Guaranteeing performance based on time-stability for energy-efficient data centers

Soongeol Kwon and Natarajan Gautam

Department of Industrial and Systems Engineering, Texas A&M University, College Station, TX, USA

ABSTRACT
We consider a system of multiple parallel single-server queues where servers are heterogeneous with resources of different capacities and can 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.

1. Introduction
We consider a data center that 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 that 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 based on 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., 2007; Lin et al., 2012; Lin et al., 2013; Liu et al., 2013; Kwon and Gautam, 2015). Using that assumption, we model each cycle as being 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); however, their workloads are so small 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 discussed scenario, 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 the servers in the data center in an energy-efficient manner while satisfying performance guarantees by achieving time-stability. For time-varying and heterogeneous systems, the arrival rates of requests change so fast that a steady state of the system is not reached, therefore making it extremely difficult to provide performance guarantees. Without assuming that the system reaches a steady state, one can provide performance guarantees by powering upon a large number of servers responding to peak workload; however, that causes a huge additional energy cost. In this context, achieving time-stability is extremely useful for managing a non-homogeneous and transient system, as it enables operators to effectively design and analyze the time-varying system, it provides guaranteed QOS as desired, and it 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 for switching between assignments.
- **Sizing**: Each server can be dynamically powered on or off across time intervals. However, 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 to be 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 appropriately performed in an integrated fashion to achieve both time-stability and energy efficiency through the suggested framework described in Sections 2 and 3. Figure 1 shows the scenario and the main objective of this study. Next we review the relevant literature and introduce the notation used in this article.

### 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 significant levels of attention from 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 the one hand, this over-provisioning of servers results in low utilization levels as reported in Barroso and Holzle (2007) and Vogels (2008). On the other hand, they incur a significant amount of energy usage for operating and cooling servers as explained and documented in Hamilton (2008) and Koomey (2009). Fortunately, to abate the skyrocketing energy consumption in data centers (see the report by Brown et al. (2008)), 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 them to enter a power-saving mode) in accordance with traffic changes, while considering performance guarantees, and thus a number of techniques for efficient server provisioning have been proposed to address this problem. Gandhi et al. (2011) 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. (2011) 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. (2015) provided an analytic framework that captures non-stationarities and stochastic variation of workloads for dynamic re-sizing in data centers. Lin et al. (2013) 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 have considered heterogeneous workloads for server provisioning and allocation problems. There is another body of literature that has proposed an optimization approach based on powering servers on/off and Dynamic Voltage/Frequency Scaling (DVFS) to minimize energy consumption. Bertini et al. (2008) 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. (2013) introduced a unified methodology that combines virtualization, speed scaling, and powering off servers, so as to efficiently operate data centers while incorporating the inherent variability and uncertainty.

Although all of the aforementioned research is complementary to our work, the key difference is that our study suggests an approach that aims to not only save an 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 has not been effectively addressed. Although time-stability has received little attention in the context of data center operation, there have been some research studies in the queueing area. Foley et al. (1986) and Barnes and Meili (1997) showed that the departure process from the $M_t / G_t / \infty$ queue can be made stationary, and recently, Whitt (2015) suggested the rate matching control algorithm, which stabilizes the queue length distribution for a $G_t / G_t / 1$ single-server queue where both arrival rate and service rate are time-varying. There is another body of literature that provides algorithms to determine appropriate staffing levels for call centers. Feldman et al. (2008) proposed a simulation-based iterative-staffing algorithm for time-stable delay probability, and Liu and Whitt (2012) suggested a formula-based algorithm to stabilize abandonment probabilities and expected delays using offered-load-based approximations for a queueing model with a non-homogeneous Poisson arrival process and customer abandonment.

In fact, our previous work (Kwon and Gautam, 2015) 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 that paper, 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 a 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 used 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 data centers that provide a hosting service for several other companies. In general, hosting data centers cluster applications of each company and assign
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 a 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 could be hosted on just one server. Also, we assume that servers are heterogeneous and host a mixture of heterogeneous application requests, and the processing speed of a server can be changed for energy efficiency (in Kwon and Gautam (2015), 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 Kwon and Gautam (2015). Therefore, our suggested model in this study is fundamentally different from the previously considered problem.

For the purpose of this study, we provide an analytical framework that 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 the queue length distributions of each powered-on server and provide performance guarantees on the waiting time of each application class. Moreover, for energy efficiency we propose an optimization model to minimize the 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 that unifies sizing, assignment, routing, and speed scaling under heterogeneous conditions, which have seldom been implemented jointly; (ii) define time-homogeneity constraints that ensure time-stability; (iii) suggest an approach to provide performance guarantees based on their time-stability; and (iv) introduce an optimization problem with an MIP formulation to reduce energy cost while considering time-stability. The remainder of this article is organized as follows: Section 2 introduces our concept 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. Notation

For the scenario considered in this study, we set the notation 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 based on a general distribution $H_a(\cdot)\sim\text{mean } 1/\theta_a$ and SCOV $C_a^2$. Also, each class $a$ has requirements $b^k_a$ 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 the maximum processing speed $\phi_{j,\ell}^a$ and capacity limit for each resource type $k \in \mathcal{K}$, $b^k_j$. 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 $\rho_{\ell}$ 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 $\rho$. 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 $\ell$, $x_{a,j,\ell}$ (e.g., $x_{a,j,\ell} = 1$ if class $a$ assigned to server $j$ in time interval $\ell$; otherwise, $x_{a,j,\ell} = 0$); (ii) whether server $j$ must be powered on or off in time interval $\ell$, $y_{j,\ell}$ (e.g., $y_{j,\ell} = 1$ if server $j$ is powered on in time interval $\ell$; otherwise, $y_{j,\ell} = 0$); (iii) the processing speed for server $j$ in time interval $\ell$, $\phi_{j,\ell}$; and (iv) the fraction of jobs of application $a$ to be routed to server $j$ in time interval $\ell$, $\tau_{a,j,\ell}$. Each server $j$ must be powered on or off in each time interval $\ell$ and thus there will be $N_j = \sum_{j \in \mathcal{N}} y_{j,\ell}$ powered-on servers in each time interval $\ell$. We summarize the set of indices, parameters, and decision variables that are used to define optimization problem in Table 1.

| Table 1. Indices, parameters, and decision variables for the considered optimization problem |
| --- |
| **Indices** |
| $\mathcal{T}$ |
| $\mathcal{A}$ |
| $\mathcal{N}$ |
| $\mathcal{K}$ |
| **Parameters** |
| $\lambda_{a,\ell}$ |
| $\gamma_{a,\ell}$ |
| $\rho_{\ell}$ |
| $b^k_a$ |
| $b^k_j$ |
| $\phi_{j,\ell}$ |
| $\phi_{j,\ell}^a$ |
| **Decision variables** |
| $x_{a,j,\ell}$ |
| $y_{j,\ell}$ |
| $\tau_{a,j,\ell}$ |

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 was 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 the aggregate workload being defined by a mixture of heterogeneous application classes. To achieve 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 based on a stationary distribution of a homogeneous \( M/G/1 \) queue. In Section 2.2, we will propose an approach to obtain a time-stable queue length distribution by stabilizing both the aggregate workload distribution and arrival rates, and in Section 3 we will show that time-stability can be valid for speed scaling. For mathematical representation, for all \( j \in N_\ell \), at time \( t \) let \( X_j(t) \) be the number of requests in the 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 the 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 independent of \( t \). Based on the time-stability in the queue length distribution, the 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 the 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 the queueing process of each powered-on server based on (i) a time-invariant aggregate workload distribution obtained by a moment-matching approximation and (ii) rate matching between the arrival rates and processing speeds, by appropriately selecting routing fractions. 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

To achieve time-stability, we seek to stabilize an aggregate workload distribution of each powered-on server, with the servers hosting 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 appropriately control the system. This way each powered-on server receives arrivals of requests with a homogeneous aggregate workload distribution \( H(\cdot) \) based on the moment-matching approximation. To begin, for the desirable traffic intensity \( \rho \) and the target nominal speed \( \phi \), we determine the minimum number of powered-on servers for each time interval \( \ell \in T, N_\ell \), that satisfies the following inequality:

\[
\sum_{a \in A} \lambda_a \tilde{H}_a(s) \leq N_\ell \tilde{H}(s) \rho \phi \theta \quad \forall s \geq 0, \tag{1}
\]

where the left-hand side is the LST of the aggregate workload distribution for the entire system and the right-hand side represents the LST of the aggregate workload distribution of \( N_\ell \) time-stable servers, where each of the \( N_\ell \) servers has the target workload distribution \( H_\ell(\cdot) \) and arrival rate of \( \rho \phi \theta \). Based on the inequality (1), \( N_\ell \) can be determined so that the workload served by each of the \( N_\ell \) servers would be less than or equal to the target workload (which is defined by traffic intensity \( \rho \), processing speed \( \phi \), 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:

\[
\sum_{a \in A} \lambda_a \tilde{H}_a(s) \leq N_\ell \rho \phi \left( \tilde{H}'(0) \right) \sum_{a \in A} \lambda_a \tilde{H}_a''(0) \leq N_\ell \rho \phi \left( \tilde{H}''(0) \right) \leq N_\ell \rho \phi \left( \tilde{H}'''(0) \right) \leq \cdots
\]

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 \( N_\ell \) servers is approximately equal to the target workload distribution \( H(\cdot) \). As discussed in Liu et al. (2012), Xu and Liu (2012), and Yao et al. (2012), 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, the arrival rate in time interval \( \ell \), \( \lambda_{0\ell} \), and the moment of workload distribution for each time interval \( \ell \), \( -\tilde{H}_0''(0), -\tilde{H}'_0''(0), -\tilde{H}_0'''(0), \ldots \) can be determined appropriately through the following equalities based on \( N_\ell \):

\[
\left( -\tilde{H}'(0) \right) \left( \lambda_{0\ell} + \sum_{a \in A} \lambda_a \right) = N_\ell \rho \phi
\]

\[
\left( -\tilde{H}'(0) \right) \left( \frac{\lambda_{0\ell}}{\theta_{0\ell}} + \sum_{a \in A} \frac{\lambda_a}{\theta_a} \right) = N_\ell \rho \phi
\]

\[
\left( -\tilde{H}''(0) \right) \left( \lambda_{0\ell} \tilde{H}_0''(0) + \sum_{a \in A} \lambda_a \tilde{H}_a''(0) \right) = N_\ell \rho \phi \tilde{H}''(0)
\]

\[
\left( -\tilde{H}'''(0) \right) \left( -\lambda_{0\ell} \tilde{H}_0'''(0) + \sum_{a \in A} \lambda_a (-\tilde{H}_a'''(0)) \right) = N_\ell \rho \phi (-\tilde{H}'''(0))
\]

\[
\vdots
\]
and then we can approximately define \( H_{0t} (\cdot) \) by combining the
derivatives. Recall that there is additional traffic that has low pri-
ority of QOS guarantees (e.g., non-interactive jobs are less sen-
sitive to the response time) and is also CPU-intensive (it does
not need other resource requirements). In this case, it is rea-
sonable to assume that this additional traffic can be assigned to
the system based on a Poisson process with parameter \( \lambda_{ag} \)
and workload distribution \( H_{0t} (\cdot) \) for each time interval \( \ell \) from
front-end proxy servers or other data centers. We would like to note
that our suggested approach allows additional traffic; thus, the
increment in workloads caused by an addition of traffic will not
degrade the desired performance but will reduce variability and
uncertainty in aggregate workload while also improving the util-
ization of CPU for each powered-on server. Note that if we suf-
ciently match many moments, then the above equalities would result in
\[
\lambda_{0t} \hat{H}_{0t}(s) + \sum_{a \in A} \rho_{at} \hat{H}_{a}(s) = N_t \hat{H}(s) \rho \phi \quad \forall s \geq 0, \tag{2}
\]
where \( H_{0t} (\cdot) \) has \(-\hat{H}'(0), -\hat{H}''(0), \ldots, \) as moments. By selecting additional traffic for several moments, our sug-
gested approach stabilizes the workload distribution for each
powered-on server.

### 2.2.2. Selecting routing fractions for time-stable arrival rates

In addition to stabilizing the aggregate workload distribution as intro-
duced in Section 2.2.1, 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 sta-
tionary time-homogeneous \( M/G/1 \) queue. Based on the mini-
um number of powered-on servers \( N_t \), arrival rates \( \lambda_{0t} \), and
workload distributions \( H_{0t} (\cdot) \) of additional traffic for each time
interval \( \ell \in T \), we seek to route arrivals of requests so that the
arrival rates into each powered-on server would be time-
homogeneous (i.e., constant across time intervals) while ensur-
ing that the aggregate workload distribution is also stabilized as
the target workload distribution \( H (\cdot) \). Let \( v_{aj\ell} \) be the fraction of
arrival requests of class \( a \) routed to server \( j \) in time interval \( \ell \),
then \( v_{aj\ell} \) can be appropriately selected based on the following
time-homogeneity constraints:
\[
\sum_{j=1}^{N_t} v_{aj\ell} = 1 \quad \forall a \in A \cup \{0\} \tag{3}
\]
\[
\frac{\lambda_{0t}}{\theta_{0t}} v_{aj\ell} + \sum_{a \in A} \frac{\lambda_{at}}{\theta_a} v_{aj\ell} = \rho \phi \quad \forall j \in \{1, 2, \ldots, N_t\} \tag{4}
\]
\[
\frac{\lambda_{0t} v_{aj\ell} \hat{H}_{0t}(s) + \sum_{a \in A} \lambda_{at} v_{aj\ell} \hat{H}_{a}(s)}{\sum_{a \in A, 0} \lambda_{at} v_{aj\ell}} = \hat{H}(s) \quad \forall j \in \{1, 2, \ldots, N_t\}, s \geq 0. \tag{5}
\]
Recall that \( \rho \) is the desired traffic intensity and \( \phi \) is the tar-
goal nominal speed. Equation (3) means that all application
requests should be routed to servers; the traffic intensity of
each powered-on server would be \( \rho \) based on Equation (4);
Equation (5) ensures that the aggregate workload distribution at
each powered-on server \( j \) is approximately equivalent to the tar-
get workload distribution \( H (\cdot) \). In fact, Equation (5) is directly
derived from Equation (2) by applying \( v_{aj\ell} \) to the routing arriv-
ing requests instead of distributing then equally to \( N_t \) powered-
on servers. Based on Equations (4) and (5), the arrival rates into
each powered-on server would remain constant at \( \rho \phi \) across
time intervals, since the mean amount of workload brought by
incoming request would be \( 1/\theta \) based on Equation (5)
(i.e., \(-\hat{H}'(0) = 1/\theta \)) and thus
\[
\sum_{a \in A, 0} \lambda_{at} v_{aj\ell} = \rho \phi, \quad \sum_{a \in A, 0} \lambda_{at} v_{aj\ell} = \rho \phi \theta \quad \forall j \in \{1, 2, \ldots, N_t\}.
\]

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

### 2.2.3. Adjusting the initial condition of the time intervals

The fundamental idea for reducing energy costs 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 dis-
tribution) 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, the queue
length distribution would not be stabilized when the length of
each time interval is not long enough to reach steady-state oper-
ation. To address this problem, we add a batch of requests to
servers that are powered on afresh at the beginning of a time
interval, so that the 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 in our previous work
(Kwon and Gautam, 2015) whose main idea is adding a num-
ber of jobs sampled from the stationary queue length distri-
bution 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 a 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 Kwon and Gautam
(2015), the key benefit of the suggested approach is that per-
formance 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
a First-Come, First-Served (FCFS) service discipline. Here
our claim is that our suggested approach can be extended to
a model using a Processor Sharing (PS) service policy. Under
the PS regime, all of 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 a PS service policy. In fact, $M/G/1$ queue with PS does produce closed-form results that are identical to those of $M/M/1$ queues with FCFS discipline (Gautam, 2012). Thus, our suggested approach for time-stability can be applied to the model that uses PS instead of FCFS for its service policy. For PS service policy, the mean queue length $L$ of $M/G/1$ would be $L = \frac{\rho}{(1 - \rho)}$ 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 introduced the notion of time-stability considered in this study and suggested an approach to achieve time-stability based on a moment-matching approximation. As we previously mentioned, if we match sufficiently many moments, then the aggregate workload distribution at every powered-on server would be approximately equivalent to $H(\cdot)$, and thus we could theoretically have stabilized the queue length distribution. Moreover, for 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 (Gautam, 2012) to determine the mean queue length $L$ as follows:

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

where $C^2$ is the SCOV of the service time and $\rho$ 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 based on 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 $L$ and $\rho$ (recall that $\rho$ is the desired traffic intensity). Since $C^2$ is solely determined by the first and second moments of $H(\cdot)$ ($-\bar{H}(0)$ and $\bar{H}''(0)$, 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:

$$\begin{align*}
\lambda_{0j}v_{0j}(-\bar{H}'(0)) + \sum_{a \in A} \lambda_{at}v_{aj}(-\bar{H}'_a(0)) &= -\bar{H}'(0) \quad \forall j \in \{1, 2, \ldots, N_j\}, s \geq 0 \tag{7} \\
\lambda_{0j}v_{0j}\bar{H}''(0) + \sum_{a \in A} \lambda_{at}v_{aj}\bar{H}''_a(0) &= \bar{H}''(0) \quad \forall j \in \{1, 2, \ldots, N_j\}, s \geq 0 \tag{8}
\end{align*}$$

In Section 3, we will formulate an MIP by using the constraints (7) and (8) instead of using Equation (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 the operational decisions that correspond to decision variables $x_{ajt}$, $y_{jt}$, $v_{ajt}$, and $\phi_{jt}$ introduced in Section 1.2. The key idea is to select decision variables, $x_{ajt}$, $y_{jt}$, $v_{ajt}$, and $\phi_{jt}$ so that they not only enforce time-stability of the queue length distribution at every powered-on server but also result in the system being energy-efficient.

#### 3.1. Assumption for energy cost

To consider the energy cost of data center operation, we define a fixed energy cost $f_j$ for a powered-on server $j$ (i.e., energy cost for server $j$ in its ”idle” state) and an operating cost $c_j$ for processing jobs at server $j$ at speed $\phi_{jt}$. 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 Lin et al. (2012) and Lin et al. (2013), the switching cost mainly consists of (i) the energy used to power servers on and off; (ii) a time delay during which the server is set up; and (iii) increased wear-and-tear on the server toggling. For (i) and (ii), first, we would like to note that the length of time intervals used in our suggested model is much longer than the duration of the setup time. For example, Gandhi et al. (2010), Guenter et al. (2011), and Gandhi et al. (2012) reported that the setup times range between 20 and 200 seconds, which are very small compared to the 1-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 1-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., $y_{jt}$ and $\phi_{jt}$ for all $j \in N$ and $t \in T$) are pre-determined, as opposed to the dynamic server provisioning considered in Lin et al. (2012) and Lin et al. (2013), 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 (Chen et al., 2005; Guenter et al., 2011; Lin et al., 2012; Lin et al., 2013; Wang et al., 2015) considers the impact of server on/off cycles on the reliability of the server. For example, the recent studies Lin et al. (2012), Lin et al. (2013), and Wang et al. (2015) proposed an approach for the dynamic server provisioning model by considering a switching cost that included a server reliability cost for wear-and-tear caused by toggling servers. However, contrasting the dynamic server provisioning algorithms proposed by Lin et al. (2012), Lin et al. (2013), and Wang et al. (2015), which change server state frequently, the servers in our problem stay at the powered-on or powered-off (or sleep) state for at least an hour and do not change server state frequently; thus, the effect of toggling servers on reliability may not be significant. Moreover, as mentioned in Chase et al. (2001) and Bodik et al. (2008), powering servers off may extend the lifetime of server components. Therefore, toggling servers may not significantly affect the reliability of servers.
For time-stability, the target aggregate workload distribution $H(\cdot)$ and arrival rates of additional traffic $\lambda_{\text{ot}}$ and workload distribution $H_{\text{ot}}(\cdot)$ 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 the workload distribution of additional traffic (indexed as "0") $1/\theta_{\text{ot}}$ and $H_{\text{ot}}'(0)$. Based on the set of indices, parameters, and decision variables, we formulate an MIP optimization problem. For each time interval $\ell \in T$, our suggested optimization problem can be formulated as follows:

\begin{align}
\text{MIP-}\ell: \text{Minimize} & \quad \sum_{j \in N} (f_j y_{j\ell} + c_j \phi_{j\ell}) \quad (9) \\
\text{s.t.} & \quad \sum_{j \in N} v_{aj\ell} = 1 \quad \forall a \in A \quad (10) \\
& \quad \frac{\lambda_{aj\ell}}{\theta_{aj\ell}} v_{aj\ell} + \sum_{a' \in A} \frac{\lambda_{a'j\ell}}{\theta_{a'j\ell}} v_{aj'\ell} = \rho \phi_{j\ell} \quad \forall j \in N \quad (11) \\
& \quad \phi_{j\ell} \leq \phi_{\text{max}} y_{j\ell} \quad \forall j \in N \quad (12) \\
& \quad \frac{\lambda_{aj0\ell}(-H'_a(0)) + \sum_{a' \in A} \frac{\lambda_{a'j0\ell}}{\theta_{aj0\ell}} v_{aj'0\ell}(-H'_{a'}(0))}{\sum_{a' \in A, j \in J} \lambda_{a'j0\ell} v_{aj'0\ell}} = -H'(0) \quad \forall j \in N \quad (13) \\
& \quad \frac{\lambda_{aj0\ell}H''_a(0) + \sum_{a' \in A} \frac{\lambda_{a'j0\ell}}{\theta_{aj0\ell}} v_{aj'0\ell}H''_{a'}(0)}{\sum_{a' \in A, j \in J} \lambda_{a'j0\ell} v_{aj'0\ell}} = H''(0) \quad \forall j \in N \quad (14) \\
& \quad \sum_{a \in A} \beta_{a} x_{aj\ell} \leq b_{j} \quad \forall j \in N, \forall k \in K \quad (15) \\
& \quad v_{aj\ell} \leq x_{aj\ell} \quad \forall a \in A, \forall j \in N \quad (16) \\
& \quad v_{aj\ell} \geq 0, x_{aj\ell} \in \{0, 1\}, y_{j\ell} \in \{0, 1\}, \phi_{j\ell} \geq 0 \quad \forall a \in A, \forall j \in N. \quad (17)
\end{align}

In the above formulation, the objective function (9) is the total energy cost in a cycle, which consists of the fixed cost and operating cost that we wish to minimize. We define the cost function for energy usage as a combination of a fixed energy cost and the operating cost using a model of power usage of typical servers adopted by Lin et al. (2013). 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 by Raghavendra et al. (2008), Gandhi et al. (2009), and Kusic et al. (2009) 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 $j$ is $\rho$, as 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 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 $\phi$, 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 a time-varying $M_j/R_i/1$ queue is stochastically identical to that of time-homogeneous $M/G/1$ queue when we adjust the processing speed (service rate) responding to a time-varying arrival rate based on constraint (4). In addition, constraint (15) limits resource $k$ to $b_{j}$ across classes assigned to server $j$ with requirements $\beta_{a}$ for class $a$, and constraint (16) ensures that if class $a$ is not assigned to server $j$, then no fraction of arriving requests is 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 $\ell \in T$, $j \in N$, and $a \in A$) can be determined by solving the above MIP-$\ell$ problem separately for each time interval $\ell \in T$. Although we solve the optimization problem to determine operational decisions independently for each time interval $\ell \in T$ (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 the overall problem into 24 MIP-$\ell$ problems) in order to achieve time-stability. This is the key benefit of our suggested approach, as 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-$\ell$ problems, each of the MIP-$\ell$ problems is indeed NP-hard (we can simply show that a well-known NP-hard Capacitated Facility Location Problem can be reduced to our suggested MIP-$\ell$ 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

In Sections 2 and 3, we introduced the notion of time-stability and suggested an approach to stabilize the queue length distribution of each powered-on server. Specifically, we derived the constraints and formulated an MIP problem to stabilize the mean queue length. Moreover, in this section, we will show that a 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, SLAs and their violations are also defined based on waiting times. Thus, a 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 the arrival rates in the online supplement. In this case, even if the queue length distribution is stabilized, the waiting time distribution would not be stabilized, as 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 \( T \), let \( X_{at} \) be the waiting time of class \( a \) in time interval \( \ell \) with Cumulative Distribution Function (CDF) \( W_{at}(\cdot) \) and \( X_{jt} \) be the waiting time of applications that are served by server \( j \) in time interval \( \ell \) with CDF \( W_j(\cdot) \). Then \( X_{at} \) can be defined as \( X_{at} = X_{jt} \) with probability \( v_{ajt} \), and the LST of the waiting time distribution for class \( a \) in time interval \( \ell \), \( \bar{W}_{at}(s) \) can be defined as

\[
\bar{W}_{at}(s) = \sum_j v_{ajt} \bar{W}_{jt}(s),
\]

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

\[
\bar{W}_{jt}(s) = \frac{(1 - \rho)s}{s - \lambda + \lambda \bar{S}_{jt}(s)}.
\]

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

\[
\bar{W}_{at}(s) = \sum_j v_{ajt} \bar{W}_{jt}(s) = \sum_j v_{ajt} \frac{(1 - \rho)s}{s - \sum_a v_{ajt} \lambda_{at} + \sum_a v_{ajt} \lambda_{at} \bar{H}(s/\phi_{jt})}.
\]

Moreover, the mean waiting time of class \( a \) in time interval \( \ell \) is

\[
\bar{X}_{at} = \sum_j v_{ajt} \bar{X}_{jt} = \sum_j \left( \frac{\lambda_{jt} E[Z_{jt}]}{2(1 - \rho)} \right) = \sum_j v_{ajt} \left( \frac{\sum_a v_{ajt} \lambda_{at}}{2(1 - \rho)} \frac{(1 + C^2)(E[Y])^2}{\phi_{jt}^2} \right).
\]

where \( C^2 \) is the 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. Furthermore, if we do not consider speed scaling and enforce that the processing speed of powered-on servers is a constant across time intervals such that \( \phi_{jt} = \phi \), \( \forall j \in N, \ell \in T \), 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 the aggregate workload distribution and queue length distribution for each powered-on server, an upper bound on the mean waiting times of each class \( a \), \( \tau_a \) (e.g., \( \bar{X}_{at} \leq \tau_a \), \( \forall a \in A, \forall \ell \in T \)) can be easily obtained based on the desired mean queue length \( \bar{L} \), which was introduced in Section 2.3, as follows:

\[
\tau_a = \frac{E[Y] \bar{L}}{\phi_{a \min}} \text{ where } \phi_{a \min} = \min(\phi_{jt}, j \in N_{a \text{on}}, \ell \in T).
\]

In other words, the upper bound \( \tau_a \) 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 \( \phi_{a \min}^j \) among the powered-on servers. Consequently, an upper bound on the mean waiting times of the overall system across time intervals \( r \) can be defined as

\[
\tau = \max\{\tau_a, a \in A\}.
\]

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

\[
r \phi_{j \max} \leq \phi_j \quad \forall j \in N,
\]

where \( r \) is the bound ratio of the minimum speed to the maximum speed of the 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 a performance bound on the mean waiting time. For the numerical evaluation, we selected two test instances: 10 servers with 20 applications \( (|N| = 10, |A| = 20) \) and 20 servers with 40 applications \( (|N| = 20, |A| = 40) \) from Gallego Arrubla et al. (2013). Recall that, although data centers generally operate hundreds or thousands of servers, considering that in most data centers applications of a single client are clustered among servers for security and confidentiality, test instances of 10 or 20 servers
are large enough to show that our suggested approach can be applied successfully in practice. We modeled the arrival process of each application's request as a non-homogeneous Poisson process with time-varying arrival rates $\lambda_{\alpha}$ for 24 1-hour time intervals (i.e., 24 equally spaced time intervals for a 1-day cycle) such that $\ell \in \mathcal{T} = \{1, 2, \ldots, 24\}$. In Figs. 2(a) and 2(b), we plot the arrival rates of both test instances (20 applications and 40 applications, respectively) across 24 1-hour time intervals. Also, for the workload distribution of each application $a$, $H_a(\cdot)$, we chose a uniform distribution with mean workload $1/\ell$, and SCOV $C^2_a$. We also assumed 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 defined fixed energy costs for powered-on server $j$ as being much higher than the unit operation costs of server $j$, $c_j$, considering the fact that energy cost of servers is dominated by fixed costs as mentioned in Barroso and Holzle (2007). In this study, we solved the suggested MIP problem using CPLEX 12.6 Concert Technology on an Intel Core i7-3740QM 2.70 GHz with a 16 GB memory, and we used Java to develop a discrete-event simulation framework. We ran simulation experiments by using decision variables, $x_{a,j,t}$, $y_{j,t}$, $\phi_{j,t}$, and $u_{a,j,t}$, which were 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 computation time obtained by solving MIP-\(\ell\) across time intervals $\ell \in \mathcal{T}$ for the two test instances. Note that we obtained 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

![Figure 2. Time-varying arrival rates across 24 1-hour time intervals: (a) arrival rates of 20 applications and (b) arrival rates of 40 applications.](image-url)

| Table 2. Objective function value and CPU Time: 10 servers and 20 applications and 20 servers and 40 applications |
|---|---|---|---|---|---|---|
| Time | Objective function (seconds) | CPU time (seconds) | Gap (%) | Time | Objective function (seconds) | CPU time (seconds) | Gap (%) | Time | Objective function (seconds) | CPU time (seconds) | Gap (%) | Time | Objective function (seconds) | CPU time (seconds) | Gap (%) |
| 1 | 282.00 | 694.49 | 768.17 | 3603.86 | 13.86 | 2 | 255.00 | 148.59 | 969.50 | 3604.75 | 0.74 | 3 | 282.00 | 1534.94 | 1029.33 | 70.78 | 0.00 | 4 | 309.00 | 408.02 | 1037.50 | 3604.86 | 0.91 | 5 | 342.00 | 128.34 | 1099.67 | 100.06 | 0.00 | 6 | 280.00 | 1076.35 | 1029.33 | 1233.63 | 0.00 | 7 | 288.00 | 163.40 | 957.67 | 3603.56 | 6.51 | 8 | 315.00 | 2007.39 | 953.33 | 3610.75 | 6.08 | 9 | 317.00 | 843.70 | 833.67 | 3607.44 | 12.76 | 10 | 356.00 | 663.96 | 868.67 | 3603.47 | 10.65 | 11 | 356.00 | 2455.92 | 822.67 | 3606.51 | 12.17 | 12 | 345.00 | 915.66 | 828.83 | 3605.96 | 13.21 | 13 | 318.00 | 180.09 | 916.83 | 3609.30 | 2.35 | 14 | 310.00 | 245.81 | 853.17 | 3605.03 | 2.13 | 15 | 295.00 | 305.95 | 689.00 | 3602.81 | 8.75 | 16 | 256.00 | 196.19 | 757.00 | 3602.22 | 4.83 | 17 | 256.00 | 203.53 | 690.33 | 3608.10 | 11.70 | 18 | 260.00 | 121.98 | 647.17 | 3603.84 | 6.89 | 19 | 295.00 | 391.58 | 625.00 | 3607.68 | 11.81 | 20 | 273.00 | 170.18 | 614.50 | 3608.63 | 10.31 | 21 | 266.00 | 89.03 | 620.67 | 3607.50 | 10.87 | 22 | 263.00 | 142.93 | 613.50 | 3605.06 | 9.50 | 23 | 263.00 | 103.35 | 627.83 | 3613.53 | 15.24 | 24 | 255.00 | 214.84 | 671.00 | 3621.70 | 11.45 |
report the objective function values of the best feasible solution and MIP gap for 3600-, 7200-, and 10 800-second time limits for the instance of 20 servers and 40 applications.

5.2. Analysis of time-stability

First, 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 selected 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 selected the desired traffic intensity as \( \rho = 0.8 \). Figures 3(a) and 4(a) depict the mean queue length of powered-on servers across the 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 with the time-varying arrival rates shown in Figs. 2(a) and 2(b). 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 obtained using the decision variables determined by solving the modified MIP problem that uses the following constraint (22) instead of time-homogeneity constraints (11), (13), and (14):

\[
\sum_{a \in A} \lambda_{at} \theta_{ajk} \leq \rho \phi_{jk} \quad \forall j \in N.
\] (22)

The above constraint (22) ensures that the average workload that arrives at every powered-on server must not exceed the desired traffic intensity \( \rho \) for fair comparison of our suggested approach. As shown in Fig. 3(b) and Fig. 4(b), the mean queue length would not be stabilized without using time-homogeneity constraints as compared with Fig. 3(a) and Fig. 4(a). Also, we compared 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 to stabilize the mean queue length, as well as provide performance guarantees on the mean waiting time. As we can see in Figs. 5(a) and 5(b), the 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, a 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 selected 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 contrasted the mean waiting times obtained by using constraint (21) with the results obtained without using constraint (21) for both test instances. Figure 6(a) and Fig. 7(a) depict the mean waiting times and the performance bounds of both test instances, respectively. Note that the mean waiting time is strictly bounded by \( \tau \), which can be derived as Equation (20) as opposed to the results obtained without using constraint (21) shown in Fig. 6(b) and Fig. 7(b).

5.4. Benchmark

In addition, we compare our approach against the algorithm proposed by Lin et al. (2013) as a benchmark. Lin et al. (2013) considered a data center optimization problem and proposed an off-line algorithm to determine the number of active servers to minimize the 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. (2009) and Rao et al. (2010). Note that Lin et al. (2013) implemented a dispatching rule that delivers an equal amount of workload to each powered-on server (i.e., load balancing). Note that Lin et al. (2013) suggested a model that uses quasi-steady-state approximations (i.e., the metrics are piecewise constant for each time interval, long enough for the system to reach a steady state) to define QOS constraints for a non-stationary system, and it is worthwhile pointing out that our suggested approach can provide a provable performance bound for time-varying and transient systems. Since Lin et al. (2013) 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. (2013) proposed a QOS constraint that the average waiting time is bounded by certain thresholds based on \( M/G/1 \) processor sharing queue, and we redefined their constraint by using multiclass \( M/G/1 \) queue with FCFS (Gautam, 2012) for the heterogeneous workload. Also, since our suggested MIP problem does not consider switching costs (i.e., cost for powering a server on and off), we modified our model by adding the following constraints (23) and (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 the cost for powering on/off, we define additional decision variables, \( u^m_{j\ell} \) (e.g., \( u^m_{j\ell} = 1 \) if server \( j \) is turned
on at the beginning of time interval $\ell$) and $u_{jl}^{\text{off}}$ (e.g., $u_{jl}^{\text{off}} = 1$ if server $j$ is turned off at the end of time interval $\ell - 1$). Then we can define the constraints to determine variables $u_{jl}^{\text{on}}$ and $u_{jl}^{\text{off}}$ as follows:

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

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

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

$$\sum_{j \in N} \sum_{\ell \in T} (c_{jl}^{\text{on}} u_{jl}^{\text{on}} + c_{jl}^{\text{off}} u_{jl}^{\text{off}}), \quad (25)$$

where $c_{jl}^{\text{on}}$ and $c_{jl}^{\text{off}}$ are the 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) and (24)) and set the 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 the “idle” power cost. In Figs. 8(a) and 8(b), we compare the total energy costs of our approach against that of Lin et al. (2013) by setting the thresholds for the mean waiting time to be equal to the upper bound obtained by our suggested approach under the same traffic intensity. As shown in Figs. 8(a) and 8(b), the total energy cost of our suggested approach is smaller than that of the approach in Lin et al. (2013) 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 as can be derived by our suggested approach.

6. Concluding remarks and future work

In this article, 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) the aggregate workload distribution, based on a moment-matching approximation, and (ii) the 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 about sizing, assignments, routing, and speed scaling to minimize energy costs while considering time-stability. In fact, we can obtain time-stability of the queue length distribution by sufficiently matching 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 moments. In addition, based on a time-stable queue length distribution, the waiting times of applications are easily controlled by obtaining bounds on the 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 that for future work.

We acknowledge that our approach is purely based on historical information and further work needs to be done before it can be implemented in practice. We believe that the 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. Furthermore, 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 online algorithm for such real-time control problems in the future.

Acknowledgements

We thank the anonymous reviewers and editors for their insightful comments that led to significant improvements in the content and presentation of this work. We are grateful to Dr. Butenko for his help in showing the complexity of the MIP.

Funding

This material is partially supported by the AFOSR under contract no. FA9550-13-1-0008.

Notes on contributors

Soongeil Kwon received B.S. and M.S. degrees in Information and Industrial Engineering from Yonsei University, Seoul, Korea, in 2005 and 2007, respectively, and he is currently pursuing a Ph.D. degree in Industrial and Systems Engineering at Texas A&M University, College Station, Texas. His research interests include stochastic programming, stochastic process and control, and simulation. Research domains include but are not limited to (i) energy-aware operations; (ii) demand-side power procurement; (iii) renewable energy and energy storage; and (iv) smart grid.

Natarajan Gautam is a Professor in the Department of Industrial and Systems Engineering at Texas A&M University with a courtesy appointment in the Department of Electrical and Computer Engineering. Prior to joining Texas A&M University in 2005, he was on the Industrial Engineering faculty at Penn State University for 8 years. He received his M.S. and Ph.D. degrees in Operations Research from the University of North Carolina at Chapel Hill and his B.Tech. from the Indian Institute of Technology, Madras. His research interests are in the areas of modeling, analysis, and performance evaluation of stochastic systems with special emphasis on optimization and control in computer, telecommunication, and information systems. He is an Associate Editor for the INFORMS Journal on Computing, IIE Transactions, and OMEGA.

References

Barnes, J.A. and Meeli, R. (1997) A stationary Poisson departure process from a minimally delayed infinite server queue with non-stationary Poisson arrivals. Journal of Applied Probability, 34(3), 767–772.

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

Bertini, L., Leite, J. and Miosé, 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.

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. Technical Report UCB/EECS-2008-127, University of California at Berkeley. Berkeley, CA.

Brown, R. (2008) Report to Congress on server and data center energy efficiency: Public Law 109-431. Lawrence Berkeley National Laboratory, Berkeley, CA.

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. ACM SIGOPS Operating Systems Review, 35, 103–116.

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

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.

Foley, R.D. (1986) Stationary Poisson departure processes from non-stationary queues. Journal of Applied Probability, 23(1), 256–260.

Gallego Arrubla, J.A., Ko, Y.M., Polansky, R.J., Pérez, E., Ntimo, 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.

Gandhi, A., Chen, Y., Gmacht, D., Arlitt, M. and Marwah, M. (2011) Minimizing data center SLA violations and power consumption via hybrid resource provisioning, in Proceedings of the Green Computing Conference and Workshops, IEEE Press, Piscataway, NJ, pp. 1–8.

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.

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

Gandhi, A., Harchol-Balter, M., Kozuch, M., et al. (2012) Are sleep states effective in data centers? in Proceedings of the Green Computing Conference, IEEE Press, Piscataway, NJ, pp. 1–10.

Gautam, N. (2012) Analysis of Queues: Methods and Applications, CRC Press, Boca Raton, FL.

Gmach, D., Rolia, J., Cherkasova, L. and Kemper, A. (2007) Workload analysis and demand prediction of enterprise data center applications, in Proceedings of the 10th International Symposium on Workload Characterization, IEEE Press, Piscataway, NJ, pp. 171–180.

Guenther, B., Jain, N. and Williams, C. (2011) Managing cost, performance, and reliability tradeoffs for energy-aware server provisioning, in Proceedings of the INFOCOM, IEEE Press, Piscataway, NJ, pp. 1332–1340.

Hamilton, J. (2008) Cost of power in large-scale data centers. Available from http://perspectives.mvdirona.com. Accessed November 28, 2008.
Koomey, J. (2011) Growth in data center electricity use 2005 to 2010. A report by Analytical Press, Oakland, CA, completed at the request of The New York Times, p. 9.

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.

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

Lin, M., Liu, Z., Wierman, A. and Andrew, L.L. (2012) Online algorithms for geographical load balancing, in Proceedings of the Green Computing Conference, IEEE Press, Piscataway, NJ, pp. 1–10.

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

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

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. ACM SIGMETRICS Performance Evaluation Review, 40, 175–186.

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.

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

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 Proceedings of the INFOCOM, IEEE Press, Piscataway, NJ, pp. 1–9.

Van, H.N., Tran, F.D. and Menaud, J.-M. (2009) SLA-aware virtual resource management for cloud infrastructures, in Proceedings of the Ninth International Conference on Computer and Information Technology, Volume 1, IEEE Press, Piscataway, NJ, pp. 357–362.

Vogels, W. (2008) Beyond server consolidation. ACM Queue, 6(1), 20–26.

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.

Whitt, W. (2015) Stabilizing performance in a single-server queue with time-varying arrival rate. Queueing Systems, 81(4), 341–378.

Xu, D. and Liu, X. (2012) Geographic trough filling for internet datacenters, in Proceedings of INFOCOM, IEEE Press, Piscataway, NJ, pp. 2881–2885.

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 Proceedings of INFOCOM, IEEE Press, Piscataway, NJ, pp. 1431–1439.

Zhu, Z., Bi, J., Yuan, H. and Chen, Y. (2011) SLA-based dynamic virtualized resources provisioning for shared cloud data centers, in Proceedings of the International Conference on Cloud Computing, IEEE Press, Piscataway, NJ, pp. 630–637.