providing the requested amount of resources. Apart from allocating resources to requests, the resource allocator can also optimize resource utilization. For instance, brownout can be triggered to deactivate optional microservices to reduce resource utilization. The service provider can also configure the available resource scheduling policies for the energy efficiency purpose.

To provision and optimize the resources by means of resource allocator, the resource monitor needs to collect the resource usage at different levels, including services utilization, VMs utilization, and hosts utilization. To minimize the overheads of frequently monitored data collection, the collection time intervals should be well configured by the service provider. For instance, the brownout mechanism can be checked every five minutes as the brownout costs are not high, while the VM migration and host scaling operations can be executed with longer time intervals, e.g., one hour.

In the following subsections, we introduce the implementation of our prototype system in details.

6.1 Implementation

To implement our prototype system, we take advantage of the OpenStack cloud resource management platform and Docker Swarm service management platform. The system is implemented with Java, OpenStack, Docker Swarm, Ansible, Eaton Power Distribution Units (ePDU) API. Our prototype system uses these open source tools to provide a self-adaptive approach to optimize, manage and provision resources for different types of workloads.

OpenStack platform is used to manage hosts and VMs. The hosts in OpenStack are called compute nodes and are running with Nova Compute Node component to connect the hypervisor and OpenStack controller. VMs are managed by Nova API to create, migrate and remove VM instances. The Neutron OVS Agent and OpenVSwitch are providing services related to the network.

Docker Swarm Platform manages the service provided by service providers. The images of services are stored in the service repository component, which can fetch the images from remote to local. The services are managed by the service manager via Docker APIs, including creation and deletion. The status of services are monitored by a service which monitors service utilization and liveness.

Our prototype system is based on these services to manage the resources and services to handle the requests. Below, we introduce the details of the components in our prototype.

Resource Allocator. It interacts with OpenStack controller via OpenStack APIs and Docker Swarm Controller via Docker APIs. It manages the physical resources on compute nodes, and these physical resources can be used for creating and deploying VMs on the nodes. Resource Manager knows the amount of resource that is used or remaining on each compute node, like the number of cores, memory, and storage. When creating a VM instance, it can also specify the instance capacity (CPU, memory, operation system and etc.) as well as other information related to VMs, such as location, images of VMs and IP address. The virtual network in a compute node is also managed by Resource Manager that uses the Neutron component, which is deployed on each compute node.

Resource Monitor. It is used to monitor the running status of the whole system from different levels, including hosts, VMs and services. We use OpenStack Ceilometer and Gnocchi components to measure the data at the host and VM level. Ceilometer is responsible for monitoring the utilization of resources of VMs and then sends the collected data to Gnocchi to aggregate the data for all the hosts. We use Docker APIs to collect the resource utilization of services deployed on VMs. Apart from monitoring the resource utilization, we also use ePDU APIs to monitor the power consumption of hosts. With these monitored data, other components, like Power Estimator and Policy Manager can use these data to make decisions, which will be introduced later.

Application Scheduler. We design our main controls in the Application Scheduler component. When requests are submitted by users, the Application Scheduler decides which requests in the batch workloads should be deferred, which microservice should be temporarily deactivated by brownout mechanism, which VM should be migrated to another host and which host should be switched to the low power mode. With the retrieved data from the Resource Monitor component, these decisions are made with the policies in the Policy Manager. After the decisions are made, Resource Provisioner exploits Resource Manager to allocate the resources to VMs, services, and requests.

Power Consumption Estimator. To achieve our objective of managing energy and support our scheduling policies, we have a power consumption estimator to predict the power consumption at a specific time period. For example, for the batch workloads, we proposed a deferring algorithm, thus we need to estimate the power consumption at the deferred time period to calibrate our algorithm. We use the workloads model shown in Equation (7) to estimate the workloads and then convert it to the total energy consumption based on the model in [25].

Policy Manager. It contains the implemented scheduling policies in our prototype, e.g., Algorithms 1, 2, 3, and 4. The Policy Manager component uses the retrieved data from Resource Monitor, and makes decisions based on system status. For example, a VM is migrated from an underutilized host to other hosts, thus the idle host can be switched to the low power mode to save power consumption; when the renewable energy is not sufficient and the system is overloaded, to ensure the QoS of service, brownout can be triggered to relieve the overloaded situation. The customized workloads processing policy, VM migration policy and host scaling policy can also be implemented for the policy manager.

ePDU API. Eaton Power Distribution Units[5] is an effective power management and monitoring device. It has outlets that allow electric devices to be connected to it. It also provides the features to read the power consumption of hosts as well as turn on/off the outlets remotely. We implemented Python scripts based on ePDU APIs to read the power data at per second rate to support part of the functions in Resource Monitor. Our scripts can also operate the hosts remotely by turning on/off the power supply to hosts to support the decision in Policy Manager. For example, a host needs to be scaled out if the whole system is underutilized; or hosts should be scaled in to support more requests.

Renewable Energy Predictor. For supporting our renewable energy experiments, we implement a renewable energy predictor that predicts the renewable energy at Denver city based on the historical data. As introduced in Section 5.5, our prediction models show that it can achieve a high accuracy. The data based on this component can also be incorporated into the scheduling policy.

5. https://powerquality.eaton.com/ePDUG3/

TABLE 2
Machines Specification

Machine	CPU	Cores	Memory	Storage	Idle Power	Full Power
3 × IBM X3500 M4	2 GHz	12	64 GB	2.9 TB	153 Watts	230 Watts
4 × IBM X3200 M3	2.8 GHz	4	16 GB	199 GB	60 Watts	150 Watts
2 × Dell OptiPlex 990	3.4 GHz	4	8 GB	399 GB	26 Watts	106 Watts

7 PERFORMANCE EVALUATION

To evaluate the performance of our proposed approach, we conduct experiments in our implemented prototype system. We first present the environment settings in Section 7.1, and then introduce the workload and application settings in Sections 7.2 and 7.3. The results are demonstrated and discussed in Section 7.4.

7.1 Environmental Settings

In our experiments, the upper utilization threshold TU_{up} and lower utilization threshold TU_{low} are configured as 80 and 20 percent respectively, as these values have been evaluated in our previous work [17], [31] that they can achieve good trade-offs between energy consumption and QoS than other values. For example, configuring the upper utilization threshold to be lower than 80 percent can trigger brownout too frequently, while the upper utilization threshold with 90 percent or higher can hardly trigger brownout. We also configure the scheduling time interval as 5 minutes and the whole scheduling period as one day.

Hardware. We utilize a micro data center of Melbourne CLOUDS lab as testbed. Our data center consists of 9 heterogeneous servers. Table 2 shows the capacity specification of the servers and their energy consumption information. To monitor the power consumption of individual machines, we use two ePDUs and all the servers are connected to them. Apart from the power monitor, the ePDUs also enable us to switch on/off power outlets connected with individual server remotely through the network. The total maximum power of the IT equipment in our system is 1.27 kWh for 8 hosts (one IBM X3500 M4 machine is regarded as the OpenStack control node and is not considered).

We assume our system is equipped with 1.63 kW PV panel,[6] which has 30 percent more power than the maximum power of hosts, as the cooling part normally consumes about 20 to 30 percent of server energy if the target temperature is 25 degree [14]. This cooling power consumption percentage is validated in the prototype system in [16]. We consider to control the data center temperature as 25 degree, according to Equation (4), $T_{sup} = 25$, then we get $CoP(T_{sup}) = 4.728$. In the following experiments, we use this value to compute the power from the cooling equipment, e.g., if the hosts consume 10 kWh, then the cooling part is 2.11 kWh.

Software. The operating systems of the servers are CentOS Linux Distribution. We use OpenStack [39] to support our cloud platform and manage the VMs. One of our most powerful machines is selected as our controller node, and other nodes are acting in the same role. In VM instances, we deploy Docker [48] containers to provide services in the form of microservices and use Docker Swarm to manage the containers cluster. Some other required software, like Java, Ansible are also installed in the VMs.

6. The total power of PV panels can be increased by adding more panels.

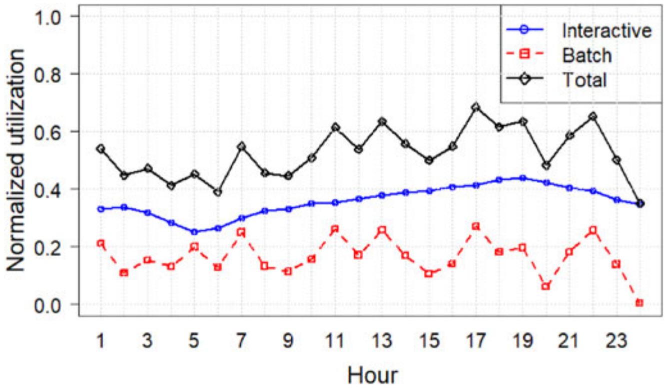


Fig. 4. Workloads distribution.

7.2 Workloads

To make the experiments as realistic as possible, we use real traces from Wikipedia and Facebook. For the interactive workloads, we use the real trace from Wikipedia requests on 2007 October 17 to replay the workload of Wikipedia users. The trace includes data on requests time stamp and their accessed pages. We filter the requests based on per second rate and generate the requests rate. The original request rate is around 1,500- 3,000 per second. We use 10 percent of the original requests and these requests can consume up to 43 percent cluster utilization.

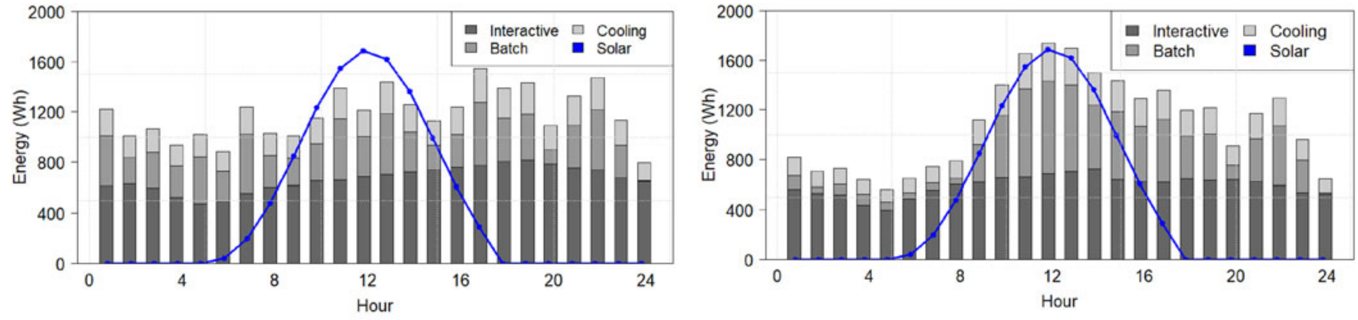
For the batch workloads, we use the traces collected in October 2009 by Facebook for applications that are executed under Hadoop environment.[7] Referring to [16], we configure the map phase of each job takes 25-13000 seconds, and the reduce phase takes 15-2600 seconds. The deadline for processing jobs is generated based on uniform distribution with $\mu = 6$ hours and $\sigma = 1$ hour in $N(\mu, \sigma^2)$. We also assume the workloads consume the maximum of cluster utilization as 27 percent as same as in [16].

Fig. 4 shows the one-day normalized resource utilization trace of the aforementioned workloads. We can clearly see the variance of utilization demand of both interactive and batch workloads, thus the workloads can be managed to fit into the availability of green energy. For instance, at hour 7, if the green energy is not sufficient to supply the all the workloads, then some batch workloads can be deferred to a later time when more green energy is available.

7.3 Application

We use the Weave Shop[8] web application that is implemented with containers as the application to process the interactive workloads derived from Wikipedia traces. The Weave Shop is a web-based shopping system for selling socks online and has multiple microservices, including user microservice to handle user login, user database microservice for user information storage, payment microservice to

7. https://github.com/SWIMProjectUCB/SWIM/wiki/Workloads-repository
8. See https://github.com/microservices-demo/microservices-demo for more details.



(a) (b)

Fig. 5. Results of (a) baseline HS (b) proposed approach GSA.

process transactions, front-end microservice to show the user interface, catalog microservice for managing item for sale etc. As these microservices are implemented independently, they can be deployed and controlled without impacting other microservices. The application is deployed by a configuration file, and 30 percent of the microservices are configured to be optional, e.g., the recommendation engine. The generated workload is communicated to the mandatory microservices (e.g., the font-end microservices) to measure the response time. The microservices are deployed on the cluster with multiple VMs. The head node of the cluster is deployed with a gateway microservice in Weave Shop that is responsible for distributing the interactive workloads to different microservices deployed on multiple VMs. The application and VMs are deployed on the hosts with dynamically adjusted.

7.4 Results

To evaluate the benefits of our proposed approach for renewable energy usage, we perform the comparison of our proposed Green-aware and Self-adaptive approach (GSA) and a state-of-the-art baseline algorithm (HS), which applies VM consolidation based on Modified Best Fit Decreasing algorithm [17] that consolidates VMs to the hosts that produce the minimum energy incrementation, and host scaling [45] that dynamically adds/removes hosts in system based on profiling and workloads prediction.

Fig. 5a shows the baseline energy consumption of interactive workloads, batch workloads and cooling during the observed time period (one day). The blue line shows the actual renewable power production. We consider the day in autumn time for Denver city. In this season, the day-length is about 12 hours, which is shorter than the summer time but longer than the winter time. In the investigated day, the system is consuming brown energy at *night time* from hour 0:00 to hour 5:00 and hour 18:00 to hour 24:00. The solar energy is available at *daytime* during hours 6:00 to 17:00. Even with taking advantage of VM consolidation and host scaling, the solar energy consumption of the system is not fully utilized. For example, at hour 11:00, the total energy consumption of the system is about 1400 Wh, while the available solar energy is more than 1500 Wh.

Fig. 5b demonstrates the energy consumption of GSA approach by using Algorithms 1, 2, 3, and 4. The blue line still shows the actual renewable power production, but the decision making is happening based on our SVM prediction model. We can observe that the power consumption of the batch workloads during 0:00 to 8:00 has been reduced, which results from the deferring operations: batch workloads are deferred to the time when solar energy is available, e.g., hour 6:00. Some batch workloads are still executed during hours from 0:00 to 8:00 due to the deadline constraints, which cannot be deferred to the time when renewable energy is available. Thus, we can find that the brown energy usage during 0:00 to 8:00 has been reduced compared in Fig. 5a. For example, at hour 1:00, the total power is reduced from 1221 to 815 Watts.

During the time when solar energy is available, GSA approach has improved the usage of renewable energy, in which the energy consumption follows the line the of predicted renewable energy. For instance, at hour 11:00, the usage of solar energy is increased from 1387 Wh to 1544 Wh compared with Fig. 5a.

We also note that the power consumption during the time when brown energy is the only source of power supply, the energy is also reduced, which exploits the brownout mechanism to reduce the energy consumption. For instance, the power at hour 18:00 is decreased from 1391 Watts to 1195 Watts.

Combining the results in Figs. 5a and 5b, we conclude that the GSA approach can improve the usage of renewable energy and reduce the usage of brown energy. To be noted, we choose the time in autumn for our experiments, as the renewable energy production in autumn is close to the time in spring, and these two seasons can represent renewable energy production about half a year. As for summer time, the day-length is extended, e.g., from hour 5:00 to hour 20:00, which means the workloads can have more possibilities to be deferred rather than being executed as soon as possible, and thus the utilization of renewable energy can be further improved. However, in the winter time, the day-length is reduced, e.g., only from hour 7:00 to 16:00. In this season, the utilization of renewable energy is not as good as in the autumn and summer seasons.

The average response time and cumulative distribution function (CDF) of response time for interactive workloads in our proposed approach and baseline are illustrated in Fig. 6. The average response time of GSA is 403.4 ms, which is less than 80 percent of the HS (513.1 ms). In GSA approach, the results also show that 95 percent requests are responded in 900 ms, and 99 percent requests are responded within 1000 ms second. While in the HS approach, only 91

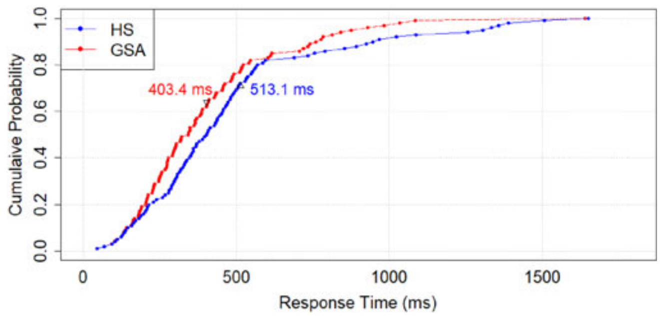


Fig. 6. Comparison of response time for interactive workloads.

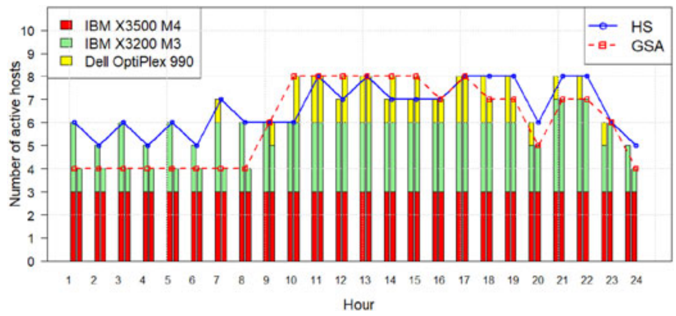


Fig. 7. Comparison of number of active hosts.

percent requests are responded within 1000 ms. It shows that GSA approach reduces brown energy usage while ensuring the QoS. The reason is that the brownout approach can relieve the overloaded situation, thus ensuring the response time.

To illustrate the reason for power reduction by GSA approach, Fig. 7 compares the active hosts during the observed time period between the HS and GSA approaches, as switching the idle hosts into the low power mode is the most effective way to save power. The results demonstrate that GSA approach uses fewer hosts during the time period when renewable energy is not sufficient, e.g., hours from 0:00 to 8:00 and 18:00 to 24:00, while when the renewable energy is available, more hosts are scaled in to utilize more renewable energy, such as the time from 10:00 to 15:00. In this way, the power of all the active hosts is reduced and the usage of renewable energy is improved. Fig. 7 also shows the type of active hosts in HS and GSA approaches. Based on the results, we can notice that the deactivation and activation are mainly operated on machines type of IBM X3200 M3 and Dell OptiPLex 990, which have less capacity than IBM X3500 M4 as shown in Table 2. The main reason is that these two types of hosts can host fewer VMs than IBM X3500 M4, and the VMs can be more easily consolidated to other machines. Therefore, after consolidation, the idle hosts can be switched to the low power mode.

Fig. 8a demonstrates the comparison of brown and renewable energy usage. During the night time (0:00 to 5:00 and 18:00 to 24:00), both approaches only use brown energy. Benefiting from proposed algorithms, our approach reduces the brown energy usage by 28 percent from 13.9 kWh to 10.8 kWh. During the daytime (6:00 to 17:00), both renewable energy and brown energy are used in two approaches. GSA approach consumes 5 percent more total energy in the daytime, while it uses 10 percent more renewable energy than the baseline from 9.9 KWh to 10.9 KWh. In total power usage comparison, the brown energy usage is reduced 21 percent, and the renewable energy usage is improved 10 percent.

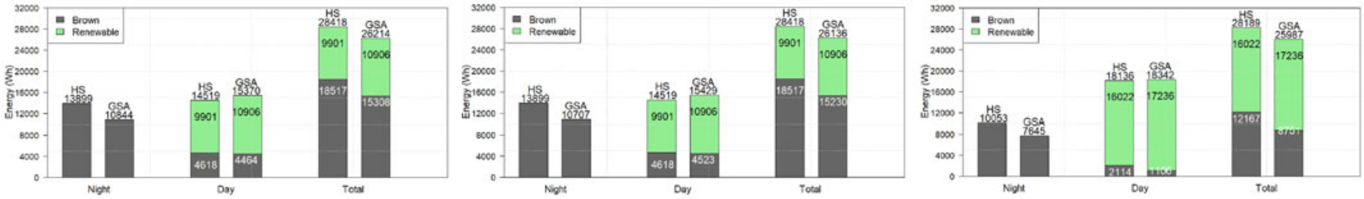
To further investigate the impacts on different configurations, we conduct another two experiments by changing the deadline of batch workloads and the availability of renewable energy. Due to the page limitation, we only demonstrate the results of different types of power usage like in Fig. 8a.

Longer Deadlines of Batch Workloads: we generate the deadline for processing jobs as uniform distribution with $\mu = 7$ hours and $\sigma = 2$ hours in $N(\mu, \sigma^2)$, which has a longer deadline than the previous configurations. The other configurations are the same as the previous experiments. In Fig. 8b, compared with HS, we can notice that the brown energy usage in the night time is reduced from about 13.9 kWh to 10.7 kWh by GSA, renewable energy usage is increased from 9.9 kWh to 10.9 kWh, and total energy usage is reduced about 8 percent by GSA. Compared with the results in Fig. 8a where the batch workloads have shorter deadlines, the brown energy consumption has been reduced, e.g., from 10.8 kWh to 10.7 kWh in the night time, which shows that longer deadline is helpful to reduce brown energy usage. The reason is that more batch workloads are deferred to the daytime when the renewable energy is available. Therefore, the brown energy usage in the night time is reduced. However, the brown energy usage in the daytime is increased a bit while the total brown energy usage is reduced.

Longer Daytime in Summer With More Varied Solar Power: since the solar power dataset of Denver city has sufficient data, thus we still use the dataset, but we change the season from autumn to summer, which has longer daytime that starts from hour 5:00 and ends at hour 19:00 and can represent solar power with variability. The other settings are configured the same as settings in Fig. 8a. Fig. 8c shows the power usage comparison with longer daytime. It can be observed that GSA can reduce the total brown energy usage from 12.2 kWh to 8.8 kWh and renewable energy usage can be improved from 16.0 kWh to 17.2 kWh. Compared with the results in Figs. 8a and 8b, the renewable energy percentage of total energy usage has been significantly increased, as the daytime is extended and the amount renewable energy is more sufficient in the summer time compared with the autumn time.

To evaluate the brownout impacts on microservices, we use the average deactivation percentage (the average number of deactivated microservices divided by the total number of microservices during the observation time). The higher the average deactivation percentage, the more microservices are deactivated, and vice verse. The average deactivation percentage for cases in Figs. 8a and 8b are 11.2 and 10.8 percent, respectively, and in the case of Fig. 8c, the value is 10.1 percent. Based on the results, we can observe that the longer deadline of batch workloads and the longer daytime result in a lower number of deactivated microservices. The reason is that a higher renewable energy availability can support more active microservices.

In summary, experiments show that GSA approach can improve the renewable energy usage for both interactive



(a) Autumn time in Denver city (b) Longer deadline of batch workloads (c) Longer daytime in summer

Fig. 8. Power usage comparison of different types of energy sources.

and batch workloads by applying brownout mechanism and deferring the execution of batch workloads. Our proposed approach can switch more machines into low power mode when renewable energy is not sufficient while the QoS of workloads is also ensured.

8 Conclusion and Future Work

Our self-adaptive approach for managing applications and harnessing renewable energy brings up many opportunities to optimize the energy efficiency problem in cloud computing environment. In this paper, we proposed a multiple layers perspective model for interactive and batch workloads by considering renewable energy. We also introduced a self-adaptive and renewable energy-aware approach deriving from the perspective model. The proposed approach improves the usage of renewable energy and reduces the usage of brown energy while ensuring the QoS requirement of workloads. We apply a solar radiation prediction method to predict solar power at Denver City and integrate it into our proposed approach. We utilize brownout mechanism to dynamically deactivate/activate optional components in the system for interactive workloads and use a deferring algorithm to defer the execution of batch workloads to the time when renewable energy is available. VM consolidation and host scaling are also applied to reduce the number of active hosts.

We developed a prototype system to evaluate the performance of our proposed approach. In the prototype system, the physical resources are managed by OpenStack and the services are managed by Docker Swarm. We take advantage of the APIs from these platforms to monitor, manage, and provision the resources to services. The effectiveness of our proposed approach is showed through the experimental evaluations with a microservices-based web system and the workloads from real traces. The results show that our proposed approach is able to improve the usage of renewable energy while satisfying the constraints of workloads.

As future work, we would like to include the battery model in [16], which can store renewable energy and improve energy usage. We also plan to extend our prototype system for multiple clouds in the different time zones to support workload shifts in data centers and minimize the carbon footprint in a global view.

Fog and Edge computing extend the cloud services to the edge of the network, which can improve the user experience and system performance by reducing latency. However, the IoT and edge devices have power constraints, for example, they are powered by battery or they need to harness renewable energy. Therefore, the energy should be used in an efficient manner. The brownout approach can support to optimize the energy usage for these devices by temporarily deactivating some optional application components. As another future work, we would like to apply the brownout approach to mobile edge computing for managing the energy usage of IoT or edge devices.

Acknowledgments

This work was supported by the Key-Area Research and Development Program of Guangdong Province (NO. 2020B010164003), Science and Technology Development Fund of Macao S.A.R (FDCT) under number 0015/2019/AKP, Shenzhen Discipline Construction Project for Urban Computing and Data Intelligence, SIAT Innovation Program for Excellent Young Researchers and ARC Discovery Project.

References

[1] C. Kilcioglu, J. M. Rao, A. Kannan, and R. P. McAfee, "Usage patterns and the economics of the public cloud," in *Proc. 26th Int. Conf. World Wide Web*, 2017, pp. 83–91. [Online]. Available: https://doi.org/10.1145/3038912.3052707
[2] Y. Chen *et al.*, "Outage prediction and diagnosis for cloud service systems," in *Proc. World Wide Web Conf.*, 2019, pp. 2659–2665. [Online]. Available: http://doi.acm.org/10.1145/3308558.3313501
[3] W. Jiang, Z. Jia, S. Feng, F. Liu, and H. Jin, "Fine-grained warm water cooling for improving datacenter economy," in *Proc. 46th Int. Symp. Comput. Architecture*, 2019, pp. 474–486. [Online]. Available: http://doi.acm.org/10.1145/3307650.3322236
[4] M. Avgerinou, P. Bertoldi, and L. Castellazzi, "Trends in data centre energy consumption under the european code of conduct for data centre energy efficiency," *Energies*, vol. 10, no. 10, 2017, Art. no. 1470.
[5] B. Whitehead, D. Andrews, A. Shah, and G. Maidment, "Assessing the environmental impact of data centres part 1: Background, energy use and metrics," *Building Environ.*, vol. 82, pp. 151–159, 2014.
[6] P. Delforge, "Data center efficiency assessment - scaling up energy efficiency across the data center industry: Evaluating key drivers and barriers," 2014. [Online]. Available: https://www.infrastructureusa.org/scaling-up-energy-efficiency-across-t he-data-center-industry/
[7] M. Pedram, "Energy-efficient datacenters," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 31, no. 10, pp. 1465–1484, Oct. 2012.
[8] A. Beloglazov and R. Buyya, "OpenStack Neat: A framework for dynamic and energy-efficient consolidation of virtual machines in OpenStack clouds," *Concurrency Comput., Pract. Experience*, vol. 27, no. 5, pp. 1310–1333, 2015.
[9] A. Belog1azov and R. Buyya, "Managing overloaded hosts for dynamic consolidation of virtual machines in cloud data centers under quality of service constraints," *IEEE Trans. Parallel Distrib. Syst.*, vol. 24, no. 7, pp. 1366–1379, Jul. 2013.
[10] B. Clement, "Rightscale 2017 state of the cloud report uncovers cloud adoption trends," 2017. [Online]. Available: https://www.rightscale.com/press-releases/
[11] S. Newman, *Building Microservices*. Sebastopol, CA, USA: O'Reilly Media, Inc., 2015.

[12] Z. Ren *et al.*, “Migrating web applications from monolithic structure to microservices architecture,” in *Proc. 10th Asia-Pacific Symp. Internetware*, 2018, pp. 7:1–7:10. [Online]. Available: http://doi.acm.org/10.1145/3275219.3275230

[13] M. Xu and R. Buyya, “Brownout approach for adaptive management of resources and applications in cloud computing systems: A taxonomy and future directions,” *ACM Comput. Surv.*, vol. 52, no. 1, pp. 8:1–8:27, 2019.

[14] Z. Liu *et al.*, “Renewable and cooling aware workload management for sustainable data centers,” *ACM SIGMETRICS Perform. Eval. Rev.*, vol. 40, no. 1, pp. 175–186, 2012.

[15] Z. Han, H. Tan, G. Chen, R. Wang, Y. Chen, and F. C. M. Lau, “Dynamic virtual machine management via approximate Markov decision process,” in *Proc. 35th Annu. IEEE Int. Conf. Comput. Commun.*, 2016, pp. 1–9.

[16] Í. Goiri, W. Katsak, K. Le, T. D. Nguyen, and R. Bianchini, “Parasol and GreenSwitch: Managing datacenters powered by renewable energy,” *ACM SIGARCH Comput. Architecture News*, vol. 41, no. 1, pp. 51–64, 2013.

[17] A. Beloglazov, J. Abawajy, and R. Buyya, “Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing,” *Future Gener. Comput. Syst.*, vol. 28, no. 5, pp. 755–768, 2012.

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

[19] H. Liu *et al.*, “Thermal-aware and DVFS-enabled big data task scheduling for data centers,” *IEEE Trans. Big Data*, vol. 4, no. 2, pp. 177–190, Jun. 2018.

[20] F. Teng, L. Yu, T. Li, D. Deng, and F. Magoulès, “Energy efficiency of VM consolidation in IaaS clouds,” *The J. Supercomputing*, vol. 73, pp. 782–809, 2017.

[21] T. H. Nguyen, M. D. Francesco, and A. Yla-Jaaski , “Virtual machine consolidation with multiple usage prediction for energy-efficient cloud data centers,” *IEEE Trans. Services Comput.*, vol. 13, no. 1, pp. 186–199, Jan./Feb. 2020.

[22] M. Xu, A. V. Dastjerdi, and R. Buyya, “Energy efficient scheduling of cloud application components with brownout,” *IEEE Trans. Sustain. Comput.*, vol. 1, no. 2, pp. 40–53, Jul.–Dec. 2016.

[23] M. S. Hasan, F. Alvares, T. Ledoux, and J.-L. Pazat, “Investigating energy consumption and performance trade-off for interactive cloud application,” *IEEE Trans. Sustain. Comput.*, vol. 2, no. 2, pp. 113–126, Apr.–Jun. 2017.

[24] X. Li, X. Jiang, P. Garraghan, and Z. Wu, “Holistic energy and failure aware workload scheduling in cloud datacenters,” *Future Gener. Comput. Syst.*, vol. 78, pp. 887–900, 2018.

[25] Z. Liu *et al.*, “Renewable and cooling aware workload management for sustainable data centers,” *ACM SIGMETRICS Perform. Eval. Rev.*, vol. 40, no. 1, pp. 175–186, 2012.

[26] M. Xu, A. N. Toosi, B. Bahrani, R. Razzaghi, and M. Singh, “Optimized renewable energy use in green cloud data centers,” in *Proc. Int. Conf. Service-Oriented Comput.*, 2019, pp. 314–330.

[27] D. Cheng, X. Zhou, P. Lama, M. Ji, and C. Jiang, “Energy efficiency aware task assignment with DVFS in heterogeneous hadoop clusters,” *IEEE Trans. Parallel Distrib. Syst.*, vol. 29, no. 1, pp. 70–82, Jan. 2018.

[28] L. Tomás, C. Klein, J. Tordsson, and F. Hernández-Rodríguez, “The straw that broke the camel’s back: Safe cloud overbooking with application brownout,” in *Proc. IEEE Int. Conf. Cloud Auton. Comput.*, 2014, pp. 151–160.

[29] M. Shahrad, C. Klein, L. Zheng, M. Chiang, E. Elmroth, and D. Wentzlaff, “Incentivizing self-capping to increase cloud utilization,” in *Proc. Symp. Cloud Comput.*, 2017, pp. 52–65.

[30] M. Xu and R. Buyya, “Energy efficient scheduling of application components via brownout and approximate Markov decision process,” in *Proc. 15th Int. Conf. Service-Oriented Comput.*, 2017, pp. 206–220.

[31] M. Xu, A. N. Toosi, and R. Buyya, “iBrownout: An integrated approach for managing energy and brownout in container-based clouds,” *IEEE Trans. Sustain. Comput.*, vol. 4, no. 1, pp. 53–66, Jan.–Mar. 2019.

[32] A. M. Al-Qawasmeh , S. Pasricha, A. A. Maciejewski, and H. J. Siegel, “Power and thermal-aware workload allocation in heterogeneous data centers,” *IEEE Trans. Comput.*, vol. 64, no. 2, pp. 477–491, Feb. 2015.

[33] 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 Trans. Parallel Distrib. Syst.*, vol. 19, no. 11, pp. 1458–1472, Nov. 2008.

[34] Docker documentation | docker documentation. 2017. [Online]. Available: https://docs.docker.com/

[35] Swarm mode overview | docker documentation. 2018. [Online]. Available: https://docs.docker.com/engine/swarm/

[36] Production-grade container orchestration - kubernetes. 2018. [Online]. Available: https://kubernetes.io/

[37] Apache mesos. 2018. [Online]. Available: http://mesos.apache.org/

[38] Vmware - official site. 2018. [Online]. Available: https://www.vmware.com/

[39] Open source software for creating private and public clouds. 2018. [Online]. Available: https://www.openstack.org/

[40] V. Nallur and R. Bahsoon, “A decentralized self-adaptation mechanism for service-based applications in the cloud,” *IEEE Trans. Softw. Eng.*, vol. 39, no. 5, pp. 591–612, May 2013.

[41] T. Chen and R. Bahsoon, “Self-adaptive and online QoS modeling for cloud-based software services,” *IEEE Trans. Softw. Eng.*, vol. 43, no. 5, pp. 453–475, May 2017.

[42] K. Zheng, X. Wang, L. Li, and X. Wang, “Joint power optimization of data center network and servers with correlation analysis,” in *Proc. IEEE Conf. Comput. Commun.*, 2014, pp. 2598–2606.

[43] T. Kaur and I. Chana, “Energy efficiency techniques in cloud computing: A survey and taxonomy,” *ACM Comput. Surv.*, vol. 48, no. 2, pp. 1–46, 2015.

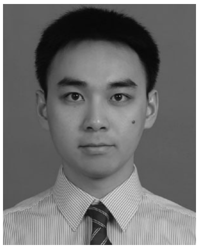
[44] J. D. Moore, J. S. Chase, P. Ranganathan, and R. K. Sharma, “Making scheduling” cool”: Temperature-aware workload placement in data centers,” in *Proc. USENIX Annu. Techn. Conf.*, 2005, pp. 61–75.

[45] A. N. Toosi, C. Qu, M. D. de Assunção, and R. Buyya, “Renewable-aware geographical load balancing of web applications for sustainable data centers,” *J. Netw. Comput. Appl.*, vol. 83, pp. 155–168, 2017.

[46] K. Y. Bae, H. S. Jang, and D. K. Sung, “Hourly solar irradiance prediction based on support vector machine and its error analysis,” *IEEE Trans. Power Syst.*, vol. 32, no. 2, pp. 935–945, Mar. 2017.

[47] S. Belaid and A. Mellit, “Prediction of daily and mean monthly global solar radiation using support vector machine in an arid climate,” *Energy Convers. Manage.*, vol. 118, pp. 105–118, 2016.

[48] Docker, “Docker documentation | docker documentation. 2017. [Online]. Available: https://docs.docker.com/



Minxian Xu (Member, IEEE) received the BSc and MSc degrees in software engineering from the University of Electronic Science and Technology of China, China, in 2012 and 2015, respectively, and the PhD degree from the University of Melbourne, Australia, in 2019. He is currently an assistant professor at the Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, China. His research interests include resource scheduling and optimization in cloud computing. He has coauthored more than 20 peer-reviewed papers published in prominent international journals and conferences, such as the *ACM Computing Surveys*, the *IEEE Transactions on Sustainable Computing*, the *IEEE Transactions on Automation Science and Engineering*, the *Journal of Parallel and Distributed Computing*, the *Concurrency and Computation: Practice and Experience*, International Conference on Service-Oriented Computing. His PhD Thesis was awarded the 2019 IEEE TCSC Outstanding PhD Dissertation Award. For more information, please visit minxianxu.info.

Adel Nadjaran Toosi (Member, IEEE) received the PhD degree in computer science and software engineering from the University of Melbourne, Australia, in 2015. He has joined faculty of information technology at Monash University, Australia as a lecturer, in May 2018. Before he joined Monash University, Australia, he worked as a research fellow in the Cloud Computing and Distributed Systems (CLOUDS) Laboratory in the School of Computing and Information Systems (CIS) at the University of Melbourne, Australia for more than three years. His thesis was one of the two theses nominated for the Chancellor's Prize for Excellence in the PhD Thesis and John Melvin Memorial Scholarship for the Best PhD Thesis in Engineering. He has made significant contributions to the areas of resource management and software systems for cloud computing. His research interests include distributed systems, cloud computing, software-defined networking (SDN), green computing, and soft computing. He is currently working on resource management for Software-Defined Networking (SDN)-enabled cloud computing environments. For more details, please visit http://adelnadjarantoosi.info.

Rajkumar Buyya (Fellow, IEEE) is a Redmond Barry distinguished professor and director of the Cloud Computing and Distributed Systems (CLOUDS) Laboratory at the University of Melbourne, Australia. He is also serving as the founding CEO of Manjrasoft, a spin-off company of the University, commercializing its innovations in Cloud Computing. He served as a future fellow of the Australian Research Council during 2012-2016. He has authored more than 625 publications and seven text books including "Mastering Cloud Computing" published by McGraw Hill, China Machine Press, and Morgan Kaufmann for Indian, Chinese and international markets respectively. He is one of the highly cited authors in computer science and software engineering worldwide (h-index=136, g-index=300, and more than 98,800 citations). He is recognized as a Web of Science Highly Cited Researcher for four consecutive years since 2016, and Scopus researcher of the Year 2017 with Excellence in Innovative Research Award by Elsevier for his outstanding contributions to Cloud computing. He served as the founding editor-in-chief of the *IEEE Transactions on Cloud Computing*. He is currently serving as co-editor-in-chief of the *Journal of Software: Practice and Experience*, which was established 50 years ago. For more information, please visit www.buyya.com.