Received March 15, 2022, accepted April 4, 2022, date of publication April 7, 2022, date of current version April 14, 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3165566

Power- and QoS-Aware Job Assignment With Dynamic Speed Scaling for Cloud Data Center Computing

YONGKYU CHO[1], (Member, IEEE), AND YOUNG MYOUNG KO[2], (Member, IEEE)
[1]Department of Industrial and Management Engineering, Kangnam University, Yongin, Gyeonggi 16979, South Korea
[2]Department of Industrial and Management Engineering, Pohang University of Science and Technology, Pohang, Gyeongbuk 37673, South Korea

Corresponding author: Young Myoung Ko (youngko@postech.ac.kr)

This work was supported in part by the Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education under Grant NRF-2016R1D1A1B04933453; and in part by the National Research Foundation of Korea (NRF) grant funded by the Korea Government [Ministry of Science and ICT (MSIT)] under Grant 2021R1G1A1094924, Grant 2021R1A4A1031019, and Grant 2021R1A2C1094699.

ABSTRACT As the usage of mission-critical mobile applications increases in Industry 4.0, such as smart manufacturing and self-driving cars, the cloud computing paradigm and its supporting data centers have become more crucial. However, a common practice in the cloud data center computing industry tends to supply a surfeit of computing resources mainly for a robust quality-of-service (QoS). In this paper, we propose a simple real-time algorithm which combines a power-aware job assignment policy for a centralized job dispatcher and a power- and QoS-aware dynamic speed scaling policy for each physical machine (PM). The job assignment policy is called ''Join the Least Power Consuming (LPC) Server'' that routes an incoming cloud job to a server spending minimum power upon request. The server-side adaptive speed scaling policy expedites energy efficiency and satisfies response time-associated QoS condition. We call this policy ''Minimizing Earliness (ME)'' since it manages the server speed towards finishing jobs at their deadlines as precisely as possible, reducing the earliness of job completion. The design principle of LPC-ME combination supports both energy efficiency and service quality required in cloud data centers. Numerical experiments compare the proposed algorithm's power consumption and response time with those of existing popular policies and demonstrate better energy efficiency with negligible degradation of service quality.

INDEX TERMS Cloud computing, data centers, energy efficiency, queueing analysis, resource management.

I. INTRODUCTION

Energy efficiency in data centers has become an essential issue as most computing today happens within cloud data centers, which consume a tremendous amount of electric energy [1]. Describing the environmental impact, data centers consume over 3% of the global electricity and produce over 2% of the total global greenhouse gas emissions [2]. While the harmful effect due to large-scale data centers has been pointed out for the last decade, typical resource utilization is still known to be 30% or less [3]; there is still huge room for improvement to reduce energy consumption.

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

For those who manage large-scale data centers, proper job assignment for load balancing is a fundamental problem because it is directly related to service quality [4]–[6]. There are well-known job assignment policies based on different types of *congestion* information, such as round-robin (no information), shortest queue length (number of jobs), and least workload (load estimation). They are simple, easy to implement, and effective in terms of load balancing [1], [6]–[8].

Inside data centers, servers are considered as main energy-consuming equipment [9]–[11]. Servers are reported to consume about 80% of the total energy while networking, and storage devices possess the remaining 20% [10]. Typically, CPU has been the most significant contributor to the power consumption in servers, as depicted in Fig. 1. As such,

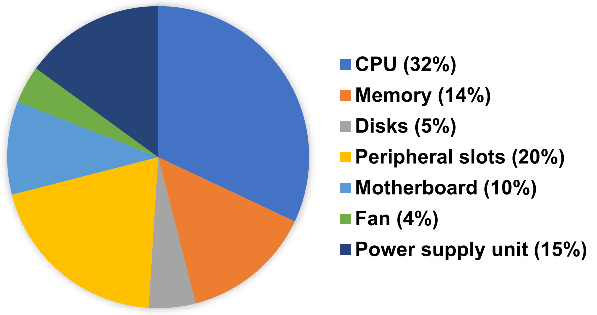


FIGURE 1. Breakdown of power consumption in a server [10].

researchers in the various fields have been developing novel technologies for pursuing more efficiency in CPU usage, for example, dynamic voltage/frequency scaling (DVFS), wake-on-LAN (WoL), and virtualization.

Alongside various energy-saving technologies, recent studies focused on achieving two contradicting objectives: energy efficiency and quality-of-service (QoS) (see [5], [9], [12]–[16] and references therein). Most of them require solving complicated optimization problems, and the solution may not be easy to interpret and implement. To this end, this paper focuses more on developing a simple algorithm that explicitly considers computer servers' power usage behavior and time-sharing concept with a response time-related QoS condition which is one of the most important performance metrics regarding service level agreements (SLAs).

Our work contributes to the green cloud computing literature by proposing a simple but powerful algorithm easily applicable to data centers. The algorithm consists of a new job assignment policy (*least-power-consuming*, LPC) and a new dynamic speed scaling policy (*minimizing earliness*, ME) to save energy and satisfy service quality simultaneously; the joint algorithm is called LPC-ME. To the best of our knowledge, this is the first work that proposes and combines two simple power- and QoS-aware policies each of which is designed for job dispatcher and physical server, respectively. The main contributions of this paper are summarized as follows:

- We propose a power-aware job assignment policy to achieve high energy efficiency. The policy utilizes the real-time power usage information of each server.
- We suggest a power- and QoS-aware speed scaling policy that takes into account the QoS condition on job response time and physical machines (PMs) with time-sharing processors. The policy utilizes the real-time information on jobs' deadlines, workloads, and current speed.
- Combining the job assignment policy with the speed scaling policy, we design a simple power- and QoS-aware real-time algorithm called LPC-ME. The energy saving effect of LPC-ME combination is supported by a short proof, numerical examples, and performance evaluation.

We note that we purposely maintain a macroscopic view of the system to focus on energy conservation and service quality. In this regard, we avoid discussing different areas of cloud systems details such as security and network topology that are out of the scope of this study.

The remainder of this paper is organized as follows. We provide a literature review of the existing green cloud computing studies in Section II. Section III describes the target system. Section IV states a mathematical model and formulation. In Section V, we propose our algorithm and explain the underlying design principle. Section VI introduces some numerical examples that account for the working mechanism and energy efficiency of the proposed method. Section VII provides the performance evaluation results and interpretation. Finally, Section VIII concludes and suggests possible extensions of the study.

II. RELATED WORK

Research studies have addressed energy conservation problems for computational tasks in different ways, from physical to logical, hardware to software, depending on their interests and objectives. People in computer systems have provided solutions based on DVFS adaptively scaling the processing capacity of CPUs. On the other hand, architectural and platform-level researchers have developed segmentalized levels of cloud virtualization that significantly enhance resource utilization in data centers. Also, many kinds of consolidation techniques have become feasible thanks to the cutting edge virtualization technologies, e.g., Containerization [17].

This study belongs to the literature on how to efficiently utilize those novel technologies in the right place. A large volume of related work has been published in the past decade, so we even reached a survey on surveys of energy efficiency in cloud-related environments [18]. As we largely divide the decision types for green cloud computing into server consolidation and real-time operation (see Fig. 2), the surveyed literature is arranged into the following two subsections.

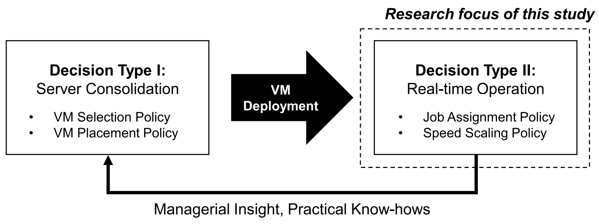


FIGURE 2. Decision types for green cloud data center computing.

A. SERVER CONSOLIDATION

In cloud data centers, servers are consolidated by virtualization and containerization technologies with efficient management of virtual machines (VMs) running on PMs. Literature on this resource management problem proposes various strategies regarding VM migration comprised by VM selection (decision on which VM to be migrated) and VM placement (decision on which PM to host the selected VMs).

Ye *et al.* [13] investigated how to assign VMs efficiently to available PMs. The authors proposed a many-objective VMs placement model considering energy consumption, resource utilization, and load balancing. The suggested method showed better performance than the previous studies with the accompanied metaheuristic algorithm named EEKnEA (Energy-Efficient Knee Point-Driven Evolutionary Algorithm). Mustafa *et al.* [9] proposed joint resource allocation techniques considering the issues of energy consumption and SLA violation. Their methods mainly focused on allocating and migrating VMs based on the dynamic threshold mechanisms that exploit servers' capacity and power information. Simulation results have shown that exploiting the real-time energy state and CPU capacity led to an effective resource management scheme. Remesh Babu *et al.* [19] proposed an SLA-aware VM allocation strategy for load balancing based on a load prediction model. The paper considered various SLA types and showed effectiveness in reducing SLA violations and load balancing. Buchbinder *et al.* [20] studied VM scheduling problem in both offline and online settings. They designed novel algorithms that exploit certain predictions about the workload and showed that the extra information gives significant improvements regarding the problem. While most related studies pursued energy efficiency utilizing migration approaches, Khan *et al.* [21] pointed out that migration itself is expensive in terms of energy consumption and performance degradation. Proposing an energy-performance-aware allocation and migration techniques that take migration cost into account, they found that not using dynamic consolidation could be more cost-efficient.

In a more macroscopic viewpoint, a body of studies focused on drawing useful managerial insights and supporting decision-making by investigating the solution structure of mathematical optimization models. Gallego Arrubla *et al.* [22] suggested a unified mixed integer programming (MIP) model that incorporates virtualization, workload routing, DVFS, and powering on/off servers simultaneously. Using the MIP model, they answered eight research questions on data center energy efficiency. Several heuristic algorithms were accompanied to provide solution methods to solve relatively large-sized problems. However, scalability remained an issue since the algorithms still cannot handle the realistic-sized problem. Cho and Ko [23] remodeled the unified MIP model developed in [22] for revisiting some of the doubtful research conclusions about operating practices in data centers. As a result of fixing the original model, they drew different insights to the previous study. While mathematical models in [22], [23] simplified stochastic nature in cloud computing workloads using average and majorizing approaches, Kwon [16] explicitly introduced a modern uncertainty quantifying concept to the mathematical models based on two-stage stochastic programming with a chance- and risk-constrained optimization. Compared to the too conservative solutions from the previous studies, it suggested a more reasonable server provisioning strategy that guarantees a certain level of QoS.

B. REAL-TIME OPERATION

Studies on this phase care the operational aspects of cloud data centers to which our paper belongs. The main concern in this phase includes appropriate job assignment (or dispatching) to servers for load balancing, demand-adaptive speed scaling of computing servers, and their joint optimization.

The research problems of job assignment in distributed systems started in the late 1970s. One of the most related studies among the oldest literature was conducted by Bonomi [24]. The author investigated the well-known Join-the-shortest-queue (JSQ) policy for parallel PS servers and demonstrated that it offers a good solution to the load balancing problem, although not necessarily optimal. Both academic and industrial attention regarding load balancing topics then has moved to cloud data centers for the past decades. Alongside, Bose and Kumar [4] provided a survey on load balancing in terms of computational workload, whereas Zhang *et al.* [5] presented another survey in terms of network load balancing.

For jointly optimizing the load balancing and server-side resource scheduling considering energy efficiency and service quality, Liu *et al.* [25] presented an integer program accompanied with a heuristic solution approach called "Most Efficient Server First." While the study was meaningful as they formally stated the problem and suggested a rough solution sketch, it has limitations that load balancing becomes arbitrary for servers with the same efficiency (i.e., identical servers). Ko and Cho [12] proposed a new load balancing and speed scaling framework that combined a distributed optimization algorithm with modern queueing theoretic analysis for taking into account the tail probability of response time. Despite the novelty in terms of methodological aspect, some technical requirements such as known a priori stationary workload processes restricted its practicality.

While the literature mentioned above mostly concentrated on the practical issues in cloud data centers, another body of literature has pursued theoretically meaningful results inspired by the industry problems. Wierman *et al.* [26] examined fundamental energy-performance tradeoff in computer speed scaling in the three metrics: optimality, fairness, and robustness. Wentao *et al.* [27] studied optimal load balancing for a certain type of cloud architecture that well reflects machine learning applications. Kwon and Gautam [28] and Cho and Ko [29] investigated methods to time stabilize the performance of stochastic service systems that well-model cloud data centers. Anton *et al.* [30] first showed that a redundancy system (e.g., MapReduce) could help improve the performance of data center computing in case the servers' capacities are sufficiently heterogeneous. Recently, Harchol-Balter [1] published a seminal paper that examines the open problems in queueing theory inspired by data center computing industry. The paper presented new queueing models, workload characteristics, and performance metrics that are all helpful for improving the operations of cloud data centers.

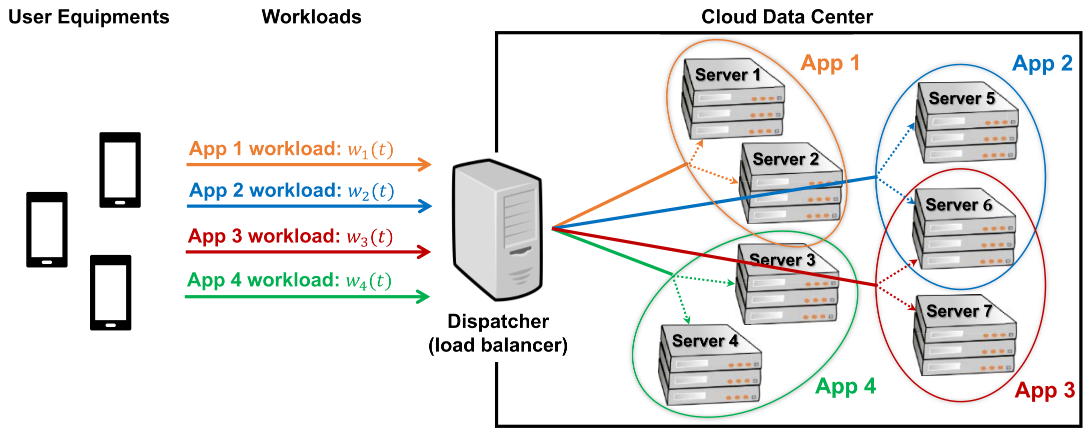


FIGURE 3. An abstracted cloud data center (descriptive case: 4 applications and 7 servers).

III. PROBLEM DESCRIPTION

Fig. 3 describes a cloud data center consisting of a centralized job dispatcher and heterogeneous processor-sharing (PS) PMs (i.e., servers) having time-sharing processors. A number of user equipments (UEs) running various cloud computing applications send their requests to the data center. At the entrance, there is a centralized job dispatcher that assigns the incoming jobs to proper servers.

We consider the virtualization technology that enables a single PM to process cloud job requests from multiple application types; server 6 in Fig. 3 shows an illustrative situation in which a single server is associated with two applications (App 2 and App 3). Regarding the virtualization, we emphasize that deploying an energy-efficient consolidation strategy is another important research topic on green cloud computing. Throughout this paper, however, we concentrate on a specific timings during which the association between the applications and servers is fixed; we assume no VM migration within the time horizon.

On the physical level resource management for each server, we consider the Dynamic Voltage/Frequency Scaling (DVFS). DVFS is an off-the-shelf technology that enables the adaptation of CPU's performance to workload [31], motivated by the need to achieve higher utilization of computing resources. Thanks to the DVFS feature, modern CPU has ability to scale up and down its processing speed dynamically.

Given the system described above, two decision points to be chosen are (i) the job assignment policy for the centralized job dispatcher and (ii) the speed scaling policy for the servers. The ideal decision should reduce the power consumption at the lowest level capable of processing all the user requests while guaranteeing the desired QoS. Throughout this paper, we adopt a popularly used performance metric, *response time*, to quantify the QoS required in cloud data centers, i.e., the servers should give the best effort to finish every job within a prespecified time budget.

In the next section, we begin by interpreting the system using the queueing-theoretic viewpoint as it provides a strong tool to describe the shape of dynamic systems. This will abstract out the important characteristics of data center operating decisions.

IV. MODEL AND FORMULATION

This section formally states the problem using mathematical notations based on the queueing-theoretic interpretation.

A. NOTATION

Table 1 provides a summary of main notations that help explain the system dynamics mathematically.

1) PREDEFINED SETS

We consider a set of applications $\mathcal{A}$ indexed by i (i.e., $i \in \mathcal{A}$) that need to run on a set of servers $\mathcal{S}$ indexed by j (i.e., $j \in \mathcal{S}$). The subscripted sets $\mathcal{A}_j$ and $\mathcal{S}_i$ denote the association between application types and servers. More precisely, the set $\mathcal{A}_j$ comprises the indices of applications that server j is hosting; the jobs of applications in $\mathcal{A}_j$ can be dispatched to server j. Inversely, the set $\mathcal{S}_i$ consists of the indices of the servers that are hosting the application i, i.e., a job of application i can be dispatched to one of the servers in $\mathcal{S}_i$.

2) SERVER RELATED PARAMETERS

We have deterministic parameters that specify a cloud data center. The servers are heterogeneous in terms of their power consumption behavior. To be more specific, we adopt a well-defined polynomial function that is convex on the positive real line used in many previous studies [11], [26], [31]; server j has a power function $p_j(\mu_j) \equiv \alpha_j + m_j \mu_j^{n_j}$ where μ_j is the speed of server j and α_j, m_j, n_j are predefined constants with $\alpha_j, m_j \geq 0$ and $n_j \geq 2$. The instantaneous speed of each server $\mu_j(t)$ (will be explained below) is assumed to be continuously controllable between the lower and upper bounds: $\mu_j(t) \in [\gamma_j, \Gamma_j]$ for $j \in \mathcal{S}$ and at any time t. Regarding the QoS, let $R_j(t)$ be the response time of a job that joins server j at time t. Since we have assumed that an user's satisfaction is attained by the response time that is not longer than a constant δ_j, the system operator should try to keep $R_j(t) \leq \delta_j$ for all $j \in \mathcal{S}$ and $t > 0$.

TABLE 1. Main notations used in our formulation.

Sets:	
$\mathcal{A}$	set of applications ($i = 1, \ldots, \lvert\mathcal{A}\rvert$)
$\mathcal{S}$	set of servers ($j = 1, \ldots, \lvert\mathcal{S}\rvert$)
$\mathcal{A}_j$	set of applications hosted by server j
$\mathcal{S}_i$	set of servers hosting application i
Server related parameters:	
$p_j(\mu_j)$	instantaneous power consumption of server j when running at speed μ_j (unit: Wh/time)
γ_j	speed lower bound of server j (unit: bits/time)
Γ_j	speed upper bound of server j (unit: bits/time)
$R_j(t)$	response time of server j for the job joined at time t (unit: time)
δ_j	response time budget for each job in server j (unit: time)
Workload related parameters:	
$\lambda_i(t)$	time-varying arrival rate of application i's job (unit: #job/time)
T_i	base inter-arrival time of application i's job with mean τ_i (unit: time)
S_i	size of application i's job with mean β_i (unit: bits/job)
$w_i(t)$	time-varying workload of application i (unit: bits/time)
Decision variables:	
$r_{ij}(t)$	ratio of application i's workloads designated to server j at time t (range: [0,1])
$\mu_j(t)$	speed of server j at time t (unit: bits/time)

3) WORKLOAD RELATED PARAMETERS

Though most ingredients in Table 1 look straightforward, the parameters that quantify cloud workload characteristics include some tricky concepts. As described in Section III, we consider a time-dependent arrival rate accompanied by a (nonexponential) random job size. The nonstationary non-Poisson process (NSNP) is one of the well-known stochastic processes that capture the properties. An NSNP arrival process is defined by a time-dependent arrival rate function $\lambda(t)$ and a base inter-arrival time T that is a random variable having mean τ and the squared coefficient of variation (SCV) C_a^2. Together with a random job size S with mean β and SCV C_s^2, the workload process $w(t)$ can be expressed by $w(t) = \beta\lambda(t)$. Attaching the subscript i, the application-specific workload is now expressed by $w_i(t) = \beta_i\lambda_i(t)$. See Cho and Ko [29] for more details about the NSNP and $GI_t/GI_t/1/PS$ model.

4) DECISION VARIABLES

Two real-numbered decision variables $r_{ij}(t)$ and $\mu_j(t)$ indicate decisions on job routing and speed scaling, respectively. First, $r_{ij}(t)$ decides the proportion of workload of application i that is assigned to server j at time t; $\sum_j r_{ij}(t) = 1$ for all application i and at any time t. Second, $\mu_j(t)$ determines the real-time speed of each server j at time t.

B. MATHEMATICAL FORMULATION

Using the notations explained above, we formulate an optimization problem (P1) for minimizing global power consumption in a cloud data center during a planning horizon $[0, \mathcal{T}]$:

$$\min_{r_{ij}(t),\mu_j(t)} \int_0^{\mathcal{T}} \sum_{j\in\mathcal{S}} p_j\left(\mu_j(t)\right) \mathrm{d}t \tag{1a}$$

$$\text{s.t.} \sum_{j\in\mathcal{A}_i} r_{ij}(t) = 1, \quad \forall i \in \mathcal{A},\ \forall t \in [0, \mathcal{T}], \tag{1b}$$

$$0 \le r_{ij}(t) \le 1, \quad \forall i \in \mathcal{A},\ \forall j \in \mathcal{S},\ \forall t \in [0, \mathcal{T}], \tag{1c}$$

$$\int_0^{\mathcal{T}} \sum_{i\in\mathcal{S}_j} w_i(t) r_{ij}(t)\mathrm{d}t < \int_0^{\mathcal{T}} \mu_j(t)\mathrm{d}t, \quad \forall j \in \mathcal{S}, \forall t \in [0, \mathcal{T}], \tag{1d}$$

$$\gamma_j \le \mu_j(t) \le \Gamma_j, \quad \forall j \in \mathcal{S},\ \forall t \in [0, \mathcal{T}], \tag{1e}$$

$$R_j(t) \le \delta_j, \quad \forall j \in \mathcal{S},\ \forall t \in [0, \mathcal{T}]. \tag{1f}$$

The objective function (1a) minimizes the total power consumption over the planning horizon. Constraints (1b) and (1c) ensure the sum of splitted workloads is equal to the original input workload. Constraint (1d) guarantees each server does not explode during the planning horizon. This corresponds to the stability condition for a time-varying queue:

$$\lim_{t\to\infty} \frac{\beta \int_0^t \lambda(s)\mathrm{d}s}{\int_0^t \mu(s)\mathrm{d}s} < 1.$$

Constraint (1e) restricts the range of server speed. Lastly, the service level in terms of response time is specified by constraint (1f).

1) NEED FOR WORK-AROUND APPROACH

If we are given the optimal values $r_{ij}^*(t)$ and $\mu_j^*(t)$, one of the natural usages of them to achieve asymptotically optimal energy efficiency is to apply the following operating scheme:

- Job assignment policy (probabilistic routing): For an arriving job of type i at time t, dispatch it to server j with probability $r_{ij}^*(t)$.
- Dynamic speed scaling policy (continuous time updating): For server j, adjust the speed at $\mu_j^*(t)$.

However, finding the optimal solutions to $r_{ij}(t)$ and $\mu_j(t)$ in real-time is limited due to the complicating factors such as nonlinear functional objective, nonstationarity, and uncertainty; it requires to solve similar problems in [12], [25]. Instead, we come up with heuristic approaches that we call *least-power-consuming* (LPC) job assignment policy and *minimizing earliness* (ME) speed scaling policy, jointly called LPC-ME, to find a high quality solution for $r_{ij}(t)$

and $\mu_j(t)$. In the next section, we explain the design principle of LPC-ME to achieve our goal.

V. ALGORITHM DESIGN

We adopt a generic formed power function for each server as discussed in [11], [31]:

$$p(\mu) = \alpha + m\mu^n, \text{ where } \alpha, m \geq 0 \text{ and } n \geq 2. \tag{2}$$

With this convex polynomial power function, our design principle starts from the following simple question: *Shorter job completion time with higher processing rate* or *longer job completion time with lower processing rate*, which one is more energy-efficient? To answer the question, we state the following proposition.

Proposition 1: For a power function in equation (2) and a workload of size s, there exists a most energy-efficient speed μ^* to finish the workload given by $\mu^* = [m/(n-1)\alpha]^{1/n}$.

Proof: Define a function $f(\mu; s) \equiv (s/\mu)p(\mu)$ to be the total power consumption required to finish a workload of size s using a constant speed μ. The total power consumption is calculated by the power function $p(\mu)$—power consumption per unit time–multiplied by time to finish the workload (s/μ). Then, the first derivative of the function in terms of μ is

$$\frac{\mathrm{d}}{\mathrm{d}\mu} f(\mu; s) = s\left[-\frac{\alpha}{\mu^2} + m(n-1)\mu^{n-2}\right],$$

with its minimizer $\mu^* = [\alpha/m(n-1)]^{1/n}$. ■

Proposition 1 implies two insights. First, an optimal server speed exists in terms of total power consumption given a constant job size. Hence, even if the job is not urgent, running the server at speed μ^* is optimal due to the energy-performance tradeoff. Second, when the server should run faster than μ^* because of the imminent job, fully utilizing the given response time budget is more energy-efficient than finishing it hastefully.

Our goal now is to manage the servers' speeds towards just meeting the jobs' deadlines but faster than the most energy-efficient point μ^*. To explain, we introduce the following queueing theoretic notations:

- $\mathcal{J}_j(t)$: Index set of jobs in server j at time t (we use k for indexing the jobs, i.e., $k \in \{1, \ldots, |\mathcal{J}_j(t)|\}$)
- $S_j^k(t)$: Remaining workload of job k in server j at time t
- $A_j^k(t)$: Arrival time of job k in server j at time t
- $Q_j(t)$: Number of jobs in server j at time t

Note that the above notations are stochastic processes in the sense that their values are randomly varying as time evolves. What we want is to finish all the jobs in a server within their due time. Each job's deadline is determined upon arrival by the server-specific completion time deadline δ_j. Now, we propose the following real-time algorithm, which is the main result of this paper.

In the following two subsections, we explain the derivation of Algorithm V. The sequence of explanation is from ME to LPC as we regard this order will be more effective in clearly delivering the rationale behind the idea.

Algorithm 1 Operate a Cloud Data Center By the Combination of the Following Two Policies

```
1) Job assignment policy: Join the least power consuming (LPC) server
    • Dispatch an arriving job to the instantaneously least power consuming associated server. That is, choose a server $j$ for a job of application $i$ at time $t$ such that
          $j \leftarrow \operatorname*{argmin}_{j \in \mathcal{S}_i} p_j\left(\mu_j(t)\right).$
2) Dynamic speed scaling policy: Minimize earliness (ME) of the work-in-process jobs' completions
    • Update the processing rate of each server towards minimizing the jobs' earliness. That is, keep the speed $\mu_j(t)$ of each server $j$ in the following manner:
          $\mu_j(t) \leftarrow \max\left[\mu_j^*, \max_{k \in \mathcal{J}_j(t)}\left\{\frac{Q_j(t)S_j^k(t)}{\delta_j - t + A_j^k(t)}\right\}\right],$   (3)
      where the $\mu_j^*$ is as defined in Proposition 1.
        – If the set of configurable processing speeds is discrete (e.g., P-States of modern CPUs [32]), choose the smallest value larger than the value calculated by expression (3).
```

A. ME: SERVER's DYNAMIC SPEED SCALING POLICY

The term *earliness*, by definition, means the quality of coming early or earlier. With the meaning of earliness in mind, consider the kth job in server j at time t. Its allowed time to stay in the server at time t is $\delta_j - \left(t - A_j^k(t)\right)$ by the response time requirement. That means we must scale the server speed towards finishing every kth job within its remaining time, $\delta_j - \left(t - A_j^k(t)\right)$, to keep the response time less than δ_j. Regarding the PS scheduling policy—all jobs in the system evenly share the processor at any given time, and multiple jobs that are supposed to be completed by their deadlines, server j's earliness-minimizing speed at time t must be chosen by the following expression:

$$\min_{k \in \mathcal{J}_j(t)} \left\{\mu_j(t) : \frac{S_j^k(t)}{\mu_j(t)/Q_j(t)} \leq \delta_j - \left(t - A_j^k(t)\right)\right\}. \tag{4}$$

Then, the expression (3) in Algorithm V is obtained by manipulating the terms in expression (4) and Proposition 1. Note that this speed scaling reduces the earliness of the job completions as smallest as possible when the job requires higher speed than μ_j^*.

B. LPC: DISPATCHER's JOB ASSIGNMENT POLICY

The LPC routing policy is motivated both by Proposition 1 and well-known greedy heuristics such as Join-the-shortest-queue (JSQ) and Least-Work-Left (LWL) [7]. The underlying idea of LPC is as follows. Assuming the speed scaling of each server is done by ME as in Algorithm V, dispatching a job to a server increases the server's speed. The following remark explains this behavior.

Remark 1: For a processor sharing server under minimizing earliness speed scaling, receiving a job increases the server speed monotonically.

Proof: Let $N_j(t)$ be the number of jobs being processed in a server j at time t, i.e., $N_j(t) \equiv |\mathcal{J}_j(t)|$. Then, the scaled speed of a server right after receiving a job can be expressed as follows:

$$\max\left[\mu_j^*, \max_{k\in\mathcal{J}_j'(t)}\left\{\frac{\left(Q_j(t)+1\right)S_j^k(t)}{\delta_j - t + A_j^k(t)}\right\}\right], \tag{5}$$

where $\mathcal{J}_j'(t) \equiv \mathcal{J}_j(t) \cup \{N_j(t)+1\}$. Since $Q_j(t) \leq Q_j(t)+1$ and $\mathcal{J}_j(t) \subseteq \mathcal{J}_j'(t)$, we notice that the expression (5) is greater than or equal to the expression (3). ■

Recalling that the server's power function is convex polynomial, increment of speed also results in increment of power consumption. Under this condition, an energy-efficient dispatcher should find a server which will increase its power consumption as small as possible upon receiving a job.

If the servers are homogeneous, dispatching a job to the server with minimum speed must be the most energy-efficient decision. Practically, however, servers are heterogeneous in terms of the power function. Hence, we just choose the server with the instantaneously least power consumption, which the proposed algorithm remains a greedy heuristic decision.

1) HOW IS THE LOAD BALANCED THROUGH LPC-ME?

Similar to Proposition 1, we may consider the opposite case where a job completion event decreases the instantaneous power consumption. By this mechanism, job arrival and completion dynamically scale up and down the server speed. Since LPC policy always assigns an incoming job to a server with the least power consumption, we notice that load balancing in a cloud data center is attained under LPC-ME combination.

In the next section, we will introduce a numerical example with graphical description to explain how the proposed method is working towards improving energy efficiency while keeping QoS condition and also balancing loads.

VI. NUMERICAL EXAMPLE

This section provides two illustrative examples with specific numbers to help understanding the concept of the proposed policies. We adopt the specific values from Tables 4, 5 which will be introduced in Section VII for performance evaluation.

A. WORKING MECHANISM

Graphics in Figs. 4, 5 describe the working mechanism of LPC-ME combination. We assume a situation that two consecutive job requests of application 4 arrive to a cloud data center. Fig. 4 depicts the operations for the first job and Fig. 6 describes the ones for the second job. The left half (i.e., drawings) of each figure shows the operational description whereas the right half shows server-specific instantaneous power usage according to processing speed (i.e., graph) and information about the work-in-process jobs: job index, remaining workload, and arrival time (i.e., table).

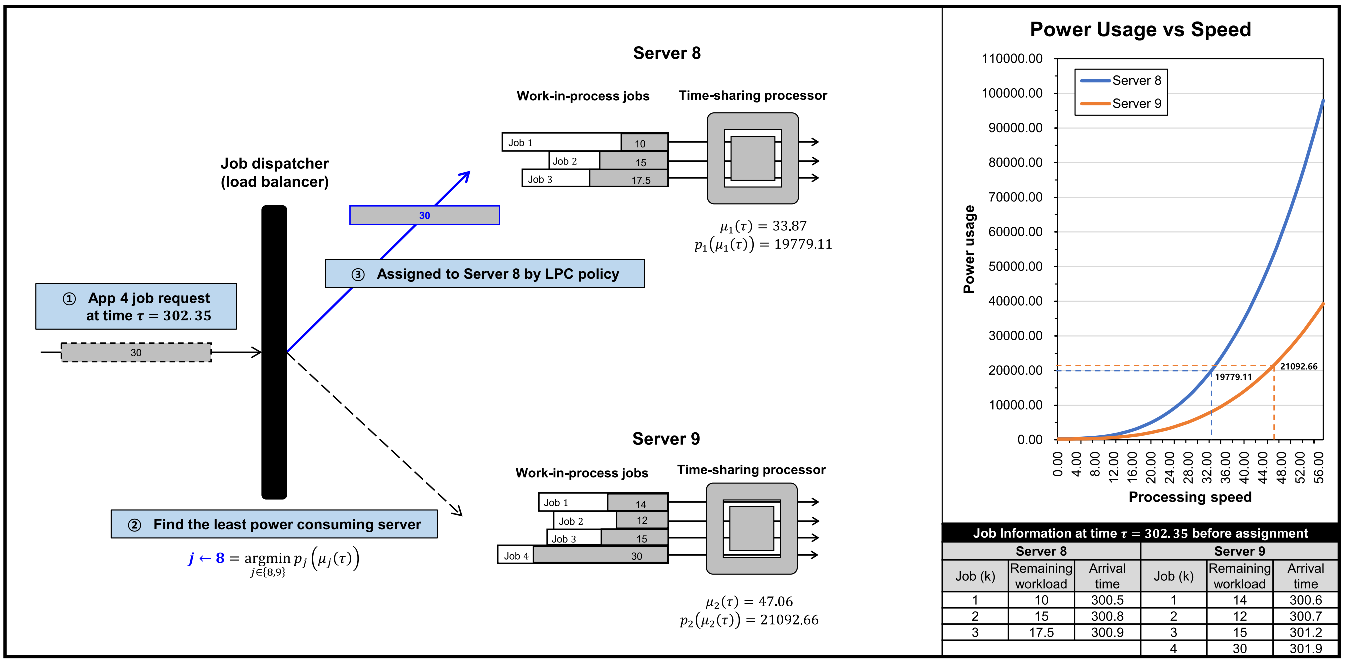
The following enumeration explains step-by-step operations of job dispatcher and servers (refer also to the sky blue colored stickers in Figs. 4, 5).

① Fig. 4(a): A job request of application 4 arrives at time $\tau = 302.35$ with job size 30.
② Fig. 4(a): A job dispatcher tries to find the least power consuming server among the associated servers with application 4. Since servers 8 and 9 are associated and server 8 is currently consuming less power (19779.11 per unit time) than server 9 (21092.66 per unit time), the dispatcher chooses server 8 as the least power consuming server.
③ Fig. 4(a): The dispatcher assigns the job to server 8 by LPC policy.
④ Fig. 4(b): Since the processor is time-sharing and a new job is added, server 8 needs to update its speed to satisfy the QoS condition for each job. Based on the remaining workload and arrival time for each work-in-process job, server 8 changes its speed to 45.16 by ME policy in expression (3) and the power usage is increased to 46404.18 per unit time accordingly. Then, each job will be processed with the rate of 11.29 (= 45.16/4) per unit time and hence all the jobs will be completed within their deadlines. Note that the processing speed will be dynamically updated regularly as well as whenever their is a new job request or a job completion.
⑤ Fig. 5(a): Another job request of application 4 arrives at time $\tau' = 302.36$ with job size 30, which is 0.1 unit time after the previous job.
⑥ Fig. 5(a): The dispatcher again tries to find the least power consuming server among the associated servers. Since server 9 is consuming less power (21092.66 per unit time) than server 8 (46404.18 per unit time), the dispatcher chooses server 9 as the least power consuming server.
⑦ Fig. 5(a): The dispatcher assigns the job to server 9 by LPC policy.
⑧ Fig. 5(b): Based on the remaining workload and arrival time for each work-in-process job, server 9 changes its speed to 58.59 per unit time by ME policy and the power usage is now increased to 40481.28 per unit time accordingly.

B. ENERGY EFFICIENCY

Fig. 6 shows a numerical example that explains why minimizing earliness is energy efficient. We assume a situation in server 9 and adopt numerical values from Table 5.

The figure consists of three parts. First, the upperleft table contains an illustrative job information (job size and response

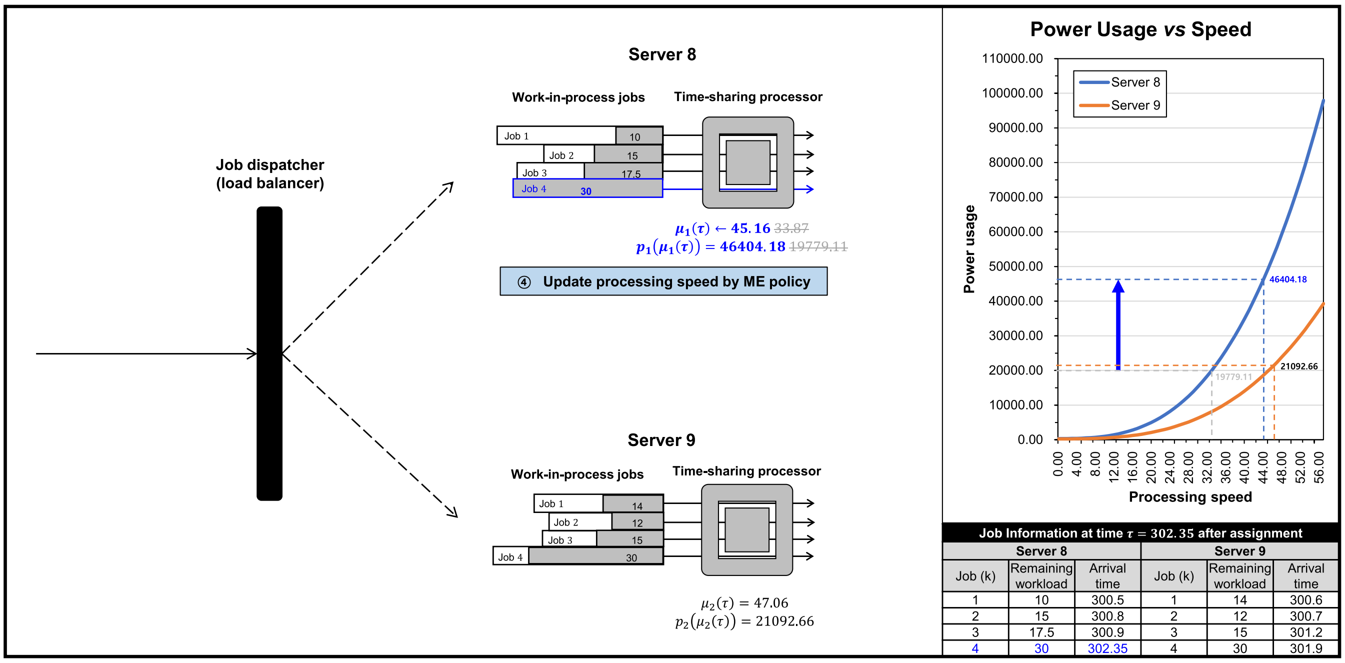


Server 8
Work-in-process jobs
Time-sharing processor
Job 1 10
Job 2 15
Job 3 17.5
Job dispatcher (load balancer)
30
$\mu_1(\tau) = 33.87$
$p_1(\mu_1(\tau)) = 19779.11$
③ Assigned to Server 8 by LPC policy
① App 4 job request at time $\tau = 302.35$
30
Server 9
Work-in-process jobs
Time-sharing processor
Job 1 14
Job 2 12
Job 3 15
Job 4 30
② Find the least power consuming server
$j \leftarrow 8 = \underset{j\in\{8,9\}}{\operatorname{argmin}}\, p_j\left(\mu_j(\tau)\right)$
$\mu_2(\tau) = 47.06$
$p_2(\mu_2(\tau)) = 21092.66$
Power Usage vs Speed
Server 8
Server 9
Power usage
Processing speed
19779.11
21092.66

Job Information at time $\tau = 302.35$ before assignment

Server 8			Server 9		
Job (k)	Remaining workload	Arrival time	Job (k)	Remaining workload	Arrival time
1	10	300.5	1	14	300.6
2	15	300.8	2	12	300.7
3	17.5	300.9	3	15	301.2
			4	30	301.9

(a) Operations before the first job assignment. A new job request is arrived and about to be assigned to server 8 by LPC job assignment policy.



Job Information at time $\tau = 302.35$ after assignment

Server 8			Server 9		
Job (k)	Remaining workload	Arrival time	Job (k)	Remaining workload	Arrival time
1	10	300.5	1	14	300.6
2	15	300.8	2	12	300.7
3	17.5	300.9	3	15	301.2
4	30	302.35	4	30	301.9

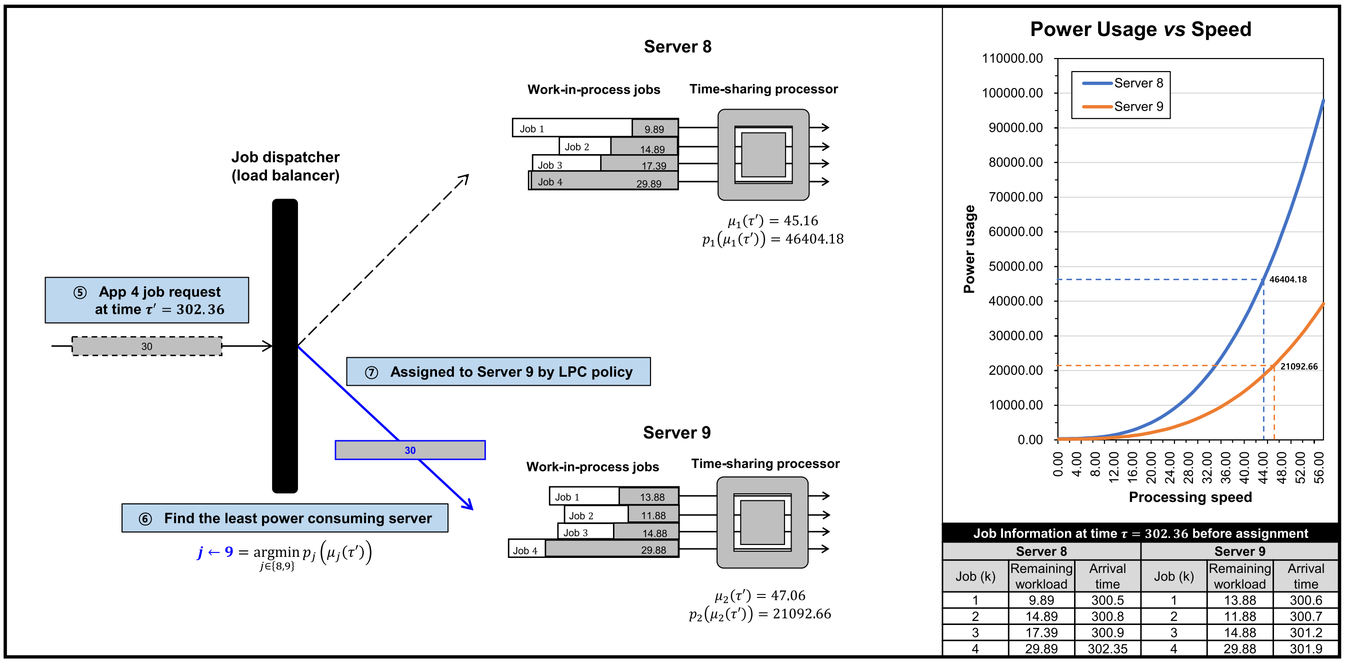
(b) Operations after the first job assignment. Server 8 receives the job and updates the processing speed by ME speed scaling policy.

FIGURE 4. An illustrative situation where a job request arrives to a cloud data center at time 302.35. The numerical values are adopted from Tables 4, 5.

time budget) to be processed. Second, the upperright and lowerleft tables and the graph want to show the varying power consumption according to the following levels of earliness of job completion:

- Non-Power-Aware (NPA): Does not care about the earliness but gives best effort to process the job.
- 75% Early: Tries to complete a job within 75% of given response time budget.
- 50% Early: Tries to complete a job within 50% of given response time budget.
- 25% Early: Tries to complete a job within 25% of given response time budget.
- Minimizing Earliness (0% Early): Tries to complete a job definitely at the deadline.

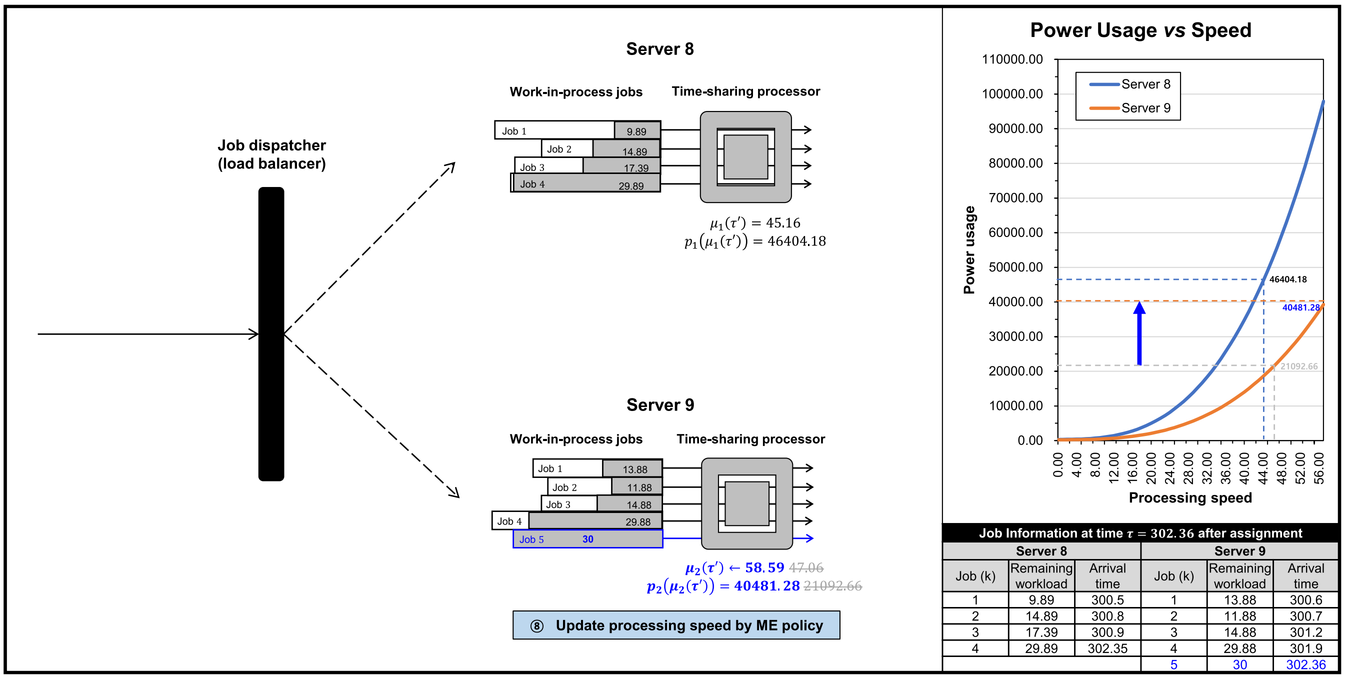
For each level of earliness, we calculate the scaled speed and corresponding working time requirement



Job Information at time $\tau = 302.36$ before assignment

Server 8			Server 9		
Job (k)	Remaining workload	Arrival time	Job (k)	Remaining workload	Arrival time
1	9.89	300.5	1	13.88	300.6
2	14.89	300.8	2	11.88	300.7
3	17.39	300.9	3	14.88	301.2
4	29.89	302.35	4	29.88	301.9

(a) Operations before the second job assignment. Another new job request is arrived and about to be assigned to server 9 by LPC job assignment policy.



Job Information at time $\tau = 302.36$ after assignment

Server 8			Server 9		
Job (k)	Remaining workload	Arrival time	Job (k)	Remaining workload	Arrival time
1	9.89	300.5	1	13.88	300.6
2	14.89	300.8	2	11.88	300.7
3	17.39	300.9	3	14.88	301.2
4	29.89	302.35	4	29.88	301.9
			5	30	302.36

(b) Operations after the second job assignment. Server 9 receives the job and updates the processing speed by ME speed scaling policy.

FIGURE 5. An illustrative situation where a job request arrives to a cloud data center at time 302.36 (0.1 unit time after the previous job arrival). The numerical values are adopted from Tables 4, 5.

(see yellow-colored cells). Then, we notice that ME is the most energy efficient compared to the others (see green-colored cells and accompanied graph).

Here, we emphasize that ME policy is nothing but trying to fully utilize the given response time budget hence results in increment of response time. However, the slight response time increment that does not violate the job deadline will not be considered critical in terms of overall service quality, which we argue that ME policy is still reasonable as the new speed scaling policy for the physical servers in cloud data centers.

VII. PERFORMANCE EVALUATION

In this section, we demonstrate the performance of the proposed method on the large set of randomly generated cloud workloads and a virtual cloud data center. All environments

An arbitrary job to be processed	
Job size	**Response time budget**
51.00	3.00

Scaled speed and required working time to complete the job (for each policy)			
Speed scaling policy	Job deadline	Scaled speed	Required working time
NPA (Best Effort)	ASAP	102.00	**0.50**
75% Early	within 0.75	68.00	**0.75**
50% Early	within 1.50	34.00	**1.50**
25% Early	within 2.25	22.67	**2.25**
ME (0% Early)	within 3.00	17.00	**3.00**

Total power consumption to complete the job according to working time (for each policy)					
Working time	NPA	75% Early	50% Early	25% Early	ME
0.00	0.00	0.00	0.00	0.00	0.00
0.25	53122.90	15784.10	2027.70	644.78	308.15
0.50	**106245.80**	31568.20	4055.40	1289.56	616.30
0.75	159368.70	**47352.30**	6083.10	1934.34	924.45
1.00	212491.60	63136.40	8110.80	2579.13	1232.60
1.25	265614.50	78920.50	10138.50	3223.91	1540.75
1.50	318737.40	94704.60	**12166.20**	3868.69	1848.90
1.75	371860.30	110488.70	14193.90	4513.47	2157.05
2.00	424983.20	126272.80	16221.60	5158.25	2465.20
2.25	478106.10	142056.90	18249.30	**5803.03**	2773.35
2.50	531229.00	157841.00	20277.00	6447.81	3081.50
2.75	584351.90	173625.10	22304.70	7092.60	3389.65
3.00	637474.80	189409.20	24332.40	7737.38	**3697.80**
3.25	690597.70	205193.30	26360.10	8382.16	4005.95
3.50	743720.60	220977.40	28387.80	9026.94	4314.10
3.75	796843.50	236761.50	30415.50	9671.72	4622.25
4.00	849966.40	252545.60	32443.20	10316.50	4930.40
4.25	903089.30	268329.70	34470.90	10961.29	5238.55
4.50	956212.20	284113.80	36498.60	11606.07	5546.70
4.75	1009335.10	299897.90	38526.30	12250.85	5854.85
5.00	1062458.00	315682.00	40554.00	12895.63	6163.00

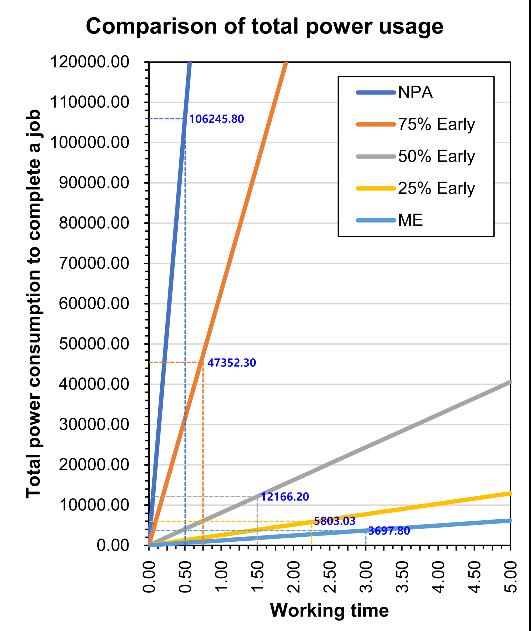


FIGURE 6. A quantitative analysis of energy efficiency according to the earliness of job completion. The server-specific numerical values are adopted from server 9 in Table 5. In short, this figure shows that fully utilizing the given response time budget is most energy-efficient in terms of completing a job.

and policies are implemented in Julia programming language [33] and run on Windows Server (Intel(R) Core(TM) i9-11900K @ 3.50GHz, 32GB of RAM). For fair comparison, we compare the performance with existing benchmarks consist of well-known job assignment policies and off-the-shelf dynamic speed scaling policies.

A. BENCHMARK

1) JOB ASSIGNMENT POLICIES

Table 2 summarizes the set of job assignment policies used for performance comparison. It includes three well-known practical policies and the proposed policy. Note that each policy utilizes different information on congestion: no information, number of jobs, workloads, and instantaneous power usage.

2) DYNAMIC SPEED SCALING POLICIES

Table 3 summarizes the set of dynamic speed scaling policies used for performance comparison. It includes three popular policies and the proposed policy. The existing three policies correspond to a linux kernel feature called *CPUFreq Governors* for state-of-the-art CPUs [34]. Here, we emphasize that the processing performance of CPU can only be configured to one of the predefined P-States, e.g., P0, P1, ..., Pn [32].

TABLE 2. Benchmarking job assignment policies. Each policy utilizes different congestion information.

Job Assignment Policy	Description
Round-Robin (RR)	Choose an associated server in circular order.
Join the Shortest Queue (JSQ)	Choose an associated server with the fewest *number of work-in-process jobs*.
Least-Work-Left (LWL)	Choose an associated server with the least *total remaining workloads*.
Least-Power-Consuming (LPC)	**[Proposed policy]** Choose an associated server with the least *instantaneous power usage*.

Simply put, the proposed ME policy adaptively changes a processor's P-State to what corresponds to the smallest speed larger than the value calculated by expression (3) in Algorithm V.

B. SIMULATION ENVIRONMENT

1) WORKLOAD GENERATION

The pattern of cloud workloads has been analyzed to show daily cycle [35]. That is, people tend to work more during day time and less at night (for example, see Fig. 7). To mimic the real-world workload characteristics that vary periodically with burstiness, we generate NSNPs for sampling the job arrival epochs according to the simulation algorithm developed in [36]. In short, NSNP is a generalization of the well-known non-homogeneous Poisson Process (NHPP).

TABLE 3. Benchmarking dynamic speed scaling policies. The policies correspond to existing DVFS governors implemented in linux kernel [34].

Dynamic Speed Scaling Policy	Description
Non-Power-Aware (NPA)	This correspnds to *Performance* governor. That is, the server is always running at maximum speed.
Power-Aware 1 (PA1)	This corresponds to *Ondemand* governor. That is, the server speed jumps to the maximum when it receives a job request and decreases speed gradually when the server is approaching idle.
Power-Aware 2 (PA2)	This corresponds to *Conservative* governor. That is, the server gracefully increases and decreases its speed according to load estimation.
Minimizing-Earliness (ME)	**[Proposed policy]** This policy adaptively changes a processor's performance state to what corresponds to the smallest speed larger than the value calculated by expression (3) in Algorithm 1.

TABLE 4. Workload generating parameters.

Application (i)	Job Request Frequency				Job Size		
	Rate of Request: $\lambda_i(t)$	Base Distribution: T_i	$\mathbb{E}[T_i]$	SCV$[T_i]$	Base Distribution: S_i	$\mathbb{E}[S_i]$	SCV$[S_i]$
1	$4 - 3\sin(\pi t/1000)$	Lognormal	1	2	Lognormal	5	1.5
2	$2 - 1.5\sin(\pi t/1000)$	Lognormal	1	1.5	Lognormal	10	2
3	$4 - 2.5\sin(\pi t/1000)$	Exponential	1	1	Lognormal	5	1
4	$10 - 3\sin(\pi t/1000)$	Lognormal	1	0.8	Lognormal	2	0.8
5	$5 - 4\sin(\pi t/1000)$	Lognormal	1	2	Lognormal	3	0.5

TABLE 5. Server configuration.

Server (j)	Processing Speed per P-State					Parameters for Power Usage			Response Time Budget (δ_j)	Associated Apps ($\mathcal{A}_j$)
	P0 (Γ_j)	P1	P2	P3	P4 (γ_j)	Param 1 (α_j)	Param 2 (m_j)	Param 3 (n_j)		
1	100	75.00	50.00	25.00	5	150	0.3333	3	3	{1}
2	102	76.50	51.00	25.50	7	250	0.2	3	3	{1}
3	99	74.25	49.50	24.75	6	220	1	3	3	{1,2}
4	105	78.75	52.50	26.25	5	150	0.6667	3	3	{1,2,3}
5	100	75.00	50.00	25.00	7	300	0.8	3	3	{2,3}
6	102	76.50	51.00	25.50	8	350	0.4	3	3	{2,3}
7	100	75.00	50.00	25.00	6	220	0.4286	3	3	{3}
8	105	78.75	52.50	26.25	7	350	0.5	3	3	{4,5}
9	102	76.50	51.00	25.50	8	400	0.6	3	3	{4,5}
10	105	78.75	52.50	26.25	10	700	0.4444	3	3	{5}

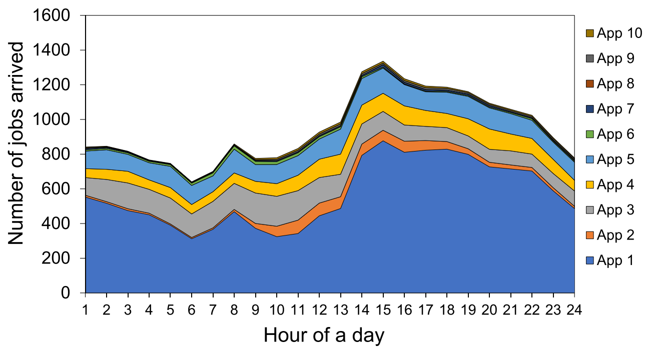


FIGURE 7. Real workload traces for 10 applications for one day collected from publicly available source [22].

TABLE 6. Experimented benchmarking policy combinations.

Index	Benchmark	Description
1	RR-NPA	Best effort all the time
2	RR-PA1	Combinations of state-of-the-arts
3	RR-PA2	
4	JSQ-NPA	
5	JSQ-PA1	
6	JSQ-PA2	
7	LWL-NPA	
8	LWL-PA1	
9	LWL-PA2	
10	LPC-ME	**Proposed method**

Appropriately choosing the input parameters of NSNP (i.e., $\lambda_i(t)$ and distributions of random variables T_i and S_i in Table 1) well describes the realistic cloud workload patterns. Table 4 provides the explicit parameters for generating synthetic workloads from five cloud applications.

We use gradual sine functions as time-varying arrival rates and general probability distributions as job size distributions. First, the time-varying arrival rates are used to reflect the daily cycle of workloads. Second, we mostly use Lognormal distribution, which is heavy-tailed as well as nonexponential, for T_i and S_i as it will simulate quite variable property of data center computing workloads. See [1] for the detailed discussion on the workload property. We also emphasize that we pick the function parameters towards representing *elephants and mice effect*, i.e., few applications with high workload levels and many others with low workload levels [5].

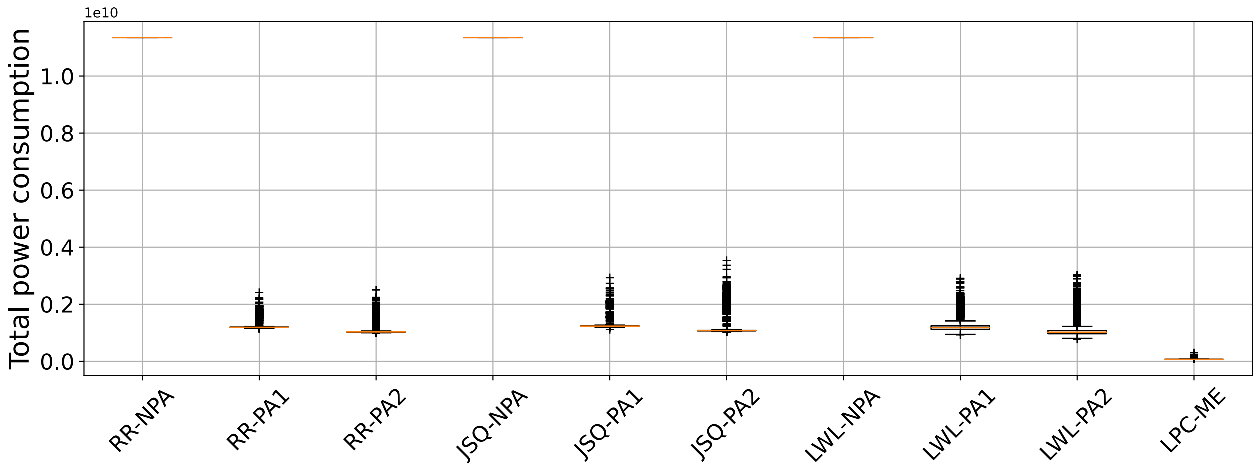


FIGURE 8. **Comparison of the total power consumptions for the benchmarking policy combinations. The cross dots mark outliers and the orange colored line marks median. The numerical data is available from Table 7 in Appendix.**

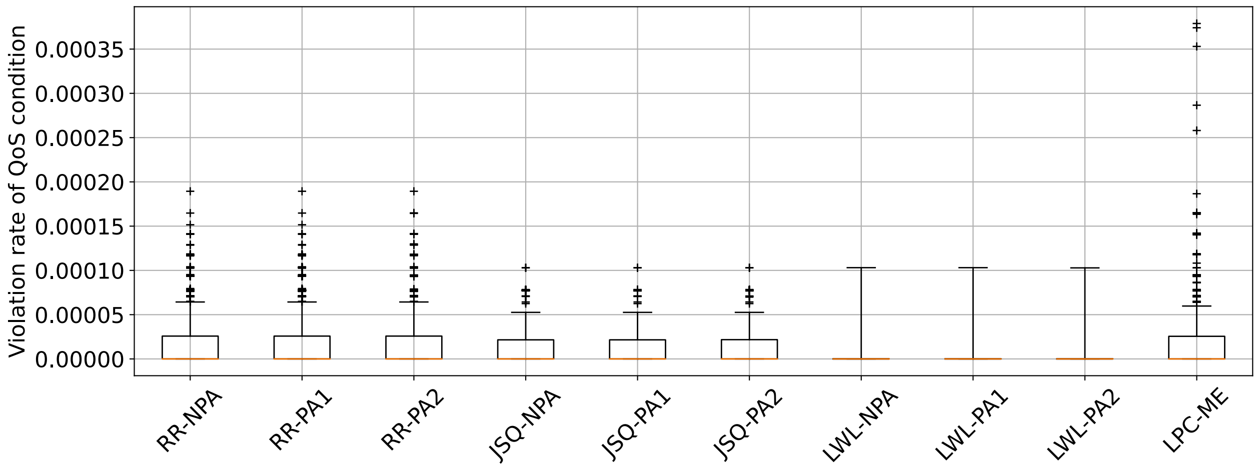


FIGURE 9. **Comparison of the violation rates of QoS conditions for the benchmarking policy combinations. The cross dots mark outliers and the orange colored line marks median. The numerical data is available from Table 8 in Appendix.**

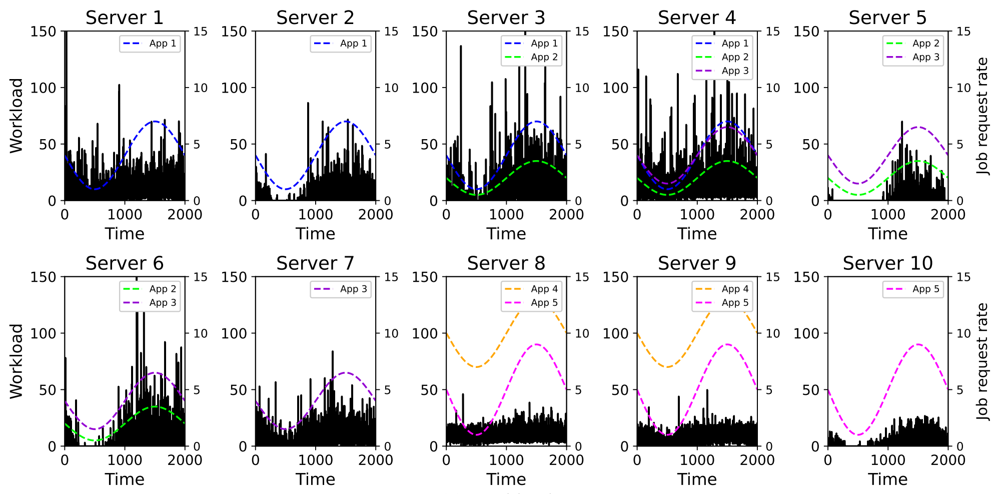
2) SERVER CONFIGURATION

We implement a small cloud data center that comprises ten heterogeneous servers for an illustrative purpose. Since the proposed algorithm does not require solving complicated optimization problems, scalability is not an issue. For example, LPC job assignment policy can be applied by implementing a real-time power usage monitoring feature a job dispatcher; each server periodically reports its instantaneous power usage to the job dispatcher. When it comes to the server's dynamic speed scaling, it does not require information exchange between servers and a dispatcher; each server estimates its instantaneous workload and perceives current speed. Table 5 summarizes the configuration of the 10 servers comprising the virtual cloud data center.

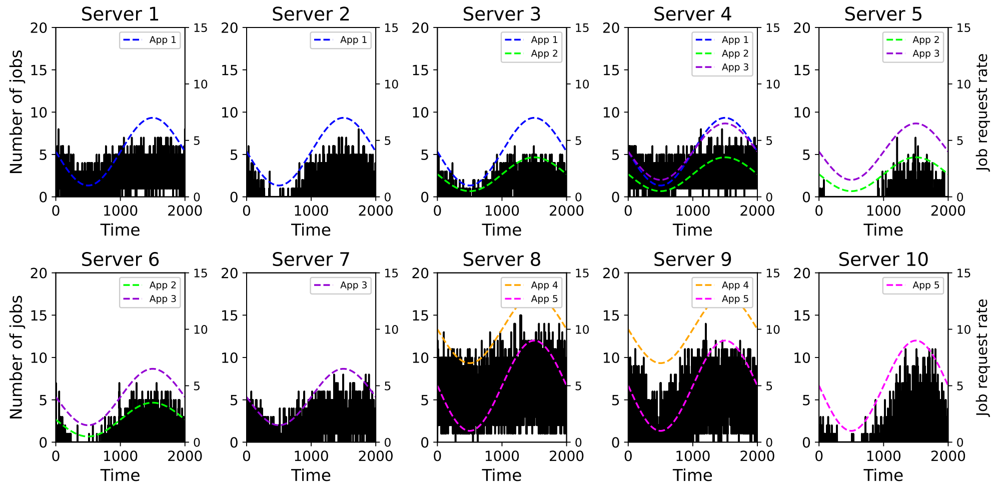
C. RESULTS AND DISCUSSION

We run 10,000 independent replications of 2,000 unit time simulation for each benchmarking policy combinations enumerated in Table 6 to get large enough samples that prevents misinterpretation due to outliers. We use 0.01 unit time for all the servers' regular speed updating intervals.

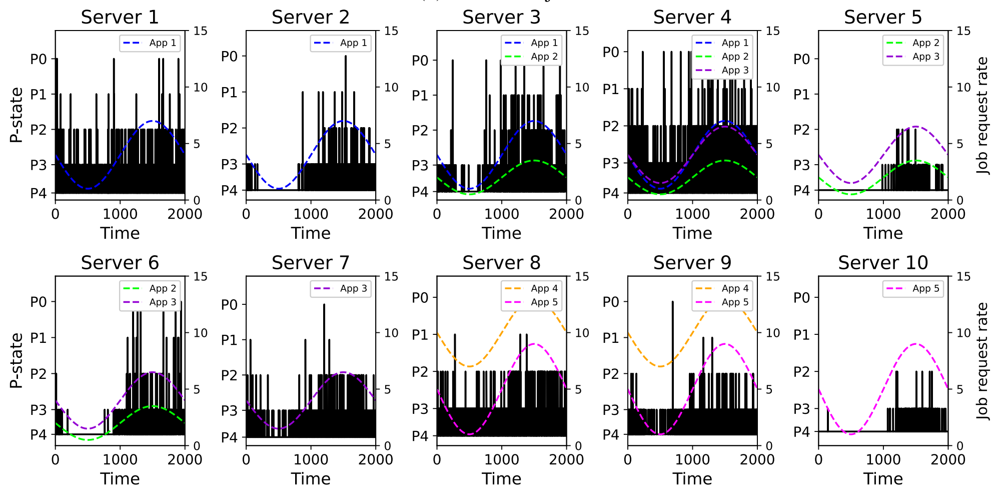
Fig. 8–10, and Tables 7, 8 (in Appendix) provide graphical and statistical summaries of the simulation results obtained from the 10,000 replications. As shown in the overall plots, frequent outliers (cross dots in Figs. 8, 9 and spikes in Fig. 10) are observed because we simulated quite variable and heavy-tailed workloads as discussed in Section VII-B1. In the following three subsections, we will discuss about energy efficiency, service quality, and load balancing effect of the proposed method.



(a) Workloads



(b) Number of jobs



(c) P-States

FIGURE 10. **Real-time server status logged in a single replication of experiments under LPC-ME algorithm. The dotted lines with y-axis on right side show the time-varying job request rates of applications which are associated with each server.**

1) ENERGY EFFICIENCY

Energy efficiency is the primary concern in this research study. As regards, the box plot in Fig. 8 summarizes the total power consumption data obtained from the 10,000 replications of simulation experiment for each benchmarks. As expected, benchmarks with NPA speed scaling policy (i.e., RR-NPA, JSQ-NPA, LWL-NPA) consistently show much higher power consumption compared to those with power-aware speed scaling policies. On the other hand, the policy combinations with power-aware speed scaling policies (i.e., RR-PA1, RR-PA2, JSQ-PA1, JSQ-PA2, LWL-PA1, LWL-PA2, and LPC-ME) show dramatic improvement in energy efficiency. Notably, we found that the proposed LPC-ME combination in average requires less than 10% of power consumed by the other existing power-aware benchmarks (see Table 7 in Appendix). Of course, we need to think about the energy-performance tradeoff and the next subsection will discuss about it.

2) SERVICE QUALITY

Service quality is another main concern in this research study as well as the highest priority of the cloud data center computing. As such, Fig. 9 depicts the violation rates of predefined QoS conditions, i.e., the ratio of delayed jobs out of all the completed jobs. As shown in the figure, we found that the benchmarks with LWL job assignment policy (i.e., LWL-NPA, LWL-PA1, LWL-PA2) outperform other benchmarks including the proposed LPC-ME combination. However, LPC-ME also demonstrates well-managed service quality showing less than 0.001% of violation rates in average and only 0.037% in worst case (see Table 8 in Appendix). In fact, the performance degradation had been predicted since the proposed ME speed scaling policy manages the processing speed tightly in order to fully utilize the given response time budget (recall the last paragraph in Section VI-B). Regarding the significant improvement in energy efficiency shown in Section VII-C1, we conclude that the slight degradation of service quality does not seem to be critical; it provides the contracted service level on the response time with probability higher than 99.999% in most cases.

3) LOAD BALANCING EFFECT

Figs. 10(a)–10(c) portray real-time server-specific status (i.e., workload, number of jobs, and servers' P-States) logged in a single replication of experiments under LPC-ME algorithm. Naturally, all the three measures tend to follow the changing trends of applications' loads. We find here that the servers with the same associated applications (e.g., servers 1, 2 and servers 8, 9) show the similar levels of workloads and number of jobs, which verifies the load balancing effect of the proposed algorithm. In addition, Fig. 10(c) demonstrates the effectiveness of LPC-ME combination in terms of scaling the servers' speeds in reaction to the change of loads.

VIII. CONCLUSION AND FUTURE WORK

This study proposes a simple but effective real-time algorithm for cloud data center computing to achieve better energy efficiency. We interpret the target system as a parallel network of heterogeneous single-server PS queues with multiple types of applications, time-varying job request processes, and controllable processing rates. We develop two policies involving LPC job assignment and ME speed scaling motivated by the convexity of processor's power usage with respect to speed. Numerical simulations have demonstrated that LPC-ME combination consistently outperforms other combinations of existing popular policies in terms of energy efficiency without significant QoS degradation.

We suggest several future research directions based on the issues not fully covered in this paper. First, we did not directly solve the formulated problem since the exact solution approach has not been available for such a problem to the best of our knowledge. If an optimal solution can be found in some ways, we can evaluate the performance gap between our heuristic algorithm and the optimum. Second, incorporating the neglected lower-level computer systems details (e.g., overheads between the executions of decisions) can give another insight regarding cloud data center computing. Third, we omitted the networking aspect in cloud data centers to concentrate on the computing aspect. We believe that jointly considering the two main aspects in cloud data centers (i.e., computing and networking) will have a significant synergy effect in terms of end-to-end performance. Last but not least, implementation on real-world cloud data centers should reveal the uncovered issues not studied in this paper.

APPENDIX

STATISTICAL SUMMARY OF PERFORMANCE RESULTS

See Figs. 8–10, and Tables 7 and 8.

TABLE 7. Statistical summary of total power consumption for the benchmarks.

Index	Benchmark	Statistics			
		Mean	Min	Median	Max
1	RR-NPA	11,347,618,785	11,347,576,490	11,347,633,213	11,347,633,213
2	RR-PA1	1,210,895,904	1,150,884,935	1,190,093,555	2,411,430,474
3	RR-PA2	1,051,107,347	991,201,595	1,029,382,157	2,500,457,506
4	JSQ-NPA	11,347,618,752	11,347,576,490	11,347,633,213	11,347,633,213
5	JSQ-PA1	1,262,894,917	1,112,359,249	1,231,521,452	2,930,781,081
6	JSQ-PA2	1,113,466,032	1,033,828,793	1,073,398,181	3,533,846,611
7	LWL-NPA	11,347,618,768	11,347,576,500	11,347,633,213	11,347,633,213
8	LWL-PA1	1,204,692,788	926,219,852	1,177,521,385	2,907,352,010
9	LWL-PA2	1,049,808,275	773,590,847	1,017,647,200	3,025,792,627
10	LPC-ME	68,602,499	61,099,697	67,149,536	298,481,616

TABLE 8. Statistical summary of QoS condition violation rate for the benchmarks.

Index	Benchmark	Statistics			
		Mean	Min	Median	Max
1	RR-NPA	0.00001267	0.00000000	0.00000000	0.00018940
2	RR-PA1	0.00001267	0.00000000	0.00000000	0.00018940
3	RR-PA2	0.00001285	0.00000000	0.00000000	0.00018939
4	JSQ-NPA	0.00000739	0.00000000	0.00000000	0.00010310
5	JSQ-PA1	0.00000739	0.00000000	0.00000000	0.00010310
6	JSQ-PA2	0.00000746	0.00000000	0.00000000	0.00010310
7	LWL-NPA	0.00000694	0.00000000	0.00000000	0.00010310
8	LWL-PA1	0.00000694	0.00000000	0.00000000	0.00010310
9	LWL-PA2	0.00000702	0.00000000	0.00000000	0.00010285
10	LPC-ME	0.00000843	0.00000000	0.00000000	0.00037892

REFERENCES

[1] M. Harchol-Balter, "Open problems in queueing theory inspired by datacenter computing," *Queueing Syst.*, vol. 97, nos. 1–2, pp. 3–37, Feb. 2021.

[2] C. Trueman, "Why data centres are the new frontier in the fight against climate change," *Computerworld*, 2019. Accessed: Mar. 1, 2022. [Online]. Available: https://www.computerworld.com/article/3431148/why-data-centres-are-the-new-frontier-in-the-fight-against-climate-change.html

[3] C. Tang, K. Yu, K. Veeraraghavan, J. Kaldor, S. Michelson, T. Kooburat, A. Anbudurai, M. Clark, K. Gogia, L. Cheng, B. Christensen, A. Gartrell, M. Khutornenko, S. Kulkarni, M. Pawlowski, T. Pelkonen, A. Rodrigues, R. Tibrewal, V. Venkatesan, and P. Zhang, "Twine: A unified cluster management system for shared infrastructure," in *Proc. 14th USENIX Symp. Oper. Syst. Design Implement. (OSDI)*, Nov. 2020, pp. 787–803. [Online]. Available: https://www.usenix.org/conference/osdi20/presentation/tang

[4] S. Bose and J. Kumar, "A survey on energy aware load balancing techniques in cloud computing," *Int. J. Adv. Res. Comput. Commun. Eng.*, vol. 4, no. 5, pp. 397–402, 2015.

[5] J. Zhang, F. R. Yu, S. Wang, T. Huang, Z. Liu, and Y. Liu, "Load balancing in data center networks: A survey," *IEEE Commun. Surveys Tuts.*, vol. 20, no. 3, pp. 2324–2352, 3rd Quart., 2018.

[6] S. A. Javadi and A. Gandhi, "User-centric interference-aware load balancing for cloud-deployed applications," *IEEE Trans. Cloud Comput.*, vol. 10, no. 1, pp. 736–748, Jan. 2022.

[7] V. Gupta, M. Harchol-Balter, K. Sigman, and W. Whitt, "Analysis of join-the-shortest-queue routing for web server farms," *Perform. Eval.*, vol. 64, nos. 9–12, pp. 1062–1081, 2007.

[8] V. Gupta and N. Walton, "Load balancing in the non-degenerate slowdown regime," 2017, *arXiv:1707.01969*.

[9] S. Mustafa, K. Bilal, S. U. R. Malik, and S. A. Madani, "SLA-aware energy efficient resource management for cloud environments," *IEEE Access*, vol. 6, pp. 15004–15020, 2018.

[10] T. L. Vasques, P. Moura, and A. de Almeida, "A review on energy efficiency and demand response with focus on small and medium data centers," *Energy Efficiency*, vol. 12, pp. 1399–1428, Jun. 2019.

[11] C. Jin, X. Bai, C. Yang, W. Mao, and X. Xu, "A review of power consumption models of servers in data centers," *Appl. Energy*, vol. 265, May 2020, Art. no. 114806.

[12] Y. M. Ko and Y. Cho, "A distributed speed scaling and load balancing algorithm for energy efficient data centers," *Perform. Eval.*, vol. 79, pp. 120–133, Sep. 2014.

[13] X. Ye, Y. Yin, and L. Lan, "Energy-efficient many-objective virtual machine placement optimization in a cloud computing environment," *IEEE Access*, vol. 5, pp. 16006–16020, 2017.

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

[15] G. Sun, Z. Xu, H. Yu, V. Chang, X. Du, and M. Guizani, "Toward SLAs guaranteed scalable VDC provisioning in cloud data centers," *IEEE Access*, vol. 7, pp. 80219–80232, 2019.

[16] S. Kwon, "QoS-aware data center operations based on chance- and risk-constrained optimization," *IEEE Trans. Cloud Comput.*, early access, Oct. 16, 2020, doi: 10.1109/TCC.2020.3031612.

[17] IBM Cloud Education. (Jun. 2019). *What is Virtualization*. Accessed: Mar. 1, 2022. [Online]. Available: https://www.ibm.com/cloud/learn/virtualization-a-complete-guide

[18] X. You, Y. Li, M. Zheng, C. Zhu, and L. Yu, "A survey and taxonomy of energy efficiency relevant surveys in cloud-related environments," *IEEE Access*, vol. 5, pp. 14066–14078, 2017.

[19] K. R. Remesh Babu and P. Samuel, "Service-level agreement–aware scheduling and load balancing of tasks in cloud," *Softw., Pract. Exper.*, vol. 49, no. 6, pp. 995–1012, Jun. 2019. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/spe.2692

[20] N. Buchbinder, Y. Fairstein, K. Mellou, I. Menache, and J. S. Naor, "Online virtual machine allocation with lifetime and load predictions," in *Proc. ACM SIGMETRICS/Int. Conf. Meas. Modeling Comput. Syst. (SIGMETRICS)*. New York, NY, USA: Association for Computing Machinery, 2021, pp. 9–10, doi: 10.1145/3410220.3456278.

[21] A. A. Khan, M. Zakarya, R. Buyya, R. Khan, M. Khan, and O. Rana, "An energy and performance aware consolidation technique for containerized datacenters," *IEEE Trans. Cloud Comput.*, vol. 9, no. 4, pp. 1305–1322, Oct./Dec. 2021.

[22] J. A. G. Arrubla, Y. M. Ko, R. J. Polansky, E. Pérez, L. Ntaimo, and N. Gautam, "Integrating virtualization, speed scaling, and powering on/off servers in data centers for energy efficiency," *IIE Trans.*, vol. 45, no. 10, pp. 1114–1136, Oct. 2013.

[23] Y. Cho and Y. M. Ko, "Energy efficiency of data center operating practices: Server clustering, powering on/off, and bang-bang control," *Networks*, vol. 71, no. 2, pp. 107–119, Mar. 2018.

[24] F. Bonomi, "On job assignment for a parallel system of processor sharing queues," *IEEE Trans. Comput.*, vol. 39, no. 7, pp. 858–869, Jul. 1990.

[25] N. Liu, Z. Dong, and R. Rojas-Cessa, "Task scheduling and server provisioning for energy-efficient cloud-computing data centers," in *Proc. IEEE 33rd Int. Conf. Distrib. Comput. Syst. Workshops*, Jul. 2013, pp. 226–231.

[26] A. Wierman, L. L. H. Andrew, and A. Tang, "Power-aware speed scaling in processor sharing systems: Optimality and robustness," *Perform. Eval.*, vol. 69, no. 12, pp. 601–622, Dec. 2012.

[27] W. Weng, X. Zhou, and R. Srikant, "Optimal load balancing in bipartite graphs," 2020, *arXiv:2008.08830*.

[28] S. Kwon and N. Gautam, "Time-stable performance in parallel queues with non-homogeneous and multi-class workloads," *IEEE/ACM Trans. Netw.*, vol. 24, no. 3, pp. 1322–1335, Jun. 2016.

[29] Y. Cho and Y. M. Ko, "Stabilizing the virtual response time in single-server processor sharing queues with slowly time-varying arrival rates," *Ann. Oper. Res.*, vol. 293, no. 1, pp. 27–55, Oct. 2020.

[30] E. Anton, U. Ayesta, M. Jonckheere, and I. M. Verloop, "Improving the performance of heterogeneous data centers through redundancy," *Proc. ACM Meas. Anal. Comput. Syst.*, vol. 4, no. 3, pp. 1–29, Nov. 2020, doi: 10.1145/3428333.

[31] T. Horvath, T. Abdelzaher, K. Skadron, and X. Liu, "Dynamic voltage scaling in multitier web servers with end-to-end delay control," *IEEE Trans. Comput.*, vol. 56, no. 4, pp. 444–458, Apr. 2007.

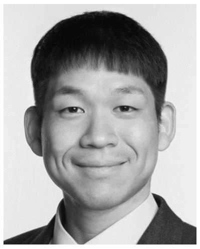
[32] Intel VTune Profiler User Guide. *P-State*. Accessed: Mar. 1, 2022. [Online]. Available: https://www.intel.com/content/www/us/en/develop/documentation/vtune-help/top/reference/energy-analysis-metrics-reference/p-state.html

[33] J. Bezanson, A. Edelman, S. Karpinski, and V. B. Shah, "Julia: A fresh approach to numerical computing," *SIAM Rev.*, vol. 59, no. 1, pp. 65–98, 2017, doi: 10.1137/141000671.

[34] D. Brodowski, N. Golde, R. J. Wysocki, and V. Kumark, *Linux CPUFreq Governors*. Accessed: Mar. 1, 2022. [Online]. Available: https://www.kernel.org/doc/Documentation/cpu-freq/governors.txt

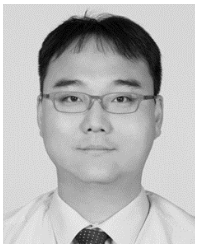
[35] U. Lublin and D. G. Feitelson, "The workload on parallel supercomputers: Modeling the characteristics of rigid jobs," *J. Parallel Distrib. Comput.*, vol. 63, no. 11, pp. 1105–1122, Nov. 2003, doi: 10.1016/S0743-7315(03)00108-4.

[36] I. Gerhardt and B. L. Nelson, "Transforming renewal processes for simulation of nonstationary arrival processes," *INFORMS J. Comput.*, vol. 21, no. 4, pp. 630–640, Nov. 2009.



YONGKYU CHO (Member, IEEE) received the B.S. degree in industrial engineering from Hanyang University, Ansan, South Korea, in 2013, and the Ph.D. degree in industrial and management engineering from the Pohang University of Science and Technology (POSTECH), Pohang, South Korea, in 2019.

In 2018, he was a Visiting Student with the Argonne National Laboratory (ANL), Mathematics and Computer Science Division, Lemont, IL, USA. From 2019 to 2021, he was a Senior Traffic Engineer at the Research and Development Team, Samsung Electronics Networks Business Division. Since 2021, he has been an Assistant Professor with the Department of Industrial and Management Engineering, Kangnam University, Yongin, South Korea. His research interests include optimal design and performance analysis of large-scale ICT infrastructures, such as energy-efficient resource management in data centers and the user plane protocols in 5G radio access networks.



YOUNG MYOUNG KO (Member, IEEE) received the B.S. and M.S. degrees in industrial engineering from Seoul National University, Seoul, South Korea, in 1998 and 2000, respectively, and the Ph.D. degree in industrial engineering from Texas A&M University, College Station, TX, USA, in 2011.

He serves as a Secretary of the INFORMS Telecommunications and Network Analytics Section. He is currently an Associate Professor with the Department of Industrial and Management Engineering, Pohang University of Science and Technology (POSTECH), South Korea. His research interests include analysis and optimization of large-scale stochastic systems, such as service systems, telecommunication networks, ICT infrastructure, and renewable energy systems. He is an Associate Editor of *IISE Transactions*.