Guaranteeing Performance based on Time-stability for Energy-efficient Data Centers

Soongeol Kwon, Natarajan Gautam
Department of Industrial and Systems Engineering,
Texas A&M University, College Station, TX, 77843-3131 USA
(e-mail: soongeol@tamu.edu, gautam@tamu.edu)

Abstract

We consider a system of multiple parallel single server queues where servers are heterogeneous with resources of different capacities, and could be powered on or off while running at different speeds when they are powered on. In addition we assume that application requests are heterogeneous with different workload distributions and resource requirements, and the arrival rates of request are time-varying. Managing such a heterogeneous, transient and non-stationary system is a tremendous challenge. We take an unconventional approach in that we force the queue lengths in each powered-on server to be time-stable (i.e. stationary). It allows the operators to guarantee performance and effectively monitor the system. We formulate a mixed integer program to minimize energy costs while satisfying time-stability. Simulation results show that our suggested approach can stabilize queue length distributions and provide probabilistic performance guarantees on waiting times. "Supplementary materials are available for this article. Go to the publisher's online edition of *IIE Transaction*, datasets, additional tables, detailed proofs, etc."

Keywords: Data center operation, resource management, time-stability, performance guarantees, energy efficiency, optimization and simulation

1 Introduction

We consider a data center which consists of a set of heterogeneous servers with different maximum processing speeds and different capacity limits of various resources, such as memory and storage. The data center hosts heterogeneous application classes which have different workload distributions, for which time-varying requests arrive with resource requirements and levels of quality of service (QoS) guarantees. In this case, servers process requests that belong to multiple classes,

whereas requests categorized into the same class are stochastically identical, and they arrive to data centers according to a piecewise constant non-homogeneous Poisson process. It is assumed that the environment process that drives arrival rates of the non-homogeneous Poisson process is cyclic. This is a fairly reasonable assumption as arrivals tend to have daily or weekly patterns that repeat in a cyclic fashion (Gmach et al. [16], Lin et al. [22], Liu et al. [26], Lin et al. [23] and Kwon and Gautam [21]). Using that assumption we model each cycle as divided into a set of intervals corresponding to the piecewise constant period for the arrival process. In addition, we assume that the number of classes is large (larger than the number of servers, e.g. 10 servers and 20 applications) but their workloads are so little that most servers host a mixture of heterogeneous classes. For the non-homogeneous arrival process, the arrival rate of each class is time-varying and also changes so fast that steady-state is not reached before arrival rates change. Based on the scenario mentioned above, we model a data center as a system of multiple parallel single server queues where a dispatcher routes arriving applications to servers.

For such a time-varying and heterogeneous system, our fundamental aim is to manage servers of data centers in an energy-efficient manner while satisfying performance guarantees by achieving time-stability. For time-varying and heterogeneous systems, arrival rates of requests change so fast that a steady-state of system is not reached, therefore making it extremely difficult to provide performance guarantees. Without assuming that system reaches steady-state, one can provide performance guarantees by powering on large number of servers responding to peak workload; however, that causes a huge additional energy cost than necessary. In this context, achieving time-stability is extremely useful for managing a non-homogeneous and transient system because it enables operators to effectively design and analyze the time-varying system, provides guaranteed QoS as desired, and effectively performs monitoring and control. Achieving time-stability and providing performance guarantees, while being mindful of energy costs, is the main objective of this study. To achieve time-stability, we consider the following control decisions that can be tuned.

- *Assignment*: Applications corresponding to each class of request can be assigned to servers such that each server hosts one or more classes and each class is hosted on multiple servers. We assume that there is no cost for the assignments as well as switching between assignments.
- *Sizing*: Each server could be dynamically powered on or off across time intervals. However,

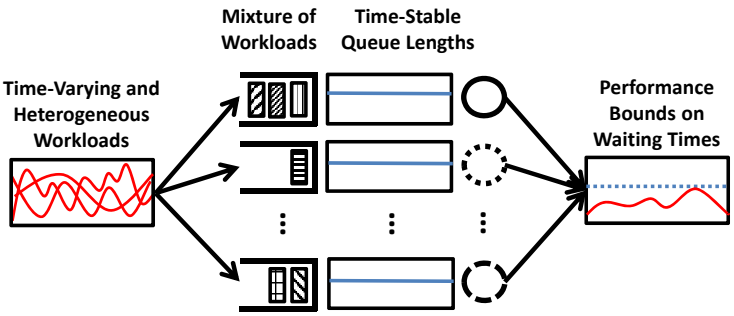


Figure 1: Achieving time-stability for multiple parallel single queues with time-varying and heterogeneous workloads

we neither consider switching costs (from on to off and vice versa) nor consider reliability costs for on-off cycles. Note that some modern servers allow for "sleep" settings instead of completely turning off servers. From a mathematical standpoint, we consider them equivalent.

- *Routing Fraction*: The dispatcher routes arriving requests to one of the powered-on servers based on predetermined routing fractions by considering assignment of classes to servers. A key assumption is that the dispatcher cannot observe the real-time state of any of the servers.
- *Speed Scaling*: For each server, it is possible to dynamically change the processing speed by scaling the voltage and frequency up to a maximum speed.

We will show that assignment, sizing, routing and speed scaling can be done appropriately in an integrated fashion to achieve both time-stability and energy-efficiency through the suggested framework described in Sections 2 and 3. Figure 1 briefly shows the scenario and the main objective of this study. Next we review the relevant literature and introduce notation used in this paper.

1.1 Literature Review

As there has been a surge in demand for cloud computing in recent years, server management in data centers has received tremendous attention in both industry and academia. Data centers provide benefits in cost reduction, flexibility, and accessibility by allowing enterprises to outsource resources for service rather than managing their own resources. As a result, data centers have challenging tasks to achieve energy efficiency and provide performance guarantees in cloud computing environments characterized by time-varying workloads with significant variation and uncertainties. In general, data centers need to provide strict QoS guarantees to users, thereby over-provisioning

their servers to respond to peak loads due to inherent uncertainty and variability in demand. On one hand this over-provisioning of servers results in low utilization as reported in Barroso and Hölzle [2] and Vogels [30]. On the other hand they incur a significant amount of energy usage for operating and cooling servers as explained and documented in Hamilton [18] and Koomey [19]. Fortunately, to abate the skyrocketing energy consumption in data centers (see report [5]), there are tremendous opportunities to conserve energy consumption in data centers, such as powering servers off and running them at slower speeds.

In this context, server provisioning plays a key role in improving utilization by selecting active servers (e.g. powering off servers or allowing to enter a power-saving mode) in accordance to traffic changes, while considering performance guarantees, and thus a number of techniques for efficient server provisioning have been proposed to address the above problem. Gandhi et al. [11] presented an approach based on a combination of predictive and reactive provisioning to correctly allocate resources in data centers such that service level agreement (SLA) violations and energy consumption are minimized. Zhu et al. [35] presented a data center architectural design based on virtualized resources in order to reduce provisioning overhead; they also proposed a dynamic provisioning technique while satisfying user's SLA and maximizing overall profits. Also, Wang et al. [31] provided an analytic framework that captures non-stationarities and stochastic variation of workloads for dynamic re-sizing in data centers. Lin et al. [23] suggested a new on-line algorithm for dynamic right sizing in data centers motivated by optimal off-line solutions to minimize energy costs including switch costs. In addition, there have been some studies that considered heterogeneous workloads for server provisioning and allocation problems. There is another body of literature that proposed an optimization approach based on powering servers on/off and dynamic voltage/frequency scaling (DVFS) to minimize energy consumption. Bertini et al. [3] proposed a mixed integer program (MIP) for the problem of selecting the servers' states and processing speeds with QoS control, and Gallego Arrubla et al. [10] introduced a unified methodology that combines virtualization, speed scaling, and powering off servers to efficiently operate data centers while incorporating the inherent variability and uncertainty.

While all of the aforementioned research is complementary to our work, the key difference is that our study suggests an approach which aims to not only save energy consumption but also achieve time-stability for providing performance guarantees. Note that, to the best of our

knowledge, the problem of achieving time-stability over time-varying traffic in data center operation has received little attention and not been effectively addressed. Although time-stability has received little attention in the context of data centers operation, there have been some research studies in the queueing area. Foley et al. [9] and Barnes et al. [1] showed that the departure process from the $M_t/G_t/\infty$ queue can be made stationary, and in recent days, Whitt [32] suggested the rate matching control algorithm, which stabilizes the queue length distribution for $G_t/G_t/1$ single-server queue where both arrival rate and service rate are time-varying. There is another body of literature which provides algorithms to determine appropriate staffing levels for call centers. Feldman et al. [8] proposed a simulation-based iterative-staffing algorithm for time-stable delay probability, and Liu and Whitt [24] suggested a formula-based algorithm to stabilize abandonment probabilities and expected delays using offered-load based approximations for a queueing model with the non-homogeneous Poisson arrival process and customer abandonment.

In fact, our previous work [21] suggests an approach to achieve time-stability in both queue lengths and sojourn times for data center operations where time-varying arrivals are cyclic. In [21], we considered a scenario of a company that owns a data center (e.g. Yahoo, Google or Facebook) and operates a large number of servers to host applications. We assumed that each application request has high squared coefficient of variation (SCOV) of workloads. Also, each application needs to be hosted on a large number of servers to handle the load (in fact, we use an asymptotic scaling, the number of servers $N_a \to \infty$ for each application a, which allowed us to attain time-stability). In comparison, this study considers a unique scenario for hosting data centers that provide hosting service to several other companies. In general, hosting data centers cluster applications of each company and assign the cluster to a group of servers for the purpose of security and confidentiality. It is important for the data centers to monitor the performance experienced by the applications, and without time-stable performance, monitoring would be difficult. This motivation is to consider time-stability. We assume that the number of application classes is much larger than the number of servers and many application classes have so little load that they would need to be hosted on just one server. Also, we assume that servers are heterogeneous and host a mixture of heterogeneous application requests, and processing speed of a server can be changed for energy efficiency (in [21], servers are homogeneous and every server runs at the same processing speed). Moreover, in this study, we formulate an MIP problem to optimally determine operational decisions including

assignment, routing, and speed scaling for time-stability as well as energy efficiency, which were not considered in [21]. Therefore, our suggested model in this study is fundamentally different from the problem considered in [21].

For the purpose of this study, we provide an analytical framework which decomposes a complex and non-stationary system into individual simpler stationary ones based on multiple strategies for assignment, sizing, routing, and speed scaling. Based on the suggested framework, our objective is to stabilize queue length distributions of each powered-on server and provide performance guarantees on waiting time of each application class. Moreover, for energy efficiency we propose an optimization model to minimize total energy cost via powering on or off servers, routing, and speed scaling, while satisfying the time-homogeneity constraints. In fact, our suggested approach enables us to utilize standard stationary queueing analysis to obtain performance guarantees for time-varying, transient, and heterogeneous systems, and we believe that our study has significance and provides useful insights for practitioners. The main contributions of this study are to: (i) provide an integrated framework unifying sizing, assignment, routing, and speed scaling under heterogeneous conditions which has seldom been implemented jointly; (ii) define time-homogeneity constraints which ensures time-stability; (iii) suggest an approach to provide performance guarantees based on time-stability; and (iv) introduce an optimization problem with an MIP formulation to reduce energy cost while considering time-stability. The remainder of the paper is organized as follows: Section 2 introduces our notion of time-stability and suggests an approach to obtain time-homogeneity constraints; Section 3 proposes an optimization problem to determine various decisions for energy conservation and time-stability; Section 4 proposes an approach to provide performance guarantees; Section 5 reports and analyzes the results of numerical experiments; and Section 6 presents conclusions and future research directions.

1.2 Notations

For the scenario considered in this study, we set the notations to define the problem and describe our approach appropriately. An arriving request belongs to one of multiple classes in a discrete set $\mathcal{A}$. We categorize incoming requests into several class types based on their mean workload and each class has a small squared coefficient of variance. The amount of work a class $a \in \mathcal{A}$ request brings is independent and identically distributed (IID) according to a general distribution $H_a(\cdot)$ with mean

Table 1: Indices, parameters, and decision variables of suggested optimization problem

Indices	
$\mathcal{T}$	index set of time intervals $\ell \in \mathcal{T}$
$\mathcal{A}$	index set of classes of requests $a \in \mathcal{A}$
$\mathcal{N}$	index set of servers $j \in \mathcal{N}$
$\mathcal{K}$	index set of types of resources $k \in \mathcal{K}$
Parameters	
$\lambda_{a\ell}$	arrival rate of class a in time interval ℓ
$1/\theta_a$	average amount of workload brought by class a
ρ	desirable traffic intensity for each powered-on server
β_a^k	requirement of resource k for class a
b_j^k	capacity limit of resource k for server j
ϕ_{max}^j	maximum processing speed of server j
Decision variables	
$x_{aj\ell}$	assignment of class a to server j in time interval ℓ
$y_{j\ell}$	whether server j must be powered-on or off in time interval ℓ
$v_{aj\ell}$	fraction of class a to route to server j in time interval ℓ
$\phi_{j\ell}$	processing speed of server j in time interval ℓ

$1/\theta_a$ and SCOV C_a^2. Also, each class a has requirements β_a^k for each resource type $k \in \mathcal{K}$ (this could be memory or storage, for example). Let $\mathcal{T}$ be a collection of contiguous time intervals and in each interval $\ell \in \mathcal{T}$ the arrival rate for each class $a \in \mathcal{A}$ remains a constant $\lambda_{a\ell}$. Also, we consider $\mathcal{N}$ heterogeneous servers and each server $j \in \mathcal{N}$ has maximum processing speed ϕ_{max}^j and capacity limit for each resource type $k \in \mathcal{K}$, b_j^k. Note that instead of defining service time distribution, which is generally used in other studies based on queueing models, our approach uses the combination of workload distribution and processing speed (which could be varying). In addition, we define the desired traffic intensity ρ for each powered-on server. In fact, we use the desired traffic intensity to create enough residual capacity for unforeseen surges by restricting the utilization of each server to be ρ. Next, for time interval $\ell \in \mathcal{T}$, server $j \in \mathcal{N}$, class $a \in \mathcal{A}$, and resource type $k \in \mathcal{K}$ we define decision variables considered in this study as follows: (i) the assignment of class a to servers j for each time interval ℓ, $x_{aj\ell}$ (e.g. $x_{aj\ell} = 1$ if class a assigned to server j in time interval ℓ, otherwise $x_{aj\ell} = 0$.), (ii) whether server j must be powered on or off in time interval ℓ, $y_{j\ell}$ (e.g. $y_{j\ell} = 1$ if server j is powered on in time interval ℓ, otherwise, $y_{j\ell} = 0$), (iii) the processing speed for server j in time interval ℓ, $\phi_{j\ell}$, and (iv) the fraction of jobs of application a to be routed to server j in time interval ℓ, $v_{aj\ell}$. Each server j must be powered on or off in each time interval ℓ, and thus there will be $N_\ell = \sum_{j\in\mathcal{N}} y_{j\ell}$ powered-on servers in each time interval ℓ. We summarize the set of indices, parameters, and decision variables which are used to define optimization problem in Table 1.

2 Time-stability

Recall that our main objective is to provide performance guarantees for time-varying and heterogeneous data centers by achieving time-stability. In this section, first we introduce the notion of time-stability considered in this study and then we suggest an approach to achieve time-stability. We conclude this section by introducing a practical application of a suggested approach considered in this study. Recall that the benefit of time-stability has been discussed in Section 1.

2.1 Our Notion of Time-stability

As described in Section 1, we consider a scenario where each server may host multiple classes of applications with aggregate workload defined by a mixture of heterogeneous application classes. For achieving time-stability, we seek to manage the system so that any powered-on server receives arrivals with a target workload distribution $H(\cdot)$ whose Laplace-Stieltjes transform (LST) is $\tilde{H}(s) = \int_0^\infty e^{-sx} dH(x)$. We denote $1/\theta$ as the target average amount of work brought by each arrival. We seek to keep the aggregate workload distribution at any instant of time in each powered-on server to be time-stable in order to ensure that the distribution of queue lengths at any time for every powered-on server is according to a stationary distribution of a homogeneous $M/G/1$ queue. In Section 2.2, we will propose an approach to obtain time-stable queue length distribution by stabilizing both the aggregate workload distribution and arrival rates, and also in Section 3 we will show that time-stability would be valid for speed scaling. For mathematical representation, for all $j \in \mathcal{N}$, at time t let $X_j(t)$ be the number of requests in queue of server j and $O_j(t)$ be the status of the server ($O_j(t) = 1$ if server j is powered-on at time t, otherwise $O_j(t) = 0$). Our approach indeed stabilizes queue length distribution $\pi(i)$ for all $i \geq 0$ which can be represented as $P\{X_j(t) = i | O_j(t) = 1\} = \pi(i)$ where $\pi(i)$ is constant and not dependent on t. Based on the time-stability in queue length distribution, performance analysis is straightforward and probabilistic guarantees on the waiting times can be provided.

2.2 Approach to Achieve Time-stability

In this section we suggest an approach to achieve time-stability introduced in Section 2.1. For time-varying arrivals of requests of heterogeneous applications, the main idea of our suggested approach is to stabilize queueing process of each powered-on server based on (i) time-invariant aggregate workload distribution obtained by moment matching approximation and (ii) rate matching between

arrival rates and processing speeds by selecting routing fractions appropriately. In the following subsections, we will derive time-homogeneity constraints for the proposed MIP problem.

2.2.1 Moment Matching Approximation for Time-invariant Workload Distribution

For achieving time-stability, we seek to stabilize an aggregate workload distribution of each powered-on server where the servers host incoming requests of multiple heterogeneous applications that have different workload distributions $H_a(\cdot)$. To achieve the invariant target workload distribution $H(\cdot)$ with LST $\tilde{H}(s)$ and average workload $1/\theta$, our objective is to control the system appropriately. This way each powered-on server receives arrivals of requests with homogeneous aggregate workload distribution $H(\cdot)$ based on moment matching approximation. To begin, for the desirable traffic intensity ρ and the target nominal speed ϕ, we determine the minimum number of powered-on servers for each time interval $\ell \in \mathcal{T}$, N_ℓ, which satisfies the following inequality

$$\sum_{a\in\mathcal{A}} \lambda_{a\ell}\tilde{H}_a(s) \leq N_\ell \tilde{H}(s)\rho\phi\theta \quad \forall s \geq 0 \tag{1}$$

where the left-hand side is the LST of aggregate workload distribution for the entire system and the right-hand side represents the LST of aggregate workload distribution of N_ℓ time-stable servers, where each of the N_ℓ servers has the target workload distribution $H(\cdot)$ and arrival rate as $\rho\phi\theta$. Based on the inequality (1), N_ℓ can be determined so that the workload served by each of the N_ℓ servers would be less than equal to the target workload (which is defined by traffic intensity ρ, processing speed ϕ, and mean workload $1/\theta$) when the incoming workload is equally distributed with the same routing fractions to the powered-on servers. In fact, we can redefine inequality (1) based on moment matching approximation for several moments as follows:

$$\sum_{a\in\mathcal{A}} \frac{\lambda_{a\ell}}{\theta_a} \leq N_\ell\rho\phi$$

$$(-\tilde{H}'(0)) \sum_{a\in\mathcal{A}} \lambda_{a\ell} \leq N_\ell\rho\phi$$

$$(-\tilde{H}'(0)) \sum_{a\in\mathcal{A}} \lambda_{a\ell}\tilde{H}''_a(0) \leq N_\ell\rho\phi\tilde{H}''(0)$$

$$(-\tilde{H}'(0)) \sum_{a\in\mathcal{A}} \lambda_{a\ell}(-\tilde{H}'''_a(0)) \leq N_\ell\rho\phi(-\tilde{H}'''(0))$$

$$\vdots$$

by taking the derivatives of the LST $\tilde{H}(s)$ at $s = 0$. Our aim is to introduce additional traffic to match the difference between the left-hand side and the right-hand side of each inequality (each moment) so that the aggregate workload distribution of each of the N_ℓ servers is approximately equal to the target workload distribution $H(\cdot)$. As discussed in [33], [34], and [25], additional traffic can be thought of as low priority jobs (or delay-tolerant jobs) that do not have time constraints that data centers need to process. For additional traffic with index 0, arrival rate in time interval ℓ, $\lambda_{0\ell}$, and the moment of workload distribution for each time interval ℓ, $-\tilde{H'}_{0\ell}(0)$, $\tilde{H''}_{0\ell}(0)$, $-\tilde{H'''}_{0\ell}(0)$, ..., can be determined appropriately through the following equalities based on N_ℓ:

$$\begin{aligned}
(-\tilde{H'}(0))\left(\lambda_{0\ell} + \sum_{a\in\mathcal{A}} \lambda_{a\ell}\right) &= N_\ell \rho\phi \\
\frac{\lambda_{0\ell}}{\theta_{0\ell}} + \sum_{a\in\mathcal{A}} \frac{\lambda_{a\ell}}{\theta_a} &= N_\ell \rho\phi \\
(-\tilde{H'}(0))\left(\lambda_{0\ell}\tilde{H''}_{0\ell}(0) + \sum_{a\in\mathcal{A}} \lambda_{a\ell}\tilde{H''}_a(0)\right) &= N_\ell \rho\phi\tilde{H''}(0) \\
(-\tilde{H'}(0))\left(-\lambda_{0\ell}\tilde{H'''}_{0\ell}(0) + \sum_{a\in\mathcal{A}} \lambda_{a\ell}(-\tilde{H'''}_a(0))\right) &= N_\ell \rho\phi(-\tilde{H'''}(0)) \\
&\vdots
\end{aligned}$$

and then we can approximately define $H_{0\ell}(\cdot)$ by combining the derivatives. Recall that there is additional traffic which has low priority of QoS guarantees (e.g. non-interactive jobs are less sensitive to response time) and is also CPU intensive (it does not need other resource requirements). In this case, it is reasonable to assume that this additional traffic can be assigned to the system according to a Poisson process with parameter $\lambda_{0\ell}$ and workload distribution $H_{0\ell}(\cdot)$ for each time interval ℓ from front-end proxy servers or other data centers. We would like to note that our suggested approach allows additional traffic so that the increment of workloads caused by an addition of traffic would not degrade the desired performance, but would reduce variability and uncertainty in aggregate workload while also improving the utilization of CPU for each powered-on server. Note that if we match sufficiently many moments, then the above equalities would result in

$$\lambda_{0\ell}\tilde{H}_{0\ell}(s) + \sum_{a\in\mathcal{A}} \lambda_{a\ell}\tilde{H}_a(s) = N_\ell \tilde{H}(s)\rho\phi\theta \quad \forall s \geq 0 \tag{2}$$

where $H_{0\ell}(\cdot)$ has $-\tilde{H}'(0)$, $\tilde{H}''(0)$, $-\tilde{H}'''(0)$, ... as moments. By selecting additional traffic for several moments, our suggested approach stabilizes workload distribution for each powered-on server.

2.2.2 Selecting Routing Fractions for Time-stable Arrival Rates

In addition to stabilizing aggregate workload distribution as introduced in Section 2.2.1, next our suggested approach stabilizes an aggregate arrival rate to each powered-on server so that the queue length distribution of each powered-on server is a stationary time-homogeneous $M/G/1$ queue. Based on the minimum number of powered-on servers N_ℓ, arrival rates $\lambda_{0\ell}$ and workload distributions $H_{0\ell}(\cdot)$ of additional traffic for each time interval $\ell \in \mathcal{T}$, we seek to route arrivals of requests so that arrival rates into each powered-on server would be time-homogeneous (i.e. constant across time-intervals) while ensuring the aggregate workload distribution is also stabilized as the target workload distribution $H(\cdot)$. Let $v_{aj\ell}$ be the fraction of arriving requests of class a routed to server j in time interval ℓ, then $v_{aj\ell}$ can be appropriately selected based on the following time-homogeneity constraints:

$$\sum_{j=1}^{N_\ell} v_{aj\ell} = 1 \quad \forall a \in \mathcal{A} \cup 0 \tag{3}$$

$$\frac{\lambda_{0\ell}}{\theta_{0\ell}} v_{0j\ell} + \sum_{a \in \mathcal{A}} \frac{\lambda a\ell}{\theta_a} v_{aj\ell} = \rho\phi \quad \forall j \in \{1, 2, ..., N_\ell\} \tag{4}$$

$$\frac{\lambda_{0\ell} v_{0j\ell} \tilde{H}_{0\ell}(s) + \sum_{a \in \mathcal{A}} \lambda_{a\ell} v_{aj\ell} \tilde{H}_a(s)}{\sum_{a \in \mathcal{A} \cup 0} \lambda_{a\ell} v_{aj\ell}} = \tilde{H}(s) \quad \forall j \in \{1, 2, ..., N_\ell\}, s \geq 0. \tag{5}$$

Recall that ρ is the desired traffic intensity and ϕ is the target nominal speed. Equation (3) means that all application requests should be routed to servers; traffic intensity of each powered-on server would be ρ based on Equation (4); Equation (5) ensures that aggregate workload distribution at each powered-on server j is approximately equivalent to the target workload distribution $H(\cdot)$. In fact Equation (5) is directly derived from Equation (2) by applying $v_{aj\ell}$ to routing arriving requests instead of distributing equally to N_ℓ powered-on servers. Based on Equations (4) and (5), arrival rates into each powered-on server would remain constant at $\rho\phi\theta$ across time intervals, since the mean amount of workload brought by incoming request would be $1/\theta$ based on Equation (5) (i.e. $-\tilde{H}'(0) = 1/\theta$) and thus,

$$\sum_{a \in \mathcal{A} \cup 0} \frac{\lambda_{a\ell}}{\theta} v_{aj\ell} = \rho\phi, \quad \sum_{a \in \mathcal{A} \cup 0} \lambda_{a\ell} v_{aj\ell} = \rho\phi\theta \quad \forall j \in \{1, 2, ..., N_\ell\}.$$

We will formulate an MIP problem by using time-homogeneity constraints (3), (4), and (5) to optimally determine $v_{aj\ell}$ to achieve time-stability in Section 3.

2.2.3 Adjusting Initial Condition of Time Intervals

The fundamental idea for reducing energy cost is to power servers on and off as they respond to time-varying workloads. To that end, we will formulate an MIP problem in Section 3 to determine operational decisions (including powering servers on and off) so that the queue length distribution of every powered-on server is stabilized across time intervals. In this case, it is intuitive to think that time-stability (stabilized queue length distribution) would be affected when servers are powered-on afresh (from powered-off state) since newly powered-on servers start processing jobs with an empty queue; however in the other case, servers process jobs with a stationary queue. Thus, queue length distribution would not be stabilized when the length of each time interval is not long enough to reach steady-state. To address this problem, we add a batch of requests to servers which are powered-on afresh at the beginning of a time interval so that queue lengths of every powered-on server would be stochastically equivalent to that of a stationary $M/G/1$ queue. We adopt an approach proposed by our previous work [21] whose main idea is adding a number of jobs sampled from the stationary queue length distribution defined by the stabilized workload distribution ($H(\cdot)$) and arrival rates ($\rho\phi\theta$). If servers selected to be powered-off are not idle at the end of time interval, then incoming requests would not be routed to those servers, and we wait until those servers complete service for the remaining requests (and then power off the servers). As we discussed in [21], the key benefit of the suggested approach is that performance bounds based on time-stability are provable and also easily derived by using standard queueing theory results. In addition, for the general case, we have the following remark for an extension of our suggested approach.

Remark 1 (Extension of Time-stability to Processor Sharing) *In Section 2.1, we introduce our notion of time-stability based on stationary $M/G/1$ queues where each server hosts a mixture of multiple heterogeneous applications with first-come first-served (FCFS) service discipline. Here our claim is that our suggested approach can be extended to a model using processor sharing (PS) service policy. Under PS regime, all the jobs or entities in the system are served simultaneously and equally share the processor at any given time; it is also fairly common to model computer servers based on PS service policy. In fact, $M/G/1$ queue with PS does produce closed form results which*

are identical to those of $M/M/1$ queues with FCFS discipline [15]. Thus, our suggested approach for time-stability can be applied to the model which uses PS instead of FCFS for service policy. For PS service policy, the mean queue length L of $M/G/1$ would be $L = \rho/(1-\rho)$, and thus the mean sojourn time would be $1/(\theta\phi(1-\rho))$.

2.3 Practical Application of the Suggested Approach

In Section 2.2, we introduce the notion of time-stability considered in this study and suggest an approach to achieve time-stability based on moment matching approximation. As we mentioned, if we match sufficiently many moments, then aggregate workload distribution at every powered-on server would be approximately equivalent to $H(\cdot)$, and thus we could theoretically have stabilized queue length distribution. Moreover, for the practical application, we can choose to match only a few moments to achieve time-stability. In this section, we will show that the mean queue lengths of every powered-on server can be reasonably stabilized across time intervals by considering only first and second moments in practice. Recall that our aim is to manage each powered-on server as a stationary $M/G/1$ queue, and we use the Pollaczek-Khintchine (P-K) formula [15] to determine the mean queue length L as follows:

$$L = \rho + \frac{\rho^2(1+C^2)}{2(1-\rho)} \tag{6}$$

where C^2 is the SCOV of service time and ρ is the traffic intensity. Based on our suggested approach, we need to determine the target workload distribution $H(\cdot)$, and in fact, $H(\cdot)$ can be determined according to the desired mean queue length. For example, let $\bar{L}$ be the desired (i.e. targeted) mean queue length; then it is possible to select C^2 of $H(\cdot)$ based on Equation (6) for given $\bar{L}$ and ρ (recall that ρ is the desired traffic intensity). Since C^2 is solely determined by the first and second moments of $H(\cdot)$ ($-\tilde{H}'(0)$ and $\tilde{H}''(0)$, respectively), the mean queue length can be stabilized as desired by determining the first and second moments of $H(\cdot)$ and matching the first and second moments as follows:

$$\frac{\lambda_{0\ell}v_{0j\ell}(-\tilde{H}'_{0\ell}(0)) + \sum_{a\in\mathcal{A}}\lambda_{a\ell}v_{aj\ell}(-\tilde{H}'_{a}(0))}{\sum_{a\in\mathcal{A}\cup 0}\lambda_{a\ell}v_{aj\ell}} = -\tilde{H}'(0) \quad \forall j \in \{1,2,...,N_\ell\}, s \geq 0 \tag{7}$$

$$\frac{\lambda_{0\ell}v_{0j\ell}\tilde{H}''_{0\ell}(0) + \sum_{a\in\mathcal{A}}\lambda_{a\ell}v_{aj\ell}\tilde{H}''_{a}(0)}{\sum_{a\in\mathcal{A}\cup 0}\lambda_{a\ell}v_{aj\ell}} = \tilde{H}''(0) \quad \forall j \in \{1,2,...,N_\ell\}, s \geq 0. \tag{8}$$

In Section 3, we will formulate an MIP by using the constraints (7) and (8) instead of using (5) to stabilize the mean queue length and provide performance guarantees on the mean waiting time.

3 Optimization Problem

In this section, we suggest an optimization problem to determine operational decisions that correspond to decision variables $x_{aj\ell}$, $y_{j\ell}$, $v_{aj\ell}$, and $\phi_{j\ell}$ introduced in Section 1.2. The key idea is to select decision variables, $x_{aj\ell}$, $y_{j\ell}$, $v_{aj\ell}$, and $\phi_{j\ell}$ so that it not only enforces time-stability of the queue length distribution at every powered-on server but also results in the system being energy-efficient.

3.1 Assumption for Energy Cost

For considering the energy cost of data center operation, we define a fixed energy cost f_j for powered-on server j (i.e. energy cost for server j at "idle" state) and an operating cost c_j for processing jobs at server j at speed $\phi_{j\ell}$. In fact, we are assuming that there is no cost and no service delay for switching server operations between time intervals. Next we briefly describe the reason we assume there would be no switching cost for our suggested problem. According to [22] and [23], the switching cost mainly consists of (i) energy used for powering servers on and off, (ii) a time delay that the server is in setup, and (iii) increased wear-and-tear on the server toggling. For (i) and (ii), firstly, we would like to note that the length of time intervals used in our suggested model is much longer than the duration of setup time. For example, authors in [12], [14], and [17] reported that setup times are ranging from 20 seconds to 200 seconds, which are very small compared to the one hour (3600 seconds) time interval length. In this case, the power consumed during setup can be negligible compared with the power consumption for running a server during a one-hour time interval, and thus, it is reasonable to only consider operating cost without considering extra setup cost. Also our suggested approach is a static control algorithm, and server schedules (e.g. $y_{j\ell}$ and $\phi_{j\ell}$ for all $j \in \mathcal{N}$ and $\ell \in \mathcal{T}$) are pre-determined, as opposed to dynamic server provisioning considered in [22] and [23], which determines whether to put idle servers to sleep or wake up servers in real-time. Thus, based on the scenario considered in our problem, servers can be set up to process jobs in advance based on the pre-determined server schedules, and thus service delay also can be negligible. In terms of (iii), a body of literature (e.g. [7], [17], [22], [23], and [31]) considers the impact of server on/off cycles on the reliability of the server. For example, the recent studies [22], [23], and [31] proposed an approach for the dynamic server provisioning model by

considering switching cost that includes server reliability cost for wear-and-tear caused by toggling servers. However, contrasting the dynamic server provisioning algorithms proposed by [22], [23], and [31], which change server state frequently, the servers in our problem would stay at powered-on or powered-off (or sleep) state for at least one hour and would not change server state frequently; thus the effect of toggling servers on reliability may not be significant. Moreover, as mentioned in [6] and [4], powering servers off may extend the lifetime of server components. Therefore, toggling servers may not significantly affect reliability of servers in our proposed problem. In addition, since the assignment of classes to servers are changing across time intervals in our suggested problem, one may argue that there may exist a cost for switching assignments. We would like to mention that cost for switching assignments can be easily ignored since all applications essentially reside in all servers and in a given time interval, the applications that are used are fired up while others are off. Also, it is reasonable to assume that the assignment of applications to servers can be changed without service delay since applications can be deployed on servers concurrently while they process other jobs. Based on the above description, we consider only fixed energy cost and operating cost without including server switching cost in our suggested problem.

3.2 Formulation

For time-stability, the target aggregate workload distribution $H(\cdot)$, and arrival rates of additional traffic $\lambda_{0\ell}$ and workload distribution $H_0(\cdot)$ are determined as described in Section 2. Recall that we define time-homogeneity constraints to stabilize the mean queue lengths by using constraints (7) and (8) based on the first and second moments of workload distribution of additional traffic (indexed as "0") $1/\theta_{0\ell}$ and $\tilde{H''_{0\ell}}(0)$. Based on the set of indices, parameters, and decision variables, we formulate an MIP optimization problem. For each time interval $\ell \in \mathcal{T}$, our suggested optimization problem can be formulated as follows:

$$\text{MIP-}\ell\text{: Minimize} \quad \sum_{j\in\mathcal{N}} (f_j y_{j\ell} + c_j \phi_{j\ell}) \tag{9}$$

$$\text{s.t.} \quad \sum_{j\in\mathcal{N}} v_{aj\ell} = 1 \qquad \forall a \in \mathcal{A} \tag{10}$$

$$\frac{\lambda_{0\ell}}{\theta_{0\ell}} v_{0j\ell} + \sum_{a\in\mathcal{A}} \frac{\lambda_{a\ell}}{\theta_a} v_{aj\ell} = \rho\phi_{j\ell} \qquad \forall j \in \mathcal{N} \tag{11}$$

$$\phi_{j\ell} \le \phi^j_{max} y_{j\ell} \qquad \forall j \in \mathcal{N} \tag{12}$$

$$\frac{\lambda_{0\ell}v_{0j\ell}(-\tilde{H}'_{0\ell}(0)) + \sum_{a\in\mathcal{A}}\lambda_{a\ell}v_{aj\ell}(-\tilde{H}'_{a}(0))}{\sum_{a\in\mathcal{A}\cup 0}\lambda_{a\ell}v_{aj\ell}} = -\tilde{H}'(0) \quad \forall j \in \mathcal{N} \tag{13}$$

$$\frac{\lambda_{0\ell}v_{0j\ell}\tilde{H}''_{0\ell}(0) + \sum_{a\in\mathcal{A}}\lambda_{a\ell}v_{aj\ell}\tilde{H}''_{a}(0)}{\sum_{a\in\mathcal{A}\cup 0}\lambda_{a\ell}v_{aj\ell}} = \tilde{H}''(0) \quad \forall j \in \mathcal{N} \tag{14}$$

$$\sum_{a\in\mathcal{A}} \beta_a^k x_{aj\ell} \le b_j^k \qquad \forall j \in \mathcal{N}, \forall k \in \mathcal{K} \tag{15}$$

$$v_{aj\ell} \le x_{aj\ell} \qquad \forall a \in \mathcal{A}, \forall j \in \mathcal{N} \tag{16}$$

$$v_{aj\ell} \ge 0, x_{aj\ell} \in \{0,1\}, y_{j\ell} \in \{0,1\}, \phi_{j\ell} \ge 0 \qquad \forall a \in \mathcal{A}, \forall j \in \mathcal{N}. \tag{17}$$

In the above formulation, the objective function (9) is total energy cost in a cycle, which consists of fixed cost and operating cost that we wish to minimize. We define the cost function for energy usage as a combination of fixed energy cost and operating cost using a model of power usage of typical servers adopted by [23]. For operating cost, we would like to note that modern CPUs can be operated at different speeds during runtime by employing DVFS, and recent studies Gandhi et al. [13], Kusic et al. [20], and Raghavendra et al. [27] reported that DVFS results in a linear power and frequency relationship without additional cost for ramp up/down processing speed as defined in our objective function (9). Constraint (10) ensures that all class a traffic is divided across various servers, and constraint (11) ensures that the traffic intensity for server j be ρ since the average work that arrives at server j is $\rho\phi_{j\ell}$ if the server runs at speed $\phi_{j\ell}$. Constraint (12) forces the server j's speed to be zero when it is off and limits its speed by its maximum value when on. Constraints (13) and (14) match the first and second moments, respectively, to stabilize the mean queue length as described in Section 2.3. Constraints (10), (11), (13), and (14) are derived from the homogeneity constraints defined in Section 2.2.2. Recall that we define constraint (4) with constant speed ϕ, but here we define constraint (11) to allow speed scaling with $\phi_{j\ell}$ for minimizing energy cost based on theoretical foundations described in the online supplement. In fact, Theorem 1 in the online supplement shows that the queue length distribution of time-varying $M_t/G_t/1$ queue is stochastically identical to that of time-homogeneous $M/G/1$ queue when we adjust processing speed (service rate) responding to time-varying arrival rate based on constraint (4). In addition, constraint (15) limits resource k to b_j^k across classes assigned to server j with requirements β_a^k for class a, and constraint (16) ensures that if class a is not assigned to server j, then no fraction of arriving requests are assigned to that server. Constraint (17) ensures non-negativity and binary

nature of the decision variables. The decision variables $x_{aj\ell}$, $y_{j\ell}$, $\phi_{j\ell}$, and $v_{aj\ell}$ (for all $\ell \in \mathcal{T}$, $j \in \mathcal{N}$, and $a \in \mathcal{A}$) can be determined by solving the above MIP-ℓ problem separately for each time interval $\ell \in \mathcal{T}$. Although we solve the optimization problem to determine operational decisions independently for each time interval $\ell \in \mathcal{T}$ (e.g. powering server on/off, assignment, routing, and speed scaling), the aggregate workload distribution and the queue length distribution of every powered-on server would be stabilized across time intervals. In other words, based on our suggested approach, we can simplify a large scale, transient, non-stationary problem by decomposing it into individual smaller problems (i.e. decompose overall problem into 24 MIP-ℓ problems) in order to achieve time-stability. This is the key benefit of our suggested approach since we can solve smaller problems instead of large-scale MIP problems while stabilizing the queue length distribution across time intervals. Although we decompose the overall problem into individual MIP-ℓ problems, each of the MIP-ℓ problems is indeed NP-hard (we can simply show that a well-known NP-hard capacitated facility location problem (CFLP) can be reduced to our suggested MIP-ℓ problem), and it is difficult to solve the problem for large-size instances; therefore, we need to develop an efficient algorithm to solve the proposed problem. Note that in this study we choose to focus on introducing the key idea of our approach to achieve time-stability and analyze numerical results for the smaller instance, and developing an algorithm to solve the suggested MIP problem is beyond the scope of this study. Based on our suggested approach, we will study approaches to solve large-scale problems in the future.

4 Performance Guarantee

Through Sections 2 and 3, we introduce the notion of time-stability and suggest an approach to stabilize the queue length distribution of each powered-on server. Specifically, we derive the constraints and formulate an MIP problem to stabilize the mean queue lengths. Moreover, in this section, we will show that performance bound on waiting time can be obtained based on the stabilized queue length distribution. In general, users of data centers commonly require QoS guarantees in terms of waiting times (or response times), or for that matter SLA and its violation are also defined based on waiting times. Thus, performance bound on waiting time would be a preferred performance measure and much more useful to design, monitor, and control data center operations. Recall that, for energy efficiency, we apply speed scaling techniques and show that

the queue length distribution would be stabilized when the processing speed of each powered-on server changes in proportion to arrival rates in the online supplement. In this case, even if the queue length distribution is stabilized, waiting time distribution would not be stabilized since the processing speed is varying across time intervals. However, based on the stabilized queue length distribution, we can derive the waiting time distribution and compute the mean waiting time of each application request for each time interval. For each time interval $\ell \in \mathcal{T}$, let $X_{a\ell}$ be the waiting time of class a in time interval ℓ with cumulative distribution function (CDF) $W_{a\ell}(\cdot)$, and $X_{j\ell}$ be the waiting time of applications which are served by server j in time interval ℓ with CDF $W_{j\ell}(\cdot)$. Then $X_{a\ell}$ can be defined as $X_{a\ell} = X_{j\ell}$ with probability $v_{aj\ell}$, and the LST of the waiting time distribution for class a in time interval ℓ, $\tilde{W}_{a\ell}(s)$ can be defined as

$$\tilde{W}_{a\ell}(s) = \sum_j v_{aj\ell} \tilde{W}_{j\ell}(s)$$

where $\tilde{W}_{j\ell}(s)$ is the LST of the waiting time distribution at server j in time interval ℓ. For the stabilized $M/G/1$ queue, the P-K transform formula for $\tilde{W}_{j\ell}(s)$ is defined as

$$\tilde{W}_{j\ell}(s) = \frac{(1-\rho)s}{s - \lambda + \lambda \tilde{S}_{j\ell}(s)}.$$

Note that aggregate arrival rates $\lambda_{j\ell}$ at server j in time interval ℓ would be $\lambda_{j\ell} = \sum_a v_{aj\ell}\lambda_{a\ell}$. In addition, for a random variable Y with a target workload distribution $H(\cdot)$ and processing speed of server j in time interval ℓ, $\phi_{j\ell}$, the LST of service time distribution of server j in time interval ℓ, $\tilde{S}_{j\ell}(s)$ can be defined as $\tilde{S}_{j\ell}(s) = \tilde{H}(s/\phi_{j\ell})$ (since service time is defined as $Y/\phi_{j\ell}$). Now we have the LST of waiting time of application a in time interval ℓ as

$$\tilde{W}_{a\ell}(s) = \sum_j v_{aj\ell} \tilde{W}_{j\ell}(s) = \sum_j v_{aj\ell} \frac{(1-\rho)s}{s - \sum_a v_{aj\ell}\lambda_{a\ell} + \sum_a v_{aj\ell}\lambda_{a\ell}\tilde{H}(\frac{s}{\phi_{j\ell}})}.$$

Moreover, the mean waiting time of class a in time interval ℓ as

$$\bar{X}_{a\ell} = \sum_j v_{aj\ell} \bar{X}_{j\ell} = \sum_j \left(\frac{\lambda_{j\ell} E[Z_{j\ell}^2]}{2(1-\rho)} \right) = \sum_j v_{aj\ell} \left(\frac{\sum_a v_{aj\ell}\lambda_{a\ell}}{2(1-\rho)} \frac{(1+C^2)(E[Y])^2}{\phi_{j\ell}^2} \right) \tag{18}$$

where C^2 is SCOV of the target workload. Based on the analysis of waiting times, it is intuitive to think that waiting times of applications can be controlled in a straightforward manner by adjusting

processing speeds of powered-on servers. Further, if we do not consider speed scaling and enforce that processing speed of powered-on servers would be constant across time intervals such that $\phi_{j\ell} = \phi \quad \forall j \in \mathcal{N}, \ell \in \mathcal{T}$, then the waiting time distributions are also stabilized as well as the queue length across time intervals. In addition, for speed scaling, the mean waiting time of each application class would be bounded by the minimum and maximum processing speeds of assigned servers. Since we have stabilized aggregate workload distribution and queue length distribution for each powered-on server, an upper bound on the mean waiting times of each class a, τ_a (e.g. $\bar{X}_{a\ell} \leq \tau_a \quad \forall a \in \mathcal{A}, \forall \ell \in \mathcal{T}$) can be easily obtained based on the desired mean queue length $\bar{L}$ which is introduced in Section 2.3 as follows:

$$\tau_a = \frac{E[Y]\bar{L}}{\phi_a^{\min}} \quad \text{where } \phi_a^{\min} = \min\{\phi_{j\ell}, j \in \mathcal{N}_a^{\text{on}}, \ell \in \mathcal{T}\}. \tag{19}$$

In other words, the upper bound τ_a is derived by assuming the worst-case scenario reflecting the situation that class a workloads are solely routed to server j with the minimum speed ϕ_{min}^j among the powered-on servers. Consequently, an upper bound on the mean waiting times of the overall system across time intervals, τ can be defined as

$$\tau = \max\{\tau_a, a \in \mathcal{A}\}. \tag{20}$$

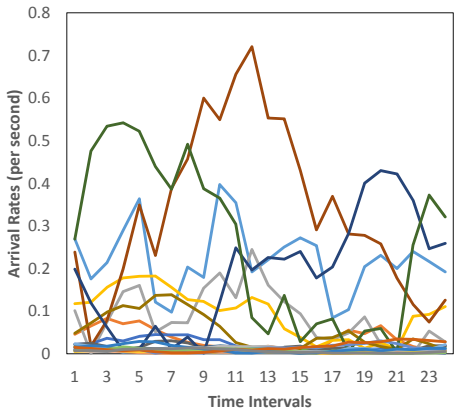
Based on the analysis of the waiting time, we define the constraints on the processing speed of each server j in time interval ℓ, $\phi_{j\ell}$, to define the upper bound on the mean waiting time as follows

$$r\phi_{\max}^j \leq \phi_j \qquad \forall j \in \mathcal{N}, \tag{21}$$

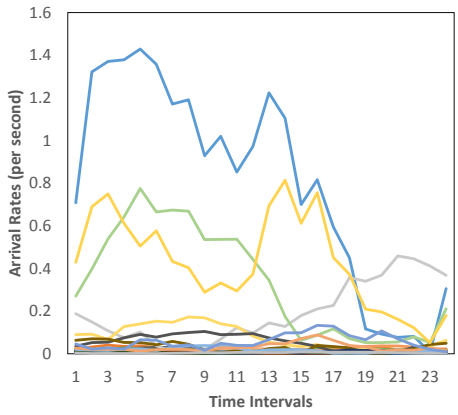
where r is the bound ratio of the minimum speed to the maximum speed of servers. We will solve our proposed MIP problem by adding constraint (21) and analyzing the results in Section 5.

5 Numerical Evaluation

In this section we introduce simulation results to show that our suggested approach stabilizes the mean queue length and we provide performance bound on the mean waiting time. For the numerical evaluation, we select two test instances; 10 servers with 20 applications ($|\mathcal{N}| = 10$, $|\mathcal{A}| = 20$) and 20 servers with 40 applications ($|\mathcal{N}| = 20$, $|\mathcal{A}| = 40$) from [10]. Recall that, although data centers generally operate hundreds or thousands of servers, considering that in most data



(a) Arrival rates of 20 applications



(b) Arrival rates of 40 applications

Figure 2: Time-varying arrival rates across 24 1-hour time intervals

centers applications of a single client are clustered among servers for security and confidentiality, test instances of 10 or 20 servers would be large enough to show that our suggested approach can be applied successfully in practice. We model the arrival process of each application's request as a non-homogeneous Poisson process with time-varying arrival rates $\lambda_{a\ell}$ for 24 one-hour time intervals (i.e. 24 equally spaced time intervals for a one-day cycle) such that $\ell \in \mathcal{T} = \{1, 2, \ldots, 24\}$. In Figures 2a and 2b, we plot the arrival rates of both test instances (20 applications and 40 applications, respectively) across 24 one-hour time intervals. Also, for the workload distribution of each application a, $H_a(\cdot)$, we choose uniform distribution with mean workload $1/\theta_a$ and SCOV C_a^2. We also assume that each application has a requirement for three types of resources ($\mathcal{K} = \{1, 2, 3\}$) and that each server has a capacity limit for those resource types. For energy cost, we define fixed energy costs for powered-on server j as much higher than unit operation costs of server j, c_j, considering the fact that energy cost of servers is dominated by fixed costs as mentioned in [2]. In this study, we solved the suggested MIP problem by using CPLEX 12.6 Concert Technology on an Intel Core i7-3740QM 2.70GHz with 16GB memory, and we used Java to develop a discrete event simulation framework. We ran simulation experiments by using decision variables, $x_{aj\ell}$, $y_{j\ell}$, $\phi_{j\ell}$, and $v_{aj\ell}$, which are determined by solving the suggested MIP problem.

5.1 Computational Time

Before analyzing the simulation results, to describe the complexity of our suggested MIP problem, we summarize in Table 2 the objective function values and the computational time obtained by

Table 2: Objective Function Value and CPU Time: 10 Servers and 20 Applications and 20 Servers and 40 Applications

Time	10 Servers and 20 Classes		20 Servers and 40 Classes with 3600 Seconds Time Limit			20 Servers and 40 Classes with 7200 Seconds Time Limit			20 Servers and 40 Classes with 10800 Seconds Time Limit		
	Objective Function	CPU Time (seconds)	Objective Function	CPU Time (seconds)	Gap (%)	Objective Function	CPU Time (seconds)	Gap (%)	Objective Function	CPU Time (seconds)	Gap (%)
1	282.00	694.49	768.17	3603.86	13.38	760.33	7208.60	12.07	760.33	10825.67	12.01
2	255.00	148.59	969.50	3604.75	0.74	962.83	7201.29	0.05	962.33	7068.53	0.00
3	282.00	1534.94	1029.33	70.78	0.00	1029.33	70.78	0.00	1029.33	70.78	0.00
4	309.00	408.02	1037.50	3604.86	0.91	1033.17	7210.60	0.50	1033.17	10800.25	0.49
5	342.00	128.34	1099.67	100.06	0.00	1099.67	100.06	0.00	1099.67	100.06	0.00
6	280.00	1076.35	1029.33	1233.63	0.00	1029.33	1233.63	0.00	1029.33	1233.63	0.00
7	288.00	163.40	957.67	3603.56	6.51	952.33	7225.78	5.99	952.33	10863.99	5.99
8	315.00	2007.39	953.33	3610.75	6.08	953.33	7210.01	6.08	945.67	10831.73	5.32
9	327.00	843.70	833.67	3607.44	12.76	833.67	7206.97	12.58	825.00	10808.16	11.53
10	356.00	663.96	868.67	3603.47	10.65	874.00	7223.31	10.98	874.00	10815.67	10.98
11	356.00	2455.92	822.67	3606.51	12.17	835.17	7218.76	13.67	835.17	10817.75	13.67
12	345.00	91.56	828.83	3605.96	13.21	817.00	7219.87	11.75	817.00	10826.99	11.75
13	318.00	180.09	916.83	3609.30	2.35	902.50	4176.90	0.79	895.33	4176.90	0.00
14	310.00	245.81	853.17	3605.03	2.13	853.17	7204.36	2.13	853.17	10810.11	2.13
15	295.00	305.95	689.00	3602.81	8.75	689.00	7210.23	8.40	689.00	10816.66	8.31
16	256.00	196.19	757.00	3602.22	4.83	757.00	7218.20	4.76	754.17	10816.77	4.40
17	256.00	203.53	690.33	3608.10	11.70	688.00	7217.12	11.29	680.50	10869.92	10.20
18	260.00	121.98	647.17	3603.84	6.89	646.17	7203.58	6.24	639.50	10811.99	5.77
19	295.00	391.58	625.00	3607.68	11.81	622.67	7219.65	11.48	620.67	10864.83	11.20
20	273.00	170.18	614.50	3608.63	10.31	607.83	7209.07	9.32	607.83	10814.04	9.32
21	266.00	89.03	620.67	3605.70	10.87	620.67	7206.45	10.46	620.33	10809.39	9.93
22	263.00	142.93	613.50	3605.06	9.50	611.67	7209.62	9.02	610.17	10818.20	8.49
23	263.00	103.35	627.83	3613.53	15.24	627.83	7217.48	15.22	614.50	10839.07	13.38
24	255.00	214.84	671.00	3621.70	11.45	671.00	7226.45	11.29	670.17	10824.64	10.28

solving MIP-ℓ across time intervals $\ell \in \mathcal{T}$ for the two test instances. Note that we obtain the optimal objective function values for the instance of 10 servers with 20 applications within 3600 seconds; however, due to the complexity of the problem, we report the objective function values of the best feasible solution and MIP gap for 3600, 7200, and 10800 seconds time limits for the instance of 20 servers and 40 applications.

5.2 Analysis of Time-stability

Firstly, we analyze the mean queue lengths of each powered-on server to check whether our suggested approach achieves time-stability. Recall that as described in Sections 2.3, our suggested approach can be utilized to stabilize the mean queue length by matching the first and second moments for the desired queue length $\bar{L}$. For the numerical evaluation, we select the desired mean queue length $\bar{L}$ and the first and second moments ($E(Y)$ and $E(Y^2)$, respectively) as $\bar{L} = 3$ with $E(Y) = 2.3$ and $E(Y^2) = 7.2737$ for 10 servers with 20 applications and $\bar{L} = 4$ with $E(Y) = 3.7$ and $E(Y^2) = 27.38$ for 20 servers with 40 applications. In both cases, we select the desired traffic intensity as $\rho = 0.8$. Figures 5a and 6a depict the mean queue length of powered-on servers across 24 time intervals of both test instances. Although there exist some variations, the mean queue lengths at each

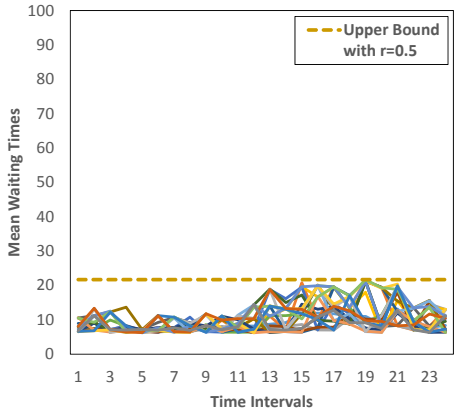
powered-on server are approximately stabilized as desired across 24 time intervals as compared to time-varying arrival rates shown in Figures 2a and 2b. We would like to note that if we match more moments for the target workload distribution, then the mean queue lengths would be more stable across time intervals. Also, we compare the mean queue length resulted by using the decision variables determined by solving the modified MIP problem which uses the following constraint (22) instead of time-homogeneity constraints (11), (13), and (14),

$$\sum_{a \in \mathcal{A}} \frac{\lambda_{a\ell}}{\theta_a} v_{aj\ell} \leq \rho \phi_{j\ell} \qquad \forall j \in \mathcal{N}. \tag{22}$$

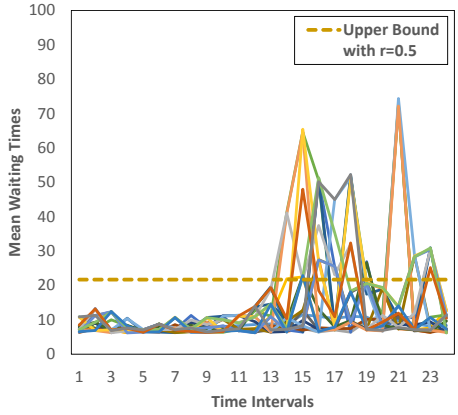
The above constraint (22) ensures that the average workload that arrives at every powered-on server must not exceed the desired traffic intensity ρ for fair comparison of our suggested approach. As shown in Figures 5b and 6b, the mean queue length would not be stabilized without using time-homogeneity constraints as compared to Figures 5a and 6a. Also, we compare the objective function values obtained by solving our suggested MIP problem with values obtained by solving the modified MIP problem to check how much additional energy cost is required for stabilizing the mean queue length, as well as provide performance guarantees on the mean waiting time. As we can see in Figures 7a and 7b, total energy cost for stabilizing the mean queue length is slightly higher, but it can be reasonably ignored since we can provide provable performance bounds.

5.3 Performance Bounds on the Mean Waiting Times

As mentioned in Section 4, performance bound on the mean waiting time of requests can be derived by using the bound ratio r of the minimum processing speed for each powered-on server based on constraint (21). Here, we select the bound ratio r as $r = 0.5$ for 10 servers with 20 applications and $r = 0.4$ for 20 servers with 40 applications, and then we contrast the mean waiting times obtained by using constraint (21) with the results obtained without using constraint (21) for both test instances. Figures 3a and 4a depict the mean waiting times and the performance bounds of both test instances, respectively. Note that the mean waiting time is strictly bounded by τ which can be derived as Equation (20) as opposed to the results obtained without using constraint (21) shown in Figures 3b and 4b.

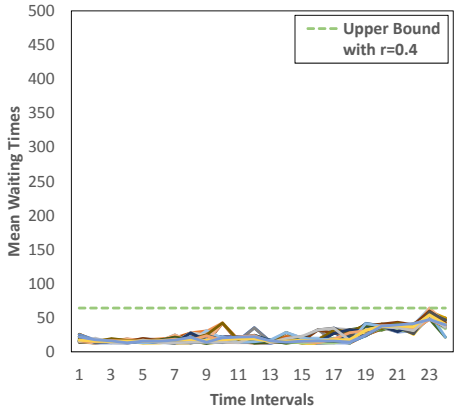


(a) with bound constraint with $r = 0.5$

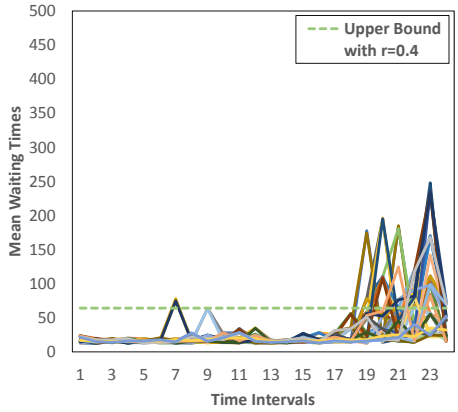


(b) without bound constraint

Figure 3: Mean waiting times for the test instance of 10 servers with 20 applications



(a) with bound constraint with $r = 0.4$

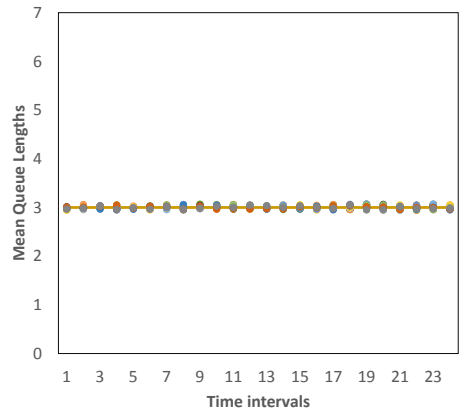


(b) without bound constraint

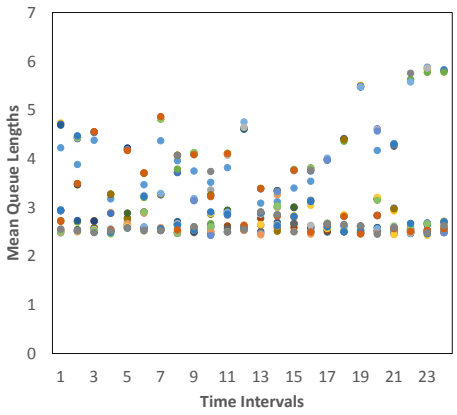
Figure 4: Mean waiting times for the test instance of 20 servers with 40 applications

5.4 Benchmark

In addition, we compare our approach against the algorithm proposed by Lin et al. [23] as a benchmark. Lin et al. [23] considered a data center optimization problem and proposed an off-line algorithm to determine the number of active servers for minimizing total energy cost, which consists of operating costs and switching costs. The main idea of their suggested approach is to minimize energy costs while satisfying the QoS constraints, which are defined by using standard queueing theory results as shown in Van et al. [29] and Rao et al. [28]. Note that Lin et al. [23] implemented a dispatching rule which delivers equal amount of workload to each powered-on server (i.e. load balancing). Note that Lin et al. [23] suggested a model that uses quasi-steady-state approximations

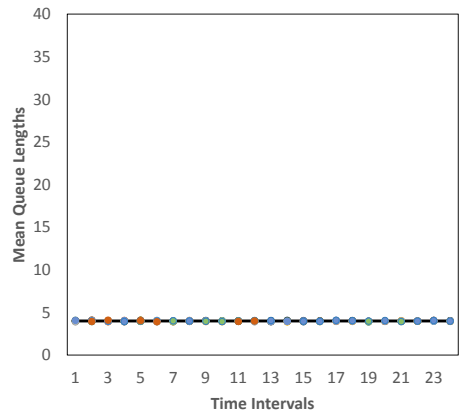


(a) With time-homogeneity constraints for $\bar{L} = 3$

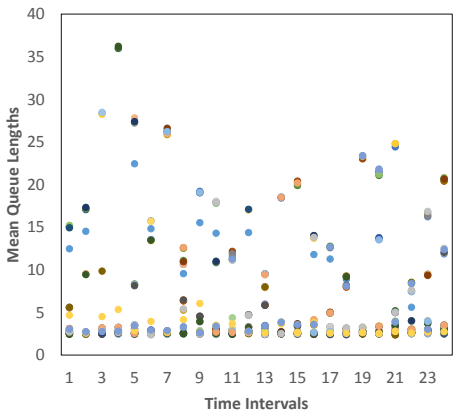


(b) Without time-homogeneity constraints

Figure 5: Mean queue lengths for the test instance of 10 servers with 20 applications



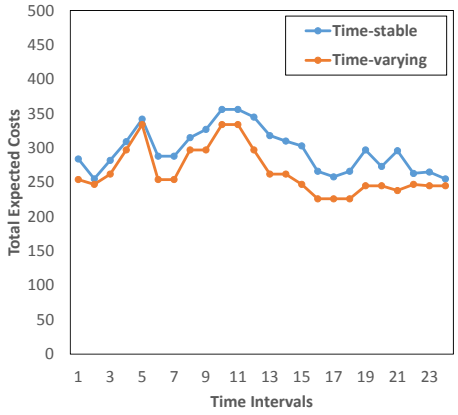
(a) With time-homogeneity constraints for $\bar{L} = 4$



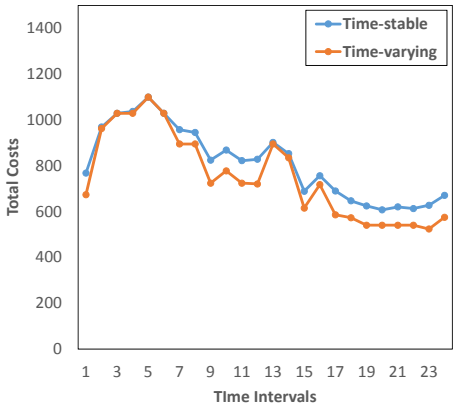
(b) Without time-homogeneity constraints

Figure 6: Mean queue lengths for the test instance of 20 servers with 40 applications

(i.e. the metrics are piecewise constant for each time interval long enough for the system to reach steady-state) to define QoS constraints for non-stationary system, and it is worthwhile pointing out that our suggested approach can provide provable performance bound for time-varying and transient systems. Since Lin et al. [23] assumed that both workloads and servers are homogeneous and did not consider resource capacities and speed scaling, we slightly modified their model and ours, and compared the results. In fact, Lin et al. [23] proposed a QoS constraint that the average waiting time is bounded by certain thresholds based on $M/G/1$ processor sharing queue, and we re-defined their constraint by using multiclass $M/G/1$ queue with FCFS (Gautam [15]) for the heterogeneous workload. Also, since our suggested an MIP problem does not consider switching costs (i.e. cost for



(a) 10 servers with 20 applications



(b) 20 servers with 40 applications

Figure 7: Comparison of energy cost for two test instances: time-stable results vs time-varying results

powering server on and off), we modified our model by adding the following constraints (23)-(24) and cost function (25) so that our MIP problem determines operational decisions while minimizing both operating costs and switching costs. In order to include cost for powering on/off, we define additional decision variables, $u_{j\ell}^{\text{on}}$ (e.g. $u_{j\ell}^{\text{on}} = 1$ if server j is turned on at the beginning of time interval ℓ) and $u_{j\ell}^{\text{off}}$ (e.g. $u_{j\ell}^{\text{off}} = 1$ if server j is turned off at the end of time interval $\ell - 1$). Then we can define the constraints to determine variables $u_{j\ell}^{\text{on}}$ and $u_{j\ell}^{\text{off}}$ as follows:

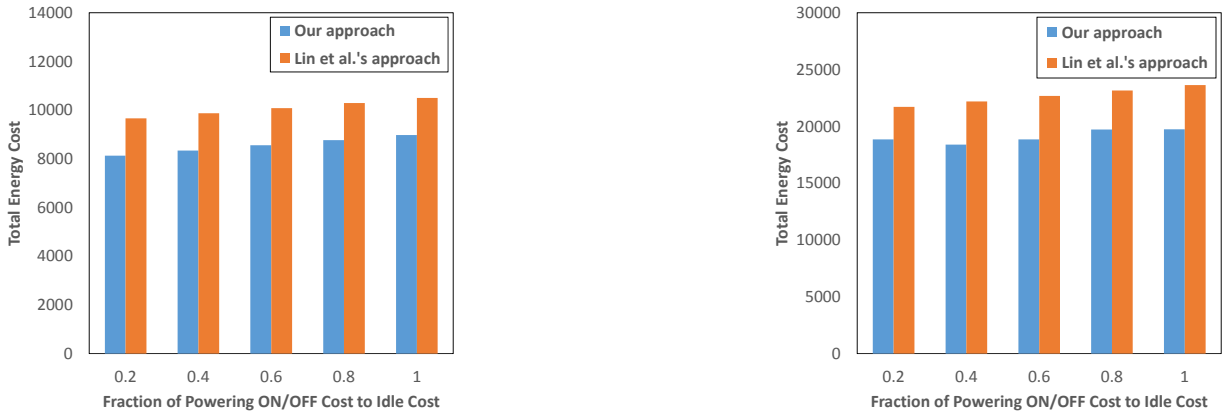
$$y_{j\ell} - y_{j(\ell-1)} \leq u_{j\ell}^{\text{on}} \quad \forall j \in \mathcal{N}, \forall \ell \in T \tag{23}$$

$$y_{j(\ell-1)} - y_{j\ell} + u_{j\ell}^{\text{on}} = u_{jt}^{\text{off}} \quad \forall j \in \mathcal{N}, \forall \ell \in T. \tag{24}$$

Recall that decision variable $y_{j\ell} = 1$ if server j is active in time interval ℓ; otherwise, $y_{j\ell} = 0$. By using $u_{j\ell}^{\text{on}}$ and $u_{j\ell}^{\text{off}}$, now we can define the cost function for turning servers on/off, which can be added to the objective function as follows:

$$\sum_{j \in \mathcal{N}} \sum_{\ell \in \mathcal{T}} (c_j^{\text{on}} u_{j\ell}^{\text{on}} + c_j^{\text{off}} u_{j\ell}^{\text{off}}), \tag{25}$$

where c_j^{on} and c_j^{off} are cost for turning servers on/off, respectively. To compare the approaches, we solved both problem test instances without considering resource capacity (as well as constraints (23)-(24)) and set cost for turning servers on/off so that the fractions of cost to power cost at "idle" state of each server j, f_j as 0.2, 0.4, 0.6, 0.8, and 1.0. For example, if the fraction is 0.2, then the cost for each time the servers turn on/off would be 20% of "idle" power cost. In the following



(a) 10 servers with 20 applications

(b) 20 servers with 40 applications

Figure 8: Comparison of total energy cost: Our approach vs Lin et al.'s approach [23]

Figures 8a and 8b, we compare total energy costs of our approach against that of Lin et al. [23] by setting thresholds for the mean waiting time as equal to the upper bound obtained by our suggested approach under the same traffic intensity. As shown in Figures 8a and 8b, the total energy cost of our suggested approach is smaller than that of the approach in Lin et al. [23] for the same upper bounds on the mean waiting time. This is because both the variability of the queue length and the waiting time get bigger for multiclass jobs (heterogeneous workloads), and thus more servers are needed to provide the same performance bounds can be derived by our suggested approach.

6 Concluding Remarks and Future Work

In this paper, we considered a fairly common scenario in cloud computing where requests of heterogeneous applications arrive at time-varying rates to a data center that consists of heterogeneous servers. For such a time-varying and heterogeneous system, it is difficult to achieve energy efficiency and provide performance guarantees simultaneously; therefore, to tackle this shortcoming, we suggested an approach to achieve time-stability. For performance guarantees, our suggested approach stabilizes (i) aggregate workload distribution based on moment matching approximation and (ii) arrival rates based on routing fractions to achieve time stable queue length distribution. We also derived time-homogeneity constraints based on our approach and formulated an MIP problem to optimally determine various decisions of sizing, assignments, routing, and speed scaling to minimize energy costs while considering time-stability. In fact, we can obtain time-stability of queue length distribution by matching sufficiently many moments based on the suggested approach. We showed

that we can also obtain reasonable time-stability for the mean queue length by matching only the first and second moment. In addition, based on time-stable queue lengths distribution, the waiting times of applications are easily controlled by obtaining bounds on processing speeds. Our suggested approach indeed enables us to provide performance guarantee on the mean waiting times, which is an extremely useful measurement in terms of QoS for both users and service providers. To evaluate our approach, we developed a simulation model and summarized results that support our claim. In the present study we do not focus on developing an algorithm to efficiently solve the proposed MIP for the large-scale problem, we leave it for future work.

We acknowledge that our approach is purely based on historical information and further work needs to be done before being implemented in practice. We believe that results from this research can be used by practitioners to develop an initial cut for setting up servers in their data centers. Then an appropriate analysis tailored to the type of applications hosted by the data center would need to be performed to determine the number of stand-by servers to be positioned to handle unexpected surges in demand. Then, traffic can be monitored and significant deviations from time-stable queue lengths in real time can be used to trigger the use of stand-by servers. Further, one could develop control algorithms to dynamically change server settings on the fly and use our proposed approach as the baseline static setting. We propose to extend our study to develop an on-line algorithm for such real-time control problems in the future.

7 Online Supplement

An online supplementary material is available for this article. Go to the publisher's online edition of *IIE Transaction*, theorem and detailed proofs for theoretical foundations for constraint (11) of the proposed MIP formulation.

Acknowledgements

This material is partially supported by the AFOSR under Contract No.FA9550-13-1-0008. We thank the anonymous reviewers and editors for their insightful comments that lead to significant improvements in the content and presentation of this work. We are grateful to Dr. Butenko for his help to show the MIP complexity.

References

[1] Barnes, J. A. and Meili, R. (1997). A stationary poisson departure process from a minimally delayed infinite server queue with non-stationary poisson arrivals. *Journal of Applied Probability*, pages 767–772.

[2] Barroso, L. and Holzle, U. (2007). The case for energy-proportional computing. *Computer*, 40(12):33–37.

[3] Bertini, L., Leite, J., and Mossé, D. (2008). Dynamic configuration of web server clusters with qos control. In *WIP Session of the 20th Euromicro Conference on Real-Time Systems*, volume 4.

[4] Bodik, P., Armbrust, M. P., Canini, K., Fox, A., Jordan, M., and Patterson, D. A. (2008). A case for adaptive datacenters to conserve energy and improve reliability. *University of California at Berkeley, Tech. Rep. UCB/EECS-2008-127*.

[5] Brown, R. et al. (2008). Report to congress on server and data center energy efficiency: Public law 109-431. *Lawrence Berkeley National Laboratory*.

[6] Chase, J. S., Anderson, D. C., Thakar, P. N., Vahdat, A. M., and Doyle, R. P. (2001). Managing energy and server resources in hosting centers. In *ACM SIGOPS Operating Systems Review*, volume 35, pages 103–116.

[7] Chen, Y., Das, A., Qin, W., Sivasubramaniam, A., Wang, Q., and Gautam, N. (2005). Managing server energy and operational costs in hosting centers. In *ACM SIGMETRICS Performance Evaluation Review*, volume 33, pages 303–314.

[8] Feldman, Z., Mandelbaum, A., Massey, W. A., and Whitt, W. (2008). Staffing of time-varying queues to achieve time-stable performance. *Management Science*, 54(2):324–338.

[9] Foley, R. D. (1986). Stationary poisson departure processes from non-stationary queues. *Journal of Applied Probability*, pages 256–260.

[10] Gallego Arrubla, J. A., Ko, Y. M., Polansky, R. J., Pérez, E., Ntaimo, L., and Gautam, N. (2013). Integrating virtualization, speed scaling, and powering on/off servers in data centers for energy efficiency. *IIE Transactions*, 45(10):1114–1136.

[11] Gandhi, A., Chen, Y., Gmach, D., Arlitt, M., and Marwah, M. (2011). Minimizing data center SLA violations and power consumption via hybrid resource provisioning. In *Green Computing Conference and Workshops (IGCC), 2011 International*, pages 1–8.

[12] Gandhi, A., Gupta, V., Harchol-Balter, M., and Kozuch, M. A. (2010). Optimality analysis of energy-performance trade-off for server farm management. *Performance Evaluation*, 67(11):1155–1171.

[13] Gandhi, A., Harchol-Balter, M., Das, R., and Lefurgy, C. (2009). Optimal power allocation in server farms. In *ACM SIGMETRICS Performance Evaluation Review*, volume 37, pages 157–168.

[14] Gandhi, A., Harchol-Balter, M., Kozuch, M., et al. (2012). Are sleep states effective in data centers? In *Green Computing Conference (IGCC), 2012 International*, pages 1–10.

[15] Gautam, N. (2012). *Analysis of queues: methods and applications*. CRC Press.

[16] Gmach, D., Rolia, J., Cherkasova, L., and Kemper, A. (2007). Workload analysis and demand prediction of enterprise data center applications. In *Workload Characterization, 2007. IISWC 2007. IEEE 10th International Symposium on*, pages 171–180.

[17] Guenter, B., Jain, N., and Williams, C. (2011). Managing cost, performance, and reliability tradeoffs for energy-aware server provisioning. In *INFOCOM, 2011 Proceedings IEEE*, pages 1332–1340.

[18] Hamilton, J. (2008). Cost of power in large-scale data centers. [Online], Available:`http://perspectives.mvdirona.com`.

[19] Koomey, J. (2011). Growth in data center electricity use 2005 to 2010. *A report by Analytical Press, completed at the request of The New York Times*, page 9.

[20] Kusic, D., Kephart, J. O., Hanson, J. E., Kandasamy, N., and Jiang, G. (2009). Power and performance management of virtualized computing environments via lookahead control. *Cluster Computing*, 12(1):1–15.

[21] Kwon, S. and Gautam, N. (2015). Time-stable performance in parallel queues with non-homogeneous and multi-class workloads. *IEEE/ACM Transactions on Networking*. (accepted), DOI: 10.1109/TNET.2015.2406280.

[22] Lin, M., Liu, Z., Wierman, A., and Andrew, L. L. (2012). Online algorithms for geographical load balancing. In *Green Computing Conference (IGCC), 2012 International*, pages 1–10.

[23] Lin, M., Wierman, A., Andrew, L. L., and Thereska, E. (2013). Dynamic right-sizing for power-proportional data centers. *IEEE/ACM Transactions on Networking*, 21(5):1378–1391.

[24] Liu, Y. and Whitt, W. (2012). Stabilizing customer abandonment in many-server queues with time-varying arrivals. *Operations Research*, 60(6):1551–1564.

[25] Liu, Z., Chen, Y., Bash, C., Wierman, A., Gmach, D., Wang, Z., Marwah, M., and Hyser, C. (2012). Renewable and cooling aware workload management for sustainable data centers. In *ACM SIGMETRICS Performance Evaluation Review*, volume 40, pages 175–186.

[26] Liu, Z., Wierman, A., Chen, Y., Razon, B., and Chen, N. (2013). Data center demand response: Avoiding the coincident peak via workload shifting and local generation. *Performance Evaluation*, 70(10):770–791.

[27] Raghavendra, R., Ranganathan, P., Talwar, V., Wang, Z., and Zhu, X. (2008). No power struggles: Coordinated multi-level power management for the data center. In *ACM SIGARCH Computer Architecture News*, volume 36, pages 48–59.

[28] Rao, L., Liu, X., Xie, L., and Liu, W. (2010). Minimizing electricity cost: optimization of distributed internet data centers in a multi-electricity-market environment. In *INFOCOM, 2010 Proceedings IEEE*, pages 1–9.

[29] Van, H. N., Tran, F. D., and Menaud, J.-M. (2009). SLA-aware virtual resource management for cloud infrastructures. In *Computer and Information Technology, 2009. CIT'09. Ninth IEEE International Conference on*, volume 1, pages 357–362.

[30] Vogels, W. (2008). Beyond server consolidation. *ACM Queue*, 6(1):20–26.

[31] Wang, K., Lin, M., Ciucu, F., Wierman, A., and Lin, C. (2015). Characterizing the impact of the workload on the value of dynamic resizing in data centers. *Performance Evaluation*, 85:1–18.

[32] Whitt, W. (2014). Stabilizing performance in a single-server queue with time-varying arrival rate. Technical report, Working paper, Columbia University, available at: www. columbia. edu.

[33] Xu, D. and Liu, X. (2012). Geographic trough filling for internet datacenters. In *INFOCOM, 2012 Proceedings IEEE*, pages 2881–2885.

[34] Yao, Y., Huang, L., Sharma, A., Golubchik, L., and Neely, M. (2012). Data centers power reduction: A two time scale approach for delay tolerant workloads. In *INFOCOM, 2012 Proceedings IEEE*, pages 1431–1439.

[35] Zhu, Z., Bi, J., Yuan, H., and Chen, Y. (2011). SLA-based dynamic virtualized resources provisioning for shared cloud data centers. In *Cloud Computing (CLOUD), 2011 IEEE International Conference on*, pages 630–637.