Time-Stable Performance in Parallel Queues with Non-Homogeneous and Multi-class Workloads

Soongeol Kwon, Natarajan Gautam, *Senior Member, IEEE,*

***Abstract*—Motivated by applications in data centers, we consider a scenario where multiple classes of requests arrive at a dispatcher at time-varying rates which historically has daily or weekly patterns. We assume that the underlying environment is such that at all times the load from each class is very high and a large number of servers are necessary which, for example, is fairly common in many data centers. In addition, each server can host one or more classes. Design, control and performance analysis under such heterogeneous and transient conditions is extremely difficult. To address this shortcoming we have suggested a holistic approach that includes a combination of sizing, assignment, and routing in an integrated fashion. Our proposed approach decomposes a multi-dimensional and non-stationary problem into a one-dimensional, simpler and stationary one, and achieves time-stability by introducing an insignificant number of dummy requests. Based on time-stability, our suggested approach can provide performance bounds and guarantees for time-varying and transient system. Moreover, we can operate the data centers in an energy-efficient manner via suggested approach.**

***Index Terms*—Data center operations, non-homogeneous and multi-class workloads, parallel server queues, queueing analysis, simulation, time-stability.**

I. Introduction

INTERNET applications hosted by data centers are characterized by time-varying workloads with significant variations and uncertainties over multiple time scales (Menasce et al. [1]). Under such workloads it is challenging to appropriately manage resources to conserve energy consumption which is skyrocketing (see report [2]) while providing a reasonable level of performance and meeting service level agreement (SLA) (Chen et al. [3]). As explained and documented in Hamilton [4] and Koomey [5], data centers consume a phenomenal amount of power similar to what an entire city would use, albeit inefficiently; Barroso and Holzle [6] indicated that servers operate most of the time between 10 and 50 percent of their maximum utilization levels, and Vogels [7] reported that many of the large analyst firms estimate that resource utilization of 15 to 20 percent is common for operation of data centers. In addition recent studies, Abts et al. [8], Lin et al. [9], Feller et al. [10], Gandhi et al. [11], Lee and Zomaya [12] and Wang et al. [13] also mentioned low utilization of data centers and proposed approach for energy efficiency.

While there are tremendous opportunities to conserve energy consumption in data centers, due to the inherent uncertainty and variability in the loads, developing provably effective methods to manage resources in data centers has been a challenge. To address this shortcoming, a number of techniques have been proposed and most of these studies focus on developing algorithms to determine the right size of servers for non-stationary workloads. In particular, Singh et al. [14] suggested a mix-aware dynamic provisioning technique using the k-means clustering algorithm to determine workload mix, Gandhi et al. [15] presented an approach to correctly allocate resources in data centers such that SLA violations and energy consumption are minimized and Lin et al. [9] proposed a new on-line algorithm for dynamic right sizing in data centers motivated by optimal offline solution for energy cost. Also Gandhi et al. [11] studied dynamic capacity management for multi-tier data centers, Wang et al. [16] provided an analytic framework that captures non-stationarities and stochastic variation of workloads for dynamic re-sizing in data centers and Gallego et al. [17] 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 of workloads. It is worthwhile pointing out that most of aforementioned approaches [15], [9], [11], [16] and [17] use quasi-steady state approximations, i.e. the metrics are piecewise constant for time periods long enough for the system to reach steady-state.

Although the challenges for right-sizing in data centers for non-stationary workloads have received significant attention, the problem of achieving time-stability over time-varying workloads has not been effectively addressed. Achieving time-stability is essential for a non-homogeneous system because it enables the system to provide guaranteed quality of service. For example, one could compute the tail probability of sojourn times and probabilistically guarantee an incoming request for an appropriate SLA. Moreover, by stabilizing a non-homogeneous system, it is possible to effectively design and analyze the system and perform monitoring and control based on time-stability. In the context of data centers, time-stability has received little attention, although there have been some research studies in the queueing area. Foley et al. [18] and Barnes et al. [19] showed that the departure process from the $M_t/G_t/\infty$ queue can be made stationary. There is another body of literature which provides algorithms to determine appropriate staffing levels for call centers. Feldman et al. [20] proposed a simulation-based iterative-staffing algorithm for time-stable delay probability, and Liu and Whitt [21] 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.

The authors are with the Department of Industrial and Systems Engineering, Texas A&M University, College Station, TX, 77843-3131 USA (e-mail: soongeol@tamu.edu, gautam@tamu.edu.)

In our case, we have modeled data centers as a system of multiple parallel single-server queues, and considered a scenario where multiple classes of requests arrive at a dispatcher at time-varying rates that historically has daily or weekly patterns. For such a scenario, we develop an approach to simultaneously determine sizing, assignment and routing appropriately so that the resulting system performance is homogeneous over time and uncertainty is controlled despite the fact that the parameters can vary extremely quickly, not allowing the system to reach steady-state. Therefore, no matter how fast arrival rates vary, our approach can provide time-stable distribution of the number of requests in the system as well as sojourn times, and this is the crucial difference between our approach and other sizing algorithms dealing with time-varying workloads.

Objective of our study is to address needs of practitioners, such as providing performance guarantees while being prudent about energy consumption. Our suggested approach provides an analytic framework simplifying a multi-dimensional, transient and non-stationary problem by decomposing into individual simpler stationary ones based on the strategies for sizing, assignment, and routing in an integrated fashion which has seldom been implemented jointly. The main contribution of our study is providing performance guarantees and bounds which can be simply derived based on stationary analysis for time-varying and transient system while considering energy efficiency. The remainder of the paper is organized as follows: Section II describes the detailed scenario for the problem and various options for decisions and control that we would consider; Section III proposes a sequential procedure to determine assignment, sizing, and routing for the suggested scenario; Section IV introduces an additional insight regarding assignment strategies; Section V describes the notion of time-stability and introduces our approach to obtain time-stability; Section VI discusses details of time-stability including extension and limitations; Section VII illustrates the experimental results to support our claims; and Section VIII presents conclusions and future research directions.

II. Analytical Framework

This section provides a detailed scenario for multi-class and non-homogeneous requests to servers that are considered in this paper followed by a stochastic model for the scenario. We then briefly state the asymptotic scaling where the arrival rates and number of servers are scaled. This section concludes with a description of various options for decisions and control such as assignment, sizing and routing.

A. Scenario and Problem Description

We have considered a system using a large number of servers with each server having its own queue with an infinite waiting area, and the servers and their queues are arranged in a parallel fashion with dispatcher depicted in Figure 1. Considering that today's data centers have hundreds or thousands of servers to process huge amount of traffic for cloud computing, an architecture with multiple servers and a single queue results in significant communication overload to update the state information of each server to dispatch requests from queue. Therefore, multiple parallel single server queue system where dispatcher routes incoming request to servers based on load balancing algorithm is indeed appropriate to design data centers. This is corroborated by recent studies, Chen et al. [22], Gandhi et al. [23] and [24] which used multiple parallel queue system to analyze data center operations. Note that we also assume that the servers are identical, however we would like to point out that the analysis can be extended to heterogeneous servers as well.

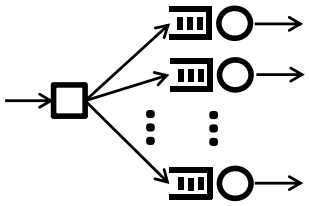


Fig. 1. System of multiple parallel single server queues

The servers process requests that belong to multiple classes. The requests that are part of a class are stochastically identical with a common non-homogeneous arrival process and also the amount of work they bring. It is assumed that a server can host multiple classes of requests and every class of request can be hosted on multiple servers. We have considered a scaled system where the arrival rate for every class is so large that several servers would need to be operational to respond to the requests of that class alone. However, the arrival rate for every class is time-varying both deterministically and stochastically. The variability is frequent-enough that in the general case one cannot expect the system to reach steady state before arrival rates change. For such a multi-class, transient and non-homogeneous system, our intent is to effectively manage resources to ensure time-stability while being mindful of energy costs. The following are issues that are considered explicitly for time-stability:

1) *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. One focus is to study the impact of host-server assignment on performance. We assume that there is no direct cost per se for the assignments as well as no costs for switching assignments.
2) *Sizing*: Each server could be dynamically powered on or off. Naturally more servers would be "on" during peak periods than during lean periods. Significant energy cost savings can be attained by powering servers off. However, this analysis neither considers switching costs (from on to off and vice versa) nor considers 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.
3) *Routing*: There is a dispatcher that is responsible for routing arriving requests to one of the queues that not

only can serve the request but also has a server that is powered on. A key assumption is that the dispatcher cannot observe the real-time state of any of the servers (however the dispatcher knows whether a server is on or off, and what classes it hosts; as we will see in the model subsequently, these do not vary in real time).

B. Model and Notation

For the problem described in the previous sub-section, here we set the notation and develop a stochastic model that would form the inputs to our analysis. We consider a system of N parallel queues and each queue is served by a single server that could be dynamically powered on or off. The dispatcher is responsible for routing arriving requests to one of the queues that not only serves the request but also has a server that is powered "on." An arriving request belongs to one of multiple classes in a discrete set $\mathcal{A}$ denoting a set of applications. The amount of work a class a (for all $a \in \mathcal{A}$) request brings is independent and identically distributed (IID) according to general distribution $H_a(\cdot)$ with mean $1/\theta_a$ and squared coefficient of variation (SCOV) C_a^2. Recall that the SCOV is the ratio of the variance to the square of the mean. For ease of exposition, as a probability distribution that can handle SCOV values greater than, equal to as well as less than one for analysis, we selected a Coxian-2 distribution for workload. Essentially, a Coxian-2 distribution is either a sum of two independent exponential distributions (with parameter $\theta_{a,1}$ and $\theta_{a,2}$) with probability p_a, or just exponentially distributed (with parameter $\theta_{a,1}$) with probability $1 - p_a$. We chose the units of $1/\theta_a$ to be kB (kilo-Bytes) with the understanding that the analysis would not be affected in any way by choosing other units.

Requests of each class arrive to the dispatcher 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. [25], Lin et al. [26], Liu et al. [27] and Lin et al. [9]). Using that assumption we modeled each cycle as divided into a set of phases $\mathcal{T}$ so that in each phase $\ell \in \mathcal{T}$, the arrival rate for every class $a \in \mathcal{A}$ remains a constant $\lambda_{a,\ell}$ per second. Although the intention is to convey the richness of the model (in that the analysis would work in such a general fashion), in practice one would typically choose something like the set of all disjointed 5-minute intervals (or hourly intervals) in a day (or a week) as the set of phases $\mathcal{T}$.

Let ϕ be a target operating speed of a powered on server in units of kB/s (kilo-Bytes per second). Therefore, a class a (for some $a \in \mathcal{A}$) arrival brings a random amount of workload W_a kB and is routed to an idle server that is capable of serving class a requests, then the service time (if all the processor capacity is allocated to this arrival) would be W_a/ϕ seconds with mean $E[W_a]/\phi = 1/(\theta_a\phi)$ and SCOV C_a^2. Note that the SCOV of service times is unaffected by the speed of service. At this time, we assume ϕ remains a constant. One easy way to accommodate heterogeneous servers is to have them all operate at ϕ (since the jobs are assumed to be CPU intensive).

In addition the analysis in this paper uses an asymptotic approach. In particular, we jointly scaled up the arrival rates and the number of servers so that together they approach infinity. Thus, we assume that we can write down for all $a \in \mathcal{A}$ and $\ell \in \mathcal{T}$

$$\lambda_{a,\ell} = N\alpha_{a,\ell}$$

where $\alpha_{a,\ell}$ is the normalized arrival rate, and study a sequence of systems by letting $N = 1, 2, \ldots,$ which is similar in spirit to the scaling in Liu et al. [21]. However, this is not the traditional fluid or diffusion limit. All we have is that at any time there is a total of N servers (some powered on and the rest powered off) and class a requests arrive at rate $\lambda_{a,\ell} = N\alpha_{a,\ell}$, then we scale N. The next section describes how to tackle the aforementioned issues in a sequential manner.

III. Sequential Decision Procedure

As described in Section II, our objective is to consider issues regarding assignment, sizing, and routing for the suggested scenario. These decisions are made at different time-granularities. Specifically, the assignments are made more-or-less one time, although it is assumed that at the beginning of each phase $\ell \in \mathcal{T}$ the assignments can be changed for some servers, possibly (but not necessarily) using virtual migration. We assume that sizing is done at the beginning of each phase $\ell \in \mathcal{T}$. In addition, there are real-time issues such as routing which is determined at every request arrival. The decision to be made is to determine the server to which an arriving request would be routed with conditions that (i) the server is powered on, and (ii) the server has been assigned the class of application that arrives.

A. Assignment

For each phase $\ell \in \mathcal{T}$ we consider two alternate extreme assignments for analysis:

- *all classes to all servers* (*pooled*) assignment
- *one class to one server* (*dedicated*) assignment

In Section V we will show that time-stability can be obtained by controlling non-homogeneous traffic based on assignment strategies. In fact it is possible to achieve time-stability by using *dedicated assignment*. Also we will introduce an additional insight about performance comparison between *dedicated assignment* and *pooled assignment* in Section IV.

B. Sizing

As described earlier, the objective is to provide time-stability while being mindful of saving energy. One of the greatest savings in energy costs results from powering servers off (or sending them to sleep states in more modern servers). Since the workload varies from phase to phase, we have evaluated the number of servers to be powered "on" in each phase, and appropriately power on or off the right number of servers. It is also assumed that there is an ample number of servers available, therefore running out of servers is out of the question. In fact, that is a reasonable assumption considering

how poorly utilized some of the servers are, as the data centers are typically well over-provisioned. Recall that N is the total number of servers available. Based on the two alternate assignments described in the previous section, we have:

- *pooled assignment*: All applications assigned to all servers; let N_l be the number of servers powered "on" in phase ℓ, $\forall\ \ell \in \mathcal{T}$
- *dedicated assignment*: Only one application assigned to one server; let $N_{a,\ell}$ class a servers be powered on in phase ℓ, $\forall\ \ell \in \mathcal{T}$ and $a \in \mathcal{A}$.

We have considered a simple strategy of using enough servers so that the average load on servers that are powered on remains constant over time as well as across servers (the latter is indeed typical in load-balancing but not the former). To determine N_ℓ and $N_{a,\ell}$, we defined ρ as a desirable traffic intensity (ρ is dimensionless) for any server that is powered on during interval ℓ. While determining the number of servers to keep the energy consumption low, we aim to create enough residual capacity for unforeseen surges by restricting the utilization of each server to be ρ. In addition we control non-homogeneous traffic in a time-homogeneous fashion by implementing ρ into sizing algorithm defined as below. We will show how ρ can be used to achieve time-stability in Section V. We select the number of "on" servers as follows:

- *Dedicated assignment*: Only one application assigned to one server
$$N_{a,\ell} = \left\lceil \frac{1}{\rho\phi}\frac{\lambda_{a,\ell}}{\theta_a} \right\rceil \tag{1}$$
- Pooled assignment: All applications assigned to all servers
$$N_\ell = \left\lceil \frac{1}{\rho\phi}\sum_{a\in\mathcal{A}}\frac{\lambda_{a,\ell}}{\theta_a} \right\rceil \tag{2}$$

for all $\ell \in \mathcal{T}$ and $a \in \mathcal{A}$. Note that under asymptotic scaling $N \to \infty$,

$$\left\lceil \frac{1}{\rho\phi}\frac{\lambda_{a,\ell}}{\theta_a} \right\rceil \to \frac{\lambda_{a,\ell}}{\rho\phi\theta_a} \text{ and hence } \sum_{a\in\mathcal{A}} N_{a,\ell} \to N_\ell. \tag{3}$$

In such a way, the total number of servers powered on in any phase would be identical for both *pooled assignment* and *dedicated assignment*. By determining the size of powered-on servers based in Equation (1), each powered-on server is assigned a desirable traffic intensity ρ in either case. According to the above sizing rules, if it is necessary to power on more servers between successive phases, we randomly select candidate servers among the powered-off servers and power them on at the beginning of a time phase. Also, to power off servers we randomly select the powered-on servers and power them off at the end of a time phase. In this case if selected server is not idle, then we set state of server as "to be off" and do not assign any requests to those servers. We will wait until selected servers complete service for the remaining requests and power off when those servers become idle. Note that those requests remaining in "to be off" servers will also have the same sojourn time distribution since under a first-come-first-served (FCFS) the sojourn times are not affected by arrivals that come later.

C. Routing

In the sequential consideration, once the assignment of classes to servers and the number of servers to be powered "on" are made for each phase $\ell \in \mathcal{T}$, the next issue is to determine the routing strategy for the dispatcher. We assume that the dispatcher sends incoming requests to servers without information of real-time states of the queues in terms of number of jobs or amount of workload. However, we assume that the dispatcher knows the assignment of classes to servers as well as whether a server is powered on or off. In that light two routing policies are considered:

- Round-robin routing: The dispatcher routes job to queues with powered-on servers in a cyclical fashion. This is straightforward in the *pooled assignment* case, while round-robin is done within a class for *dedicated assignment*.
- Bernoulli routing: The dispatcher routes jobs to queues with eligible servers in a random fashion. In the pooled assignment case, in phase ℓ (for any $\ell \in \mathcal{T}$) select any of the N_ℓ servers with probability $1/N_\ell$ and route to that server. For the *dedicated case*, if the arriving job belongs to class a, then the dispatcher selects one of the $N_{a,\ell}$ servers with equal probability.

Harchol-Balter et al. [28] showed that round-robin routing results in better performance than Bernoulli routing. Clearly, other policies such as join the shortest queue and join the least workload queue would perform better, but they require real-time state information (which is assumed inappropriate for large-scale data centers setting). It is worthwhile noting that the round-robin policy works better because the dispatcher selects the queue which was the least recently selected (among candidate queues), and that queue naturally is also the one with the smallest expected number of jobs and smallest expected workload. We will continue to use both round-robin and Bernoulli policies for load balancing, although it is fairly clear that round-robin results in better performance. One of the reasons for continuing to use the Bernoulli policy is the convenience in analytic models, especially to obtain insights.

IV. Additional Insight: dedicated is better than pooling

This section describes an additional insight regarding assignment strategies based on our analytical framework. In general, because of the benefits of pooling resources mentioned in the literature, the intuition is that performance would be better when we assign as many applications as possible to a server. However, based on two alternate assignments defined in Section III-A we will show that *dedicated assignment* would be better. Although we have the same number of "on" servers for both *dedicated assignment* and *pooled assignment* in each time period, the queue lengths (or the sojourn times) of overall system would be higher when we use *pooled assignment* than use *dedicated assignment*. Consider a single server that is always on with time-homogeneous arrivals, i.e. $\lambda_{a,\ell}$ does not vary with ℓ for all $a \in \mathcal{A}$ and i.e. $\lambda_{a,\ell} = \lambda_a\ \forall \ell \in \mathcal{T}$. This may appear strange given that we started the article with non-homogeneous arrivals, however subsequently we will

show that this setting is in fact what is realized in the main problem in Section V-B1. It is also assumed that the servers are identical. Consider two cases for the assignments mentioned above, *dedicated assignment* or *pooled assignment*. Recall that in either case, each powered-on server faces the same traffic intensity of ρ when we determine the number of servers to be powered on according to Equation (1) for *dedicated assignment* and Equation (2) for pooled assignment.

Theorem 1: If $\mathcal{C}_a^2$ is identical for all $a \in \mathcal{A}$ and Bernoulli routing is used, then the mean sojourn time (and total number in the system) of pooled assignment is higher than *dedicated assignment* in a steady state.

Proof: An arriving class a job in steady state brings a workload W_a and service time $S_a = (W_a/\phi)$ for any $a \in \mathcal{A}$. Since we assume that $\mathcal{C}_a^2$ is identical for all $a \in \mathcal{A}$, we can use $\mathcal{C}^2$ as SCOV of the amount of work for all $a \in \mathcal{A}$. Note that each server has the same traffic intensity ρ. Based on our sizing strategies, we can calculate the total number of requests in the whole system for each assignment strategy by using the Pollaczek-Khintchine formula (P-K formula) (Gautam [29]) as follows:

- for the *dedicated assignment*, the number of servers for each application a is

$$N_a = \frac{1}{\rho}\frac{\lambda_a}{\phi\theta_a}$$

and arrival rate Λ_a for each server of application a is

$$\Lambda_a = \frac{\lambda_a}{\frac{1}{\rho}\frac{\lambda_a}{\phi\theta_a}} = \rho\phi\theta_a. \tag{4}$$

Thus, the expected number of requests in each queue (server) of application a in steady state is

$$L = \rho + \frac{\Lambda_a^2}{2}\frac{(Var[S_a] + (E[S_a])^2)}{(1-\rho)}. \tag{5}$$

Then, we have the total number of requests in the whole system for *dedicated assignment* case given by

$$\begin{aligned} L_{dedicated} &= \sum_{a\in\mathcal{A}} \frac{1}{\rho}\frac{\lambda_a}{\phi\theta_a}\left(\rho + \frac{\Lambda_a^2}{2}\frac{(Var[S_a] + (E[S_a])^2)}{(1-\rho)}\right) \\ &= \sum_{a\in\mathcal{A}} \frac{\lambda_a}{\phi\theta_a} + \frac{\rho}{2}\frac{\left(1+\mathcal{C}^2\right)}{(1-\rho)}\sum_{a\in\mathcal{A}}\frac{\lambda_a}{\phi\theta_a}. \end{aligned} \tag{6}$$

by substituting for (4), and realizing that $\mathcal{C}^2 = \mathcal{C}_a^2 = \frac{Var[W_a]}{\frac{1}{\theta_a^2}}$.

- for the *pooled assignment*, the total number of servers is

$$N = \sum_{a\in\mathcal{A}} \frac{1}{\rho}\frac{\lambda_a}{\phi\theta_a}.$$

In this case, we need to use the Pollaczek-Khintchine formula for multi-class queue, thus the number of requests in each queue (server) is

$$L = \rho + \frac{1}{2}\frac{\Lambda^2 E[S^2]}{(1-\rho)} \tag{7}$$

where

$$\Lambda = \frac{\sum_{a\in\mathcal{A}}\lambda_a}{\sum_{a\in\mathcal{A}}\frac{1}{\rho}\frac{\lambda_a}{\phi\theta_a}} \tag{8}$$

and

$$E[S^2] = \frac{\sum_{a\in\mathcal{A}}\lambda_a E[S_a^2]}{\sum_{a\in\mathcal{A}}\lambda_a} = \frac{\sum_{a\in\mathcal{A}}\lambda_a\left(Var[S_a] + \frac{1}{\phi^2\theta_a^2}\right)}{\sum_{a\in\mathcal{A}}\lambda_a}.$$

Then, we can calculate the total number of requests in the whole system for *pooled assignment* case as

$$\begin{aligned} L_{pooled} &= \left(\rho + \frac{1}{2}\frac{\Lambda^2 E[S^2]}{(1-\rho)}\right)\sum_{a\in\mathcal{A}}\frac{1}{\rho}\frac{\lambda_a}{\phi\theta_a} \\ &= \sum_{a\in\mathcal{A}}\frac{\lambda_a}{\phi\theta_a} + \frac{\rho}{2}\frac{\left(1+C^2\right)}{(1-\rho)}\left(\frac{\sum_{a\in\mathcal{A}}\frac{\lambda_a}{\phi^2\theta_a^2}}{\sum_{a\in\mathcal{A}}\frac{\lambda_a}{\phi\theta_a}}\right)\sum_{a\in\mathcal{A}}\lambda_a. \end{aligned} \tag{9}$$

by substituting for (8), and realizing that $\mathcal{C}^2 = \mathcal{C}_a^2 = \frac{Var[W_a]}{\frac{1}{\theta_a^2}}$.

Based on Equation (6) and (9), $L_{dedicated} \leq L_{pooled}$ if

$$\left(\sum_{a\in\mathcal{A}}\frac{\lambda_a}{\phi\theta_a}\right)^2 \leq \left(\sum_{a\in\mathcal{A}}\frac{\lambda_a}{(\phi\theta_a)^2}\right)\sum_{a\in\mathcal{A}}\lambda_a. \tag{10}$$

We can represent left-hand side of Equation (10) as

$$\left(\sum_{a\in\mathcal{A}}\frac{\lambda_a}{\phi\theta_a}\right)^2 = \sum_{i\in\mathcal{A}}\sum_{j\in\mathcal{A}}\lambda_i\lambda_j\frac{1}{\phi\theta_i}\frac{1}{\phi\theta_j}. \tag{11}$$

Likewise the right-hand side of Equation (10) as

$$\left(\sum_{a\in\mathcal{A}}\frac{\lambda_a}{(\phi\theta_a)^2}\right)\sum_{a\in\mathcal{A}}\lambda_a = \sum_{i\in\mathcal{A}}\sum_{j\in\mathcal{A}}\lambda_i\lambda_j\frac{1}{(\phi\theta_i)^2}. \tag{12}$$

Now, using the fact that

$$\sum_{i\in\mathcal{A}}\sum_{j\in\mathcal{A}}\lambda_i\lambda_j\left(\frac{1}{\phi\theta_i} - \frac{1}{\phi\theta_j}\right)^2 \geq 0$$

we can show Equation (10) is true as since

$$2\sum_{i\in\mathcal{A}}\sum_{j\in\mathcal{A}}\lambda_i\lambda_j\frac{1}{\phi\theta_i}\frac{1}{\phi\theta_j} \leq 2\sum_{i\in\mathcal{A}}\sum_{j\in\mathcal{A}}\lambda_i\lambda_j\frac{1}{(\phi\theta_i)^2}. \tag{13}$$

Finally, by using Little's Law (Gautam [29]), the sojourn times of *dedicated assignment* case is better than the pooled case. ■

From Theorem 1, we can conclude that the *dedicated assignment* appears to be more effective than the pooled assignment for the total number of requests as well as the mean sojourn time.

Remark 1: Based on Theorem 1, we make the following comments: (i) even if we assign a subset of applications to each server (not pooling *all* classes), it would still be worse than having a dedicated server for each application; (ii) we conjecture that if the $\mathcal{C}^2$ were different for the applications, the result would still remain (we will verify this conjecture in the numerical studies in Section VII-B1); (iii) we require the arrival rates to be homogeneous across time for each application, and it turns out, as shown in the next section, that this requirement would be satisfied as we will create time-stationary queues as described in Section V.

V. Time-Stability

As we previously mentioned, the main objective is to suggest an approach which provides performance bounds and guarantees based on time-stability for the non-homogeneous and transient system. In this section, we describe our notion of time-stability and the approach to obtain time-stability based on the suggested analytical framework.

A. Notion of Time-stability

As we described in Section II-A, we consider a system of N parallel queues with a single dispatcher. Each queue has a single server that may be on or off at time t. For all $n \in \{1,\dots,N\}$, at time t let $X_n(t)$ be the number of jobs in queue n and $O_n(t)$ be the status of the server (with $O_n(t)=1$ denoting 'on' and $O_n(t)=0$ denoting 'off'). Let $\lambda_a(t)$ be the arrival rate of class a jobs at the dispatcher at time t. We assume that we can divide time into arbitrarily small intervals (of appropriate time units) such that $\lambda_a(t)=\lambda_a([t])$, i.e. the arrival rate stays constant in the interval $[t,t+1)$ for all t (the notation $[t]$ denotes the integer part of t). Let $W_a(t)$ be the sojourn times experienced by a class a job that arrives into the dispatcher at time t.

As described in Section III, we seek to obtain a policy for deciding (i) $\sum_n O_n([t])$, the total number of servers that would be 'on' in interval $[t,t+1)$ for all t (sizing); (ii) the allocation scheme of applications to servers (assignment); (iii) the policy for routing requests from despatcher to server queues (routing). Based on this, our key objective is to ensure time-stability of both queue lengths for powered-on servers as well as sojourn times for a class of application. In other words, for all $t \in [0,\infty)$, $s \in [0,\infty)$, and $i \in \{0,1,2,\dots\}$,

$$\begin{aligned} P\{X_n(t)=i|O_n(t)=1\} &= \pi_a(i) \\ P\{W_a(t)\le s\} &= \Psi_a(s) \end{aligned}$$

where $\pi_a(i)$ and $\Psi_a(s)$ are computable constants that are not dependent on t. That is the sense of time-stability we aim to achieve. In the following sections we will show that we can achieve the aforementioned time-stability via (i) dedicated assignment of applications to servers, (ii) sizing rule for dedicated assignment in Equation (1), (iii) either Bernoulli routing or round-robin routing, (iv) dummy requests, and (v) adjusting the remaining work for the head-of-line job. In fact, if we use round-robin (or Bernoulli) routing, then $\pi_a(i)$ is the stationary probability that a $D/G/1$ (or $M/G/1$) queue has i jobs in the system and $\Psi_a(s)$ is the CDF of sojourn times of an arbitrary job of the corresponding queue.

B. Approach to Obtain Time-stability

In the previous section, we introduce the notion of time-stability considered in this study. Based on our notion of time stability, in this section we suggest an approach to obtain time-stability which consists of two main procedures. First we decompose non-homogeneous, multiple, parallel single-server queue system into individual simple time-homogeneous queues, and then we add "dummies" to ensure the steady state of each class a server while powering servers on and off. We describe details of the procedure in the following subsections.

1) Non-homogeneous traffic control: As described in Section III-C, we consider two routing strategies, round-robin and Bernoulli, and the following theorem characterizes the arrival process for both round-robin and Bernoulli routing based on pooled assignment.

Theorem 2: For the pooled assignment strategy, each server that is powered on during phase ℓ gets arrivals deterministically (exponentially) at rate

$$\frac{\sum_{a\in\mathcal{A}}\lambda_{a,\ell}}{\frac{1}{\rho}\sum_{a\in\mathcal{A}}\frac{\lambda_{a,\ell}}{\phi\theta_a}}$$

for all $\ell \in \mathcal{T}$ and $a \in \mathcal{A}$, under Round-robin (Bernoulli) routing, as $N\to\infty$. And the expected workload (in KB/s) that each request brings (by conditioning on the class) is

$$\sum_{a\in\mathcal{A}}\left(\frac{\lambda_{a,\ell}}{\sum_{b\in\mathcal{A}}\lambda_{b,\ell}}\right)\frac{1}{\theta_a}.$$

Proof: The net arrival rate to the dispatcher in phase ℓ is $\sum_{a\in\mathcal{A}}\lambda_{a,\ell}$. Thus the time between request arrival at the dispatcher in phase ℓ is exponentially distributed with parameter $\sum_{a\in\mathcal{A}}\lambda_{a,\ell}$. Then, due to round-robin routing, each server that is powered on in phase ℓ observes inter-arrival time which is the sum of N_ℓ IID exponentially distributed times with parameter $\sum_{a\in\mathcal{A}}\lambda_{a,\ell}$. Thus, the inter-arrival times to a powered on server is according to an Erlang distribution with mean $N_\ell/\sum_{a\in\mathcal{A}}\lambda_{a,\ell}$ and variance $N_\ell/\left(\sum_{a\in\mathcal{A}}\lambda_{a,\ell}\right)^2$. In the limit as $N\to\infty$, using the expression for N_ℓ in Equation (2), the mean term converges to

$$\frac{\sum_{a\in\mathcal{A}}\frac{1}{\rho\phi}\frac{\lambda_{a,\ell}}{\theta_a}}{\sum_{a\in\mathcal{A}}\lambda_{a,\ell}}$$

while the variance term converges to zero. Thus the time between arrivals become deterministic in the limit as $N\to\infty$ and each server that is on during phase ℓ gets arrivals deterministically at rate

$$\frac{\sum_{a\in\mathcal{A}}\lambda_{a,\ell}}{\sum_{a\in\mathcal{A}}\frac{1}{\rho\phi}\frac{\lambda_{a,\ell}}{\theta_a}}.$$

Now, we can compute the expected workload (in KB) that each request brings (by conditioning on the class) as

$$\sum_{a\in\mathcal{A}}\left(\frac{\lambda_{a,\ell}}{\sum_{b\in\mathcal{A}}\lambda_{b,\ell}}\right)\frac{1}{\theta_a}$$

and thus by multiplying by the expected arrival rate the expected workload arrival is $\rho\phi$ KB/s. Now, if round-robin routing is replaced with Bernoulli routing, then the only change in the theorem would be to replace both occurrences of the word "deterministically" with "exponentially distributed." This is because after a Bernoulli split, the resulting processes are Poisson processes with identical rates as the deterministic arrivals (however, note that we do not require the $N\to\infty$ for this case). Otherwise, everything else remains the same. ■

Theorem 2 concludes that for either routing case, round-robin or Bernoulli, pooling all applications in one server (*pooled assignment*) would result in a non-homogeneous system without time-stability because each server has time-varying arrival rates for $\ell \in \mathcal{T}$ under both routing strategies.

However, the next theorem shows that time-stability can possibly be obtained with *dedicated assignment* strategy.

Theorem 3: For the *dedicated assignment* strategy, each server of application a that is powered on at any time gets arrivals deterministically (exponentially) at rate $\rho\phi\theta_a$ under round-robin (Bernoulli) routing strategies and each arrival brings work according to CDF $H_a(\cdot)$ as $N \to \infty$. Also, each powered "on" class a server faces an expected workload $\rho\theta_a$ KB/s at all times.

Proof: During phase ℓ, requests of class a arrive according to a Poisson process with mean rate $\lambda_{a,\ell}$. First consider round-robin routing. For the *dedicated assignment*, each server hosting class a and is powered on in phase ℓ observes inter-arrival time which is the sum of $N_{a,\ell}$ IID exponentially distributed times with parameter $\lambda_{a,\ell}$ since for each class a the dispatcher performs a round-robin of the servers within the class. Thus the inter-arrival times to a class-a powered-on server is according to an Erlang distribution with mean $N_{a,\ell}/\lambda_{a,\ell}$ and variance $N_{a,\ell}/\lambda_{a,\ell}^2$. In the limit as $N \to \infty$, $N_{a,\ell} \to \infty$ the mean term converges to $1/(\rho\theta_a)$ by substituting for $N_{a,\ell}$ from Equation (1), while the variance converges to zero. Thus, the time between arrivals becomes deterministic in the limit as $N \to \infty$ and each class-a server that is on during phase ℓ gets arrivals deterministically at rate $\rho\phi\theta_a$. The expected workload (in KB) that each request brings (by conditioning on the class) to a class-a server is $1/\theta_a$, and thus the expected workload arrival rate is $\rho\phi$ KB/s. With Bernoulli routing instead of round-robin, the resulting split processes going into each powered-on server are Poisson process, and each server gets arrivals exponentially at rate $\rho\phi\theta_a$. ■

Based on Theorem 3, when only one application is assigned to a server (*dedicated assignment*), each server of application a that is powered on at any time phase gets homogeneous arrival process and also each arrival brings work according to CDF $H_a(\cdot)$. These will form the building blocks for creating time-stable queue length processes in powered-on servers. The next section describes how to obtain time-stability with powering on and off schemes.

2) Time-stability by Adding Dummy Requests: The previous section showed that each individual server of application a gets time-homogeneous arrival process and workload distribution with *dedicated assignment*. However, our concern is whether powering servers on would cause problems for achieving time-stable performance. It is intuitive to think that stationarity would be affected during times when servers are powered on and off, i.e. between phases. In other words, homogeneous arrival process and workload distribution are not sufficient to achieve time-stable performance since the initial conditions in an interval are different when powering servers on. This is especially the case when time intervals are short and steady-state is not reached, then the initial conditions become significant.

To address this problem of initial conditions, we introduce dummy requests to adjust the initial number of requests in a queue of a newly powered-on servers. In order to ensure the steady-state of each class a server that is powered on afresh at the beginning of an interval, we generated dummy requests sampled from the stationary distribution of a $D/G/1$ queue for round-robin routing or an $M/G/1$ queue for Bernoulli routing. For $M/G/1$ queue case under a FCFS (note that the formulas have to be tweaked appropriately for other polices such as versions of processor sharing), we can use the probability generating function of the stationary queue length distribution (Gautam [29]) for class a server,

$$\pi_a(i) = \frac{(1-\rho)(1-i)\tilde{G}(\lambda_a - \lambda_a i)}{\tilde{G}(\lambda_a - \lambda_a i) - i} \tag{14}$$

where $\lambda_a = \rho\phi\theta_a$ and $\tilde{G}(s) = \int_0^\infty e^{-sx} dG(x)$, the Laplace-Stieltjes transform (LST) of the service time distribution $G(\cdot)$. Note that service time would be X/ϕ seconds with a random amount of workload X kB and processing speed ϕ kB/s, and we have $G(y) = P[Y \leq y] = P[\frac{X}{\phi} \leq y] = P[X \leq \phi y] = H(\phi y)$ where $H(\cdot)$ is cumulative distribution function for workload. From Equation (14), we derive moment-generating function of the stationary queue length distribution for class a server defined by workload distribution $H(\cdot)$,

$$\pi_a(i) = \frac{(1-\rho)(1-i)\tilde{H}(\theta_a\rho - \theta_a\rho i)}{\tilde{H}(\theta_a\rho - \theta_a\rho i) - i} \tag{15}$$

where $\tilde{H}(s) = \int_0^\infty e^{-sx} dH(x)$, the LST of workload distribution. Now, we can initially populate the number of requests in queue by sampling from the distribution in Equation (15). For $D/G/1$ queue case, we do not have an exact formula for the stationary queue length distribution, but instead, we can simulate a single $D/G/1$ queue offline and obtain the distribution numerically. Note that such a simulation is extremely inexpensive and straightforward.

In addition since the objective is to create a time-homogeneous system, at any given time the system characteristics must be stationary. In particular, at times when a server is powered on, not only the number of dummy requests be according to the stationary distribution but the amount of work completed for the request at the head of the line (if any) must also be stationary. Using results from renewal theory, we know that the remaining work for the job at the head of the line is according to its stationary excess distribution (Gautam [29]). Stationary excess distribution $F_e(t)$ associated with CDF $F(t)$ in terms of the mean $\tau = -\tilde{F}'(0)$ such that

$$F_e(t) = \frac{1}{\tau}\int_0^t [1 - F(u)]du. \tag{16}$$

We now illustrate the stationary excess distribution and its computation for the Coxian-2 random variable that will be used in Section VII. It results in the following theorem for the stationary excess distribution of Coxian-2 distribution.

Theorem 4: The stationary excess distribution of Coxian-2 distribution is also Coxian-2 distribution albeit with different parameters.

Idea of proof: By using the LST we can easily show that CDF of Coxian-2 distribution can be represented as a linear combination of two CDFs of exponential distribution. Moreover, stationary excess distribution of Coxian-2 distribution can be defined as a linear combination of two CDFs of exponential distribution which means that stationary excess distribution is also Coxian-2 distribution.

Next, we introduce dummy traffic to adjust the arrival rate to each powered-on server under *dedicated assignment*. Recall that we determined the number of class a servers powered on in phase ℓ, $N_{a,\ell}$ using Equation (1) as described in Section III-B,

$$N_{a,\ell} = \left\lceil \frac{1}{\rho\phi} \frac{\lambda_{a,\ell}}{\theta_a} \right\rceil$$

to ensure that each powered-on server gets a desirable traffic intensity ρ in any time phase for both *pooled assignment* and *dedicated assignment*. In case N is finite, we need to adjust $\lambda_{a,\ell}$ by adding dummy traffic for class a so that the net arrival rate in phase ℓ is $N_{a,\ell}\rho\phi\theta_a$. Adding dummy traffic can ensure a homogeneous arrival process for each class with *dedicated assignment*. In this case the amount of additional dummy traffic would be

$$\left(\left\lceil \frac{1}{\rho\phi} \frac{\lambda_{a,\ell}}{\theta_a} \right\rceil - \frac{1}{\rho\phi} \frac{\lambda_{a,\ell}}{\theta_a} \right) \rho\phi\theta_a \leq 1 \times \rho\phi\theta_a.$$

Note that the maximum amount of additional traffic into each powered on class a server would be less than $\rho\phi\theta_a / \left\lceil \frac{1}{\rho\phi} \frac{\lambda_{a,\ell}}{\theta_a} \right\rceil$ and if total number of N is large (which is fairly common in data centers), then the amount of additional traffic would be insignificant. We will compare actual arrivals with adjusted arrivals in Section VII-B6). Now, based on the results from previous sections and strategy for dummies, we can arrive at the following theorem which shows that time-stable performance can be achieved by the suggested approach.

Theorem 5: The number of requests in any powered on server processing class a requests at any time in an interval would be stationary according to the stationary distribution of an $D/G/1$ or $M/G/1$ queue depending on round-robin or Bernoulli routing, thereby resulting in a time-stable performance.

Proof: We need to show that initial conditions of class a servers, especially those powered on afresh, in an every time interval ℓ are according to stationary queues with dummy requests. Considering an arbitrary class a and an arbitrary interval of time ℓ. For convenience, we let the beginning of this interval be time $t = 0$ and select an arbitrary class a server that is powered on afresh at time $t = 0$, i.e. powered on in interval ℓ but powered-off in the previous interval. Clearly, by adding "dummy" jobs as described above, the number of jobs in the server as well as the amount workload at time $t = 0$ are according to those of a stationary $D/G/1$ ($M/G/1$) queue under round-robin (Bernoulli) routing. Also, since the arrival process and the amount of work an arrival brings remain unchanged throughout the time the server is on (even if it is over multiple intervals) with *dedicated assignment* as described in Section V-B1, the workload process is Markovian for Bernoulli routing and delayed semi-Markovian for Round-robin routing, due to stationarity and ergodicity properties which would result in time-stable performance. Thus the number in the system or the workload observed at any time t during the server's on-time sojourn would remain stationary regardless of powering servers on and off (note that this includes time intervals beyond ℓ). ■

In Section VI-A, we will show that time-stability of the number of requests in system could be extended to time-stability of sojourn times in a straightforward fashion.

3) Step-by-Step Procedure: The following is a procedure to achieve time-stability:

Step 1. Off-line Phase

Step 1.1. By using dedicated assignment, determine the number of servers for each class a and for each time period $N_{a,\ell}$ by using Equation (1).

Step 1.2. Obtain the queue length distribution $\pi_a(i)$ for $M/G/1$ queue analytically or $D/G/1$ queue via simulation to sample from for initial number of dummy requests for initial condition.

Step 1.3. Add dummy traffic so that arrival rate for class a for time ℓ is $N_{a,\ell}\rho\phi\theta_a$.

Step 2. On-line Phase

Step 2.1. At the beginning (or end) of each time period, compute the difference in the number of servers between consecutive time periods based on the number of servers computed in [Step 1.1].

Step 2.2. If $N_{a,\ell} < N_{a,\ell+1}$, then

Step 2.2.1. Select $N_{a,\ell+1} - N_{a,\ell}$ servers to be powered on randomly among the "off" servers.

Step 2.2.2. Add dummy requests to each newly powered on server by sampling the number of dummy requests from the queue length distribution $\pi_a(i)$ derived in [Step 1.3].

Step 2.2.3. Adjust the amount of remaining work of the very first dummy request of each newly powered on server based on the stationary excess distribution.

Step 2.3. If $N_{a,\ell} > N_{a,\ell+1}$, then

Step 2.3.1. Select $N_{a,\ell} - N_{a,\ell+1}$ servers randomly among the "on" servers.

Step 2.3.2. If selected server is idle, then just power off selected server.

Step 2.3.3. Otherwise, set status of server as "to be off" and do not route incoming requests to that server, then power off when server completes service of the last remaining request.

C. Performance Bounds and Guarantees

As we described in the previous Section V-B2, dummies are used to (i) adjust the initial number of requests in a queue of newly powered-on servers and (ii) adjust the class dependent arrival rate to each powered-on server. Although adding dummies is crucial to obtain time-stability, it also degrades performance and thus practitioners may have concerns about this issue. In this situation if the practitioners choose not to add dummy requests, then time-stability predictions would be an upper bound on actual performance. In other words, the mean queue length would be time-varying without using dummies, but strictly bounded by time-stable performance which can be obtained by adding dummies. From both theoretical and practical points of view, such performance bounds are extremely useful since bounds are provable and derived by stationary analysis of queueing model (e.g. P-K formula) for non-homogeneous and transient system. Note

that it is difficult to yield time-stable performance or obtain the provable bounds on performance of time-varying system especially when steady-state cannot be reached.

Remark 2: Time-stable and provable performance bounds cannot be obtained by simply assuming stationarity without adding dummies. To explain this, let S1 be an original system which determines $N_{a,\ell}$ by using (1), but does not add both types of dummies. In fact in S1, arrival rates of class a requests into each powered-on server j, $\lambda_{a,j,\ell}$ which can be defined as $\lambda_{a,j,\ell} = \frac{\lambda_{a,\ell}}{N_{a,\ell}}$ where $\sum_{j\in\mathcal{N}} \lambda_{a,j,\ell} = \lambda_{a,\ell}$, would be time-varying across time intervals. In other words, $\lambda_{a,j,\ell} \neq \rho\phi\theta_a$ for all $\ell \in \mathcal{T}$, since $\lambda_{a,j,\ell} = \lambda_{a,\ell}/N_{a,\ell}$ but $N_{a,\ell} = \left\lceil \frac{1}{\rho\phi}\frac{\lambda_{a,\ell}}{\theta_a} \right\rceil \neq \frac{1}{\rho\phi}\frac{\lambda_{a,\ell}}{\theta_a}$. In this case, we can use standard PK formula for $M/G/1$ queue model by assuming stationarity to compute the mean queue length. Mean queue length of class a server in time interval ℓ, $L_{a,\ell}$ can be computed as,

$$L_{a,\ell} = \frac{\lambda_{a,j,\ell}}{\phi\theta_a} + \frac{1}{2}\left(\frac{\lambda_{a,j,\ell}}{\phi\theta_a}\right)^2 \left(\frac{1+C_a^2}{1-\frac{\lambda_{a,j,\ell}}{\phi\theta_a}}\right).$$

Based on above equation, our claim is that we cannot obtain time-stable upper bound on the mean queue length and thus $L_{a,1} \neq L_{a,2} \neq \cdots \neq L_{a,T} \neq \bar{L}_a$ where $\bar{L}_a$ is an upper bound on the mean queue length obtained by our suggested approach, since arrival rates $\lambda_{a,j,\ell}$ are different across time intervals without adding dummy traffic as described in Section V-B2. In this case, $L_a^{\max} = \max\{L_{a,1}, L_{a,2}, \ldots, L_{a,T}\}$ would be an upper bound, however $\bar{L}_a \leq L_a^{\max}$. In other words, assuming steady-state itself is not enough to obtain time-stable and provable upper bound $\bar{L}_a$, and our suggested approach provides essential conditions to obtain an upper bound $\bar{L}_a$ which is provable and can be applied to transient system without assuming steady state assumption (which is impossible for real system).

Although time-stable performance bounds provided by our suggested approach are useful, it is important to analyze the gap between time-varying actual performance with performance bounds. First of all, it is reasonable to expect that the gap between bounds and actual performance would be bigger when arrival rates are increasing more drastically since the actual performance is highly dependent with the increment of the number of servers. In other words, since every newly powered-on server starts serving incoming requests with empty queue, the mean queue length would be decreasing when the number of server is increasing. In addition the gap between bounds and actual performance is highly dependent with variance of workloads and also system utilization based on analysis of queueing model (e.g. P-K formula (5) and (7) used in Section IV). Considering that providing performance bounds and guarantees based on time-stability opposed to time-varying and transient system has not been addressed before our study, we believe that our study has both theoretical and practical contributions. In Section VII-B3 we will introduce simulation results to compare the time-varying actual performance with time-stable bounds and analyze the gap for the different SCOV of workload distribution and the desired traffic intensity ρ.

VI. Discussion on Time-stability

In this section we discuss details of time-stability obtained by our suggested approach in terms of its extension and limitations.

A. Extension to Sojourn Times

As introduced in Section V-A, our suggested approach stabilizes queue length distribution. Then we consider sojourn times as users of data center need to get Quality of Service (QoS) guarantees in terms of sojourn times. In fact, when the distribution of the queue lengths is stabilized, performance analysis of system is very straightforward in terms of sojourn times. Since the distribution of number of jobs in each powered-on class a server is time-stable, the amount of work brought by jobs is time-invariant, and service speed is constant for each server, the sojourn time distribution would also be time-stable. Therefore based on queue length distribution, we can derive time-stable sojourn time distribution which enables us to provide probabilistic guarantees of the response times for incoming requests. In other words, under a FCFS regime, distribution of sojourn time of class a at time t, $W_a(t)$, can be defined as $\Psi_a(w) = P[W_a(t) \leq w]$ (which is not dependent on time t). Providing probabilistic guarantees on sojourn times (as well as queue lengths) based on time-stability has significant benefits since for transient system with time-varying and non-stationary load, it is extremely difficult to provide guaranteed SLA without assumption for steady-state. For example, our approach is able to provide a bound τ on average sojourn time such that $E[W_a(t)] \leq \tau$, or tail probability of response time for bound τ such that $P[W_a(t) \leq \tau] \geq 1-\epsilon$ which would remain unchanged across time. Without assuming that system reaches steady-state in each time interval, the only way to provide guarantees is running a large number of servers which causes a much higher energy consumption. In this context, achieving time-stability and providing performance bounds and guarantees based on dummies is the key benefit of our suggested approach.

As described in Section V-B1, under suggested framework we can decompose our system into simpler homogeneous queues, $D/G/1$ queue for round-robin routing and $M/G/1$ queue for Bernoulli routing. In this case, for the $M/G/1$ queue we have the LST of the sojourn time distribution $\tilde{\Psi}_a(s)$ for class a request as (Gautam [29]),

$$\tilde{\Psi}_a(s) = \frac{(1-\rho)s\tilde{G}(s)}{s-\lambda_a(1-\tilde{G}(s))} \tag{17}$$

where $\lambda_a = \rho\phi\theta_a$ and $\tilde{G}(s) = \int_0^\infty e^{-sx}dG(x)$, the LST of service time distribution. Although we do not have a specific formula for the sojourn time distribution of $D/G/1$ queue case, we can derive the sojourn time distribution from simulation with $D/G/1$ queue setting. Note that it is not easy to derive continuous sojourn time distribution, thus we can derive it based on queue length distribution, $\pi_a(i)$ itself. Indeed, we can apply derived sojourn time distribution to each server under round-robin routing in time-stable manner. Based on our analysis, we can also obtain time-stable performance bound on sojourn times as well as queue lengths. In Section

VII-B2 we will introduce simulation results which show that the mean and standard deviation of sojourn times are stabilized with our suggested approach.

B. Time Interval Length

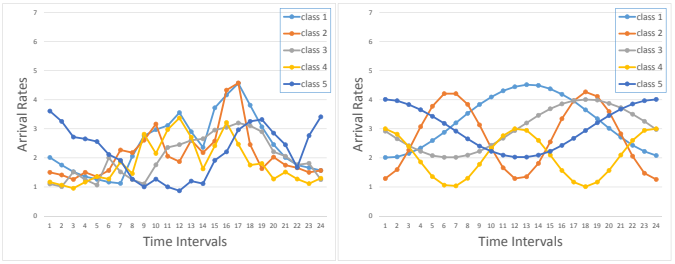
In order to model time-varying arrivals of requests, we assume that requests arrive according to a piecewise constant non-homogeneous Poisson process where arrival rates of requests of application classes stay constant in each time interval. In this situation, we need to carefully think about the effect of time interval length in terms of whether our suggested approach would be robust to time interval length. In other words, we need to check whether distribution of queue lengths or sojourn times would be time-stable with small time interval length when arrival rates change very fast. In this context, we would like to note that our suggested approach would perform well when time interval length too small to reach steady-state within each time interval and has a sense of the robustness to time interval length. Note that for implementation it is reasonable to assume that the service times and inter arrival times of requests are much smaller than time interval length since the case where the service times are longer than time interval length is unlikely in practice for data centers. In Section VII-B4, we will compare the simulation results with different time interval lengths to show robustness of our approach.

C. System Size (Total Number of Servers)

In this study, we consider a fairly common situation in data centers where the traffic of requests is very high and a large number of servers are necessary, and thus we use the asymptotic scaling where both the arrival rates and number of servers are scaled with size N. In fact, our suggested approach itself has limitation with small size N, since for round-robin routing arrival rate into each powered-on server would not be time-homogeneous if size N is small as shown in proof of Theorem 3. Therefore queue length distribution is also non-homogeneous with small size N. Note that for the case of using Bernoulli routing, arrival rate into each powered-on server would be time-stable regardless of size N. In Section VII-B5 we will compare simulation results with small size N for both round-robin and Bernoulli routing cases to check the limitation of our approach.

VII. Numerical Evaluation

In this section we describe the simulation settings and then analyze the results of simulation experiments to evaluate our approach. We verify our additional insight for assignments and show that our suggested approach provides time-stability in both queue length distributions and sojourn time distributions based on simulation results. Also we analyze performance bounds and effects of both time interval length and system size N to time-stability.



(a) Pattern A (b) Pattern B

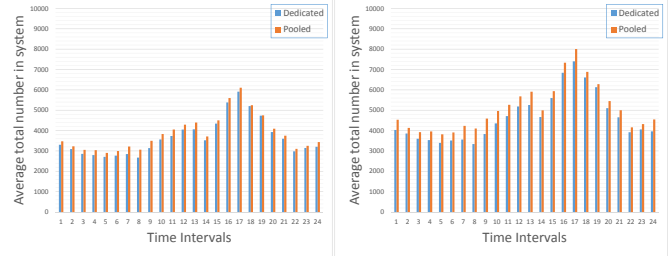
Fig. 2. Normalized arrival rate $\alpha_{a,\ell}$ for the 5 classes for 1 cycle of 24 equal-length phases

A. Simulation Experiments

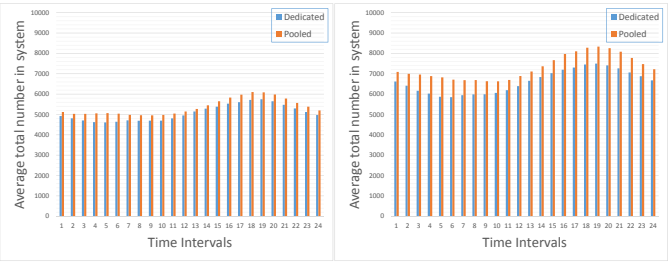
We developed a simulation on a Java platform with $N = 1000$ possible servers using two sets of input data for the arrival rates and two sets for the workloads. The input data will be discussed in the latter part of this section. We used 5 classes of requests, hence $\mathcal{A} = \{1, 2, 3, 4, 5\}$ and 24 equally spaced time intervals (time interval length is 60 minutes), hence $\mathcal{T} = \{1, 2, ..., 24\}$. We assume that the request inter-arrival times are much shorter than the time intervals and it is crucial to note that although for the analysis we do not require the intervals be equally spaced, it is that way to avoid a cumbersome presentation.

Next we describe the 5 classes' workload characteristics. Note that we used Coxian-2 distribution for the workload described in Section II-B. We define two sets of amount of work data, both having the same mean amount of work $1/\theta_a$ for all $a \in \mathcal{A}$ as 20, 15.2381, 25, 17.619, 21 seconds. These two sets have different conditions for SCOV, one has the same value of SCOV, 0.7, for all $a \in \mathcal{A}$ and the other has SCOV of the amount of work for classes 1, 2, 3, 4 and 5 as 1, 0.8887, 2.2, 1.335, and 0.9501 respectively. Since we used different SCOV for amount of work (but the mean amount of work is the same), we needed to define the parameters of the Coxian-2 distribution, θ_1, θ_2 and p, differently for each set of amount of work. For the same SCOV case, we used probability p as 0.9375, 0.6099, 0.8, 0.9591 and 0.8748 for classes 1, 2, 3, 4 and 5 respectively. Also, for the different SCOV case, we have probability p as 0.9, 0.95, 0.05, 0.1 and 0.55 for classes 1, 2, 3, 4 and 5 respectively. The θ_1 and θ_2 values can be obtained using the fact that the mean and SCOV of the Coxian-2 distribution are $\frac{1}{\theta_1} + \frac{p}{\theta_2}$ and $\frac{\frac{1}{\theta_1^2} + \frac{2p-p^2}{\theta_2^2}}{\frac{1}{\theta_1^2} + \frac{p^2}{\theta_2^2} + \frac{2p}{\theta_1\theta_2}}$ respectively. Note that we considered only one processing speed, $\phi = 0.52$.

Also, we used two data sets for arrival rates, pattern A and B for performance analysis. Graphs of the arrival rates $\alpha_{a,\ell}$ for two arrival rate patterns are provided in Figures 2a and 2b respectively. In pattern A notice that arrival rate of some classes are correlated with others over time and the peak times are not necessarily the same. Our intention was to select a representative sample to illustrate both heterogeneity as well as issues such as correlation. Also, in pattern B, we defined arrival rate as sinusoidal function for $t \in \mathcal{T}$. The sinusoidal form of the arrival rate is clearly a mathematical abstraction which

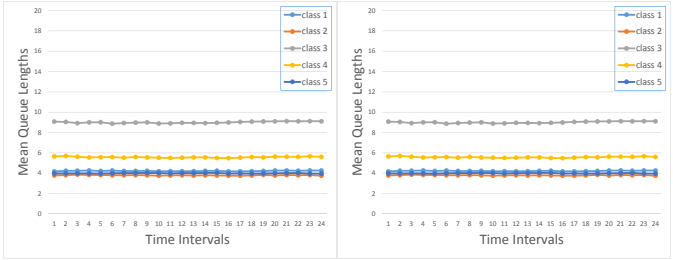


(a) Pattern A - same SCOV (b) Pattern A - different SCOV

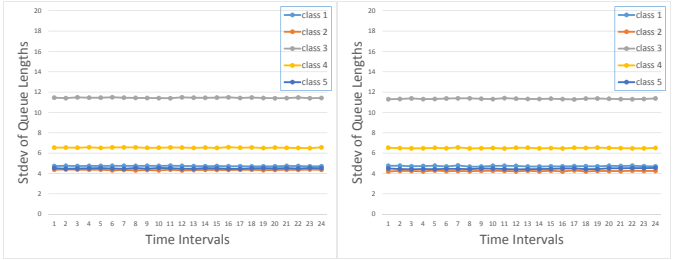


(c) Pattern B - same SCOV (d) Pattern B - different SCOV

Fig. 3. Comparing average total number in system across all classes over 1 cycle: Dedicated vs Pooled



(a) Mean for Pattern A (b) Mean for Pattern B



(c) Stdev for Pattern A (d) Stdev for Pattern B

Fig. 4. Mean and standard deviation of queue length for the 5 classes across a cycle with round-robin routing

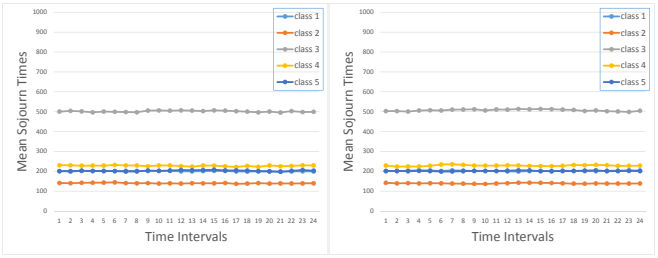
has the essential property of producing significant fluctuations over time (Liu and Whitt [21]). This particular arrival rate pattern is by no means critical for our approach; our approach applied to an arbitrary arrival rate but it is convenient to show how it achieved time-stable performance with time-varying arrival rates. The number of powered-on servers $N_{a,\ell}$ would be determined proportional to the arrival rate by our sizing rule in Equation (1) in Section III-B. In all our simulations we only considered FCFS because implementing a processor sharing scheme with a large number of servers is extremely cumbersome with long run times. However, it is important to note that the time-stable results would be valid for any work-conserving policy such as processor sharing, limited processor sharing, etc. To enable a meaningful set of simulations in a reasonable time, we have only presented the FCFS case.

B. Results and Findings

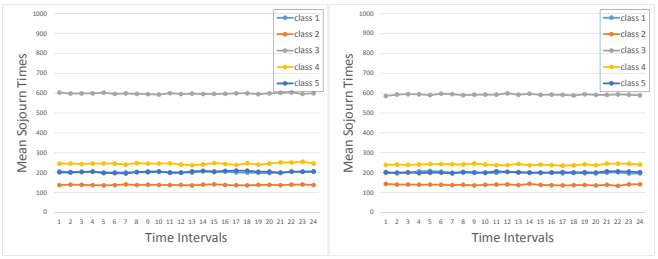
For the purpose of performance analysis, we define baseline condition which consists of the *dedicated assignment*, sizing as described in Section III-B and round-robin routing with traffic intensity $\rho = 0.9$. We will evaluate our approach by using baseline condition in following sections.

1) Performance comparison between assignment strategies: First, we compare the performance of assignment strategies to verify our insight described in Section IV. As described in Theorem 1, we use Bernoulli routing for the *dedicated assignment* and *pooled assignment*. Note that the total number of servers powered on at any time period is the same for both assignment strategies, we can make a fair comparison between assignment strategies. Since, as described in Section V-B1, the pooled assignment results in a non-homogeneous system, it would not be possible to use "dummy" requests for pooled assignment cases. Therefore, we compare the *dedicated assignment* case without using "dummy" requests. We compare the average total number of requests in the system by plotting it across time (also averaged across all classes) with constant SCOV in Figure 3a for arrival rate pattern A and in Figure 3c for pattern B. From the results, we can verify that *dedicated assignment* is better than pooled assignment. Moreover, we try to compare assignments with different SCOV defined in the Section VII-A to check our conjecture that our insight can be extended to more general cases where SCOV is not constant and each class has a different SCOV value. Figures 3b and 3d show that *dedicated assignment* is also better than pooled assignment with a different SCOV value for each arrival rate pattern. Since cases with different SCOV values are regarded as more general, we will consider only different (and high) SCOV for further analysis.

2) Analysis of Time-Stability: Next we analyze the time-stability of our suggested approach. As described in Section V, our approach stabilizes the distributions of the queue lengths as well as the sojourn times (see Section VI-A). Based on both time-stable distributions, first we show that the mean and standard deviation of queue lengths for 5 classes are time-stable for round-robin (baseline) in Figure 4. Note that both round-robin and Bernoulli routing result in time-stable performance as mentioned in Section V-B1, but round-robin routing indeed results in better performance than Bernoulli routing as described in Section III-C. For this reason we analyze time-stable performance of baseline (which use round-robin routing) for further analysis. Also based on Figures 4 and 5, it is worthwhile to indicate that our time-stable performance does not depend on arrival rate patterns which verifies the discussion in Section V-B1. In addition we check that distribution of sojourn times is also stabilized described in Section VI-A. As we indicated, Figure 5 show that the mean and standard deviation of sojourn times for 5 classes are time-stable.

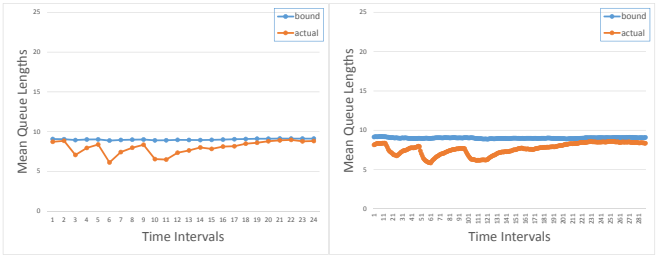


(a) Mean for Pattern A (b) Mean for Pattern B

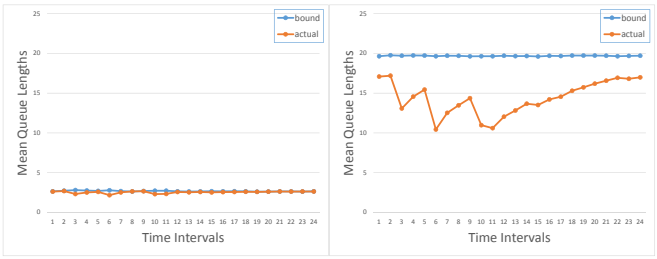


(c) Stdev for Pattern A (d) Stdev for Pattern B

Fig. 5. Mean and standard deviation of sojourn times for the 5 classes across a cycle with round-robin routing

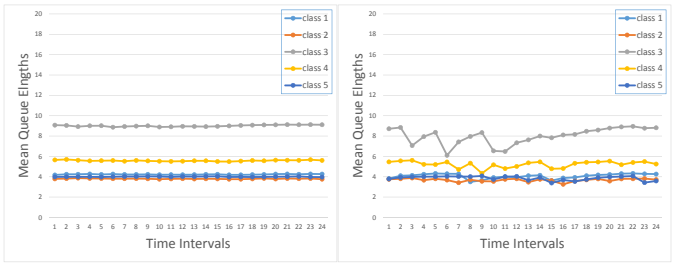


(a) 60-minutes, SCOV=2.2, $\rho = 0.9$ (b) 5-minutes



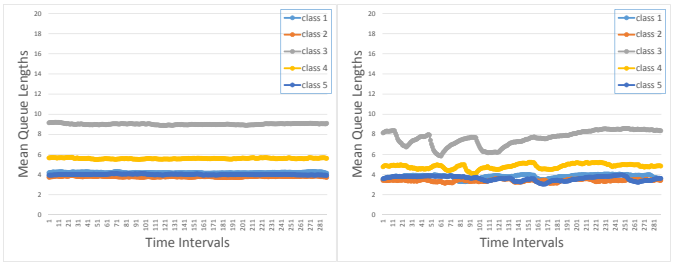
(c) SCOV=0.6 (d) $\rho = 0.95$

Fig. 7. Analysis of the gap between bound and actual performance of class 3 for different conditions



(a) With dummies (b) Without dummies

Fig. 6. Performance of the mean queue length for the 5 classes across 24 60-minutes time intervals



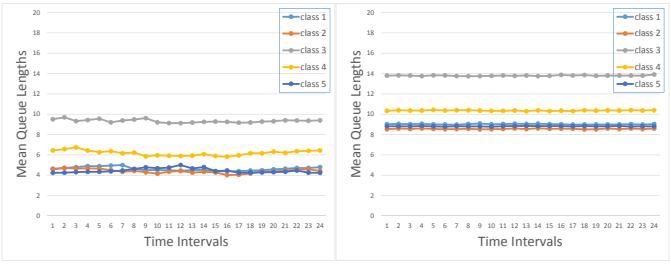
(a) With dummies (b) Without dummies

Fig. 8. Bounds and actual performance of the mean queue length for the 5 classes across 288 5-minutes time intervals

3) Bounded Performance: In Section V-C we mentioned that our time-stable performance measures would be an upper bound on actual performance without using dummies. Figure 6 compares the actual time-varying performance obtained our approach without dummies as opposed to time-stable performance bound. As we already mentioned, if dummies are not used then the mean queue lengths are time-varying across time intervals (due to empty queue of newly powered-on servers), but they are strictly bounded by the time-stable mean queue lengths obtained by adding dummies. In addition, we have claimed that the gap between actual performance and bound would be affected by both variance of workload (i.e. SCOV of workload distribution) and utilization (which can be controlled by the desired traffic intensity ρ in our approach), but it is not dependent on the time interval length. Figure 7 compares the differences between actual performance and bound for application class 3 (which shows the largest variation without dummies) according to the different conditions of SCOV, utilization and time interval length. As we expected, the performance gap would be bigger with bigger SCOV and higher utilization, but the same with smaller time interval length. Although the performance gap seems to be large for bigger SCOV of workload distribution and higher utilization, considering that it is also difficult to analyze dynamics of time-varying and transient system for both cases, we believe that our suggest approach still provide significant benefits based on time-stability.

4) The Effect of Time Interval Length: As we discussed in Section VI-B, in order to check whether our suggested approach performs well for the case with smaller time interval length, we run simulation for 288 5-minutes time intervals by decomposing 24 60-minutes interval into smaller ones with the same daily pattern. Recall that we used data set which has the mean service times of 5 classes as 38.46, 29.034, 48.0769, 33.8826, and 40.3846 seconds, and thus we believe that 5 minutes time intervals are appropriate to check the case of smaller time interval length. Figure 8 shows the bound and actual performance of the mean queue lengths for 288 5-minutes time intervals, and based on the comparison with Figure 6 we can conclude that time-stability obtained by our suggested approach is robust to time interval length.

5) The Effect of System Size: As we mentioned in Section VI-C, our suggested approach has a limitation that perfor-



(a) Round-robin with $N = 100$ (b) Bernoulli with $N = 100$

Fig. 9. Bounds on the mean queue length with $N = 100$ for both routing policies: Round-robin and Bernoulli

TABLE I
PERCENTAGE GAP BETWEEN THE ACTUAL ARRIVAL RATES (A) AND ADJUSTED ARRIVAL RATES (B): $\frac{(B-A)}{A} \times 100(\%)$

Time Intervals	Class Indices				
	class 1	class 2	class 3	class 4	class 5
1	0.6200	0.3275	0.4073	1.6294	0.2857
2	0.2857	0.9125	1.0880	1.1895	0.1143
3	1.4000	0.7370	1.0880	0.6569	0.6984
4	0.5333	0.3275	0.3392	1.6294	0.0755
5	1.0880	0.1000	1.6229	0.3462	0.5042
6	1.7391	1.0540	0.1520	1.9990	0.8163
7	2.1091	1.0100	1.0880	0.5058	0.8722
8	0.4488	1.4226	0.3392	0.7533	1.6229
9	0.4073	0.4062	0.4073	0.5570	0.2857
10	0.7390	0.4250	0.5531	0.0717	1.6229
11	0.3935	0.3774	0.3711	0.8464	0.2857
12	0.1915	1.2682	0.0947	0.6986	2.2521
13	0.0552	0.4062	0.0800	0.3462	0.2857
14	0.5702	1.4226	0.3109	1.2685	1.2987
15	0.5568	1.1706	0.2631	0.7151	0.8722
16	0.3663	0.7084	0.0446	0.4384	0.2857
17	0.2857	0.5750	0.0350	0.8281	0.4746
18	0.3737	0.2857	0.2426	0.1776	0.1143
19	0.5049	1.7351	0.0552	0.3462	0.6234
20	0.2857	1.3512	0.4073	1.9990	0.0902
21	0.6200	0.0350	0.4488	0.9365	0.0583
22	0.2857	0.5136	0.5531	1.9990	1.2987
23	0.6909	0.3275	0.8800	1.4194	0.4883
24	1.1484	1.0540	0.3392	0.1192	0.2857
Overall	0.5318	0.7370	0.3752	0.7847	0.5165

mance would not be stabilized with smaller size N for round-robin routing. To analyze the limitation of our suggested approach, we have run simulation by scaling with size $N = 100$ instead of $N = 1000$, summarized the results as shown in Figure 9. As shown in Figure 9, performance by using Bernoulli routing is stabilized with smaller size $N = 100$ (but as we mentioned performance is wore than round-robin), however round-robin does not yield time-stable performance for the case of size $N = 100$.

6) Dummy Traffic Analysis: In Section V-B2, we claimed that the amount of dummy traffic to adjust arrival rates of each application a is insignificant. We show percentile gap between the actual arrival rates and adjusted arrival rates in Table I, and the additional dummy traffic is reasonably negligible to the actual arrivals.

VIII. CONCLUDING REMARKS AND FUTURE WORKS

A number of approaches have been studied to manage resources in data centers over non-homogeneous workloads; those approaches have mainly focused on determining right-sizing of servers to minimize energy cost while considering SLA violation conditions. However, the aforementioned studies ignore achieving time-stability which makes it convenient to analyze system, provide probabilistic guarantees and performance bound under transient conditions. To the best of our knowledge, achieving time-stability over time-varying workloads while considering sizing, assignment and load balancing in integrated fashion for data centers operations has not been addressed. In this context, we suggest an approach to effectively reduce energy consumption by powering on and off just the right number of servers while being able to provide performance bounds and guarantees over fast varying arrival rates that steady-state cannot typically be reached.

This paper asks if time-stability can be attained using a combination of sizing, assignment, and routing in an integrated fashion. We have suggested an analytic framework simplifying a large scale, multi-dimensional, and non-stationary problem by decomposing into individual simpler stationary ones, and have introduced dummy requests to achieve time stability based on decomposed settings. Performance bounds and probabilistic guarantees introduced in this study are provable and simply derived by stationary analysis based on suggested framework. Also, we have introduced additional insight regarding assignment strategies and addressed extension and limitation of our suggested approach. One could consider the following in the future: (i) suggest real-time speed scaling control by varying ϕ for time-stable performance, (ii) instead of all classes with a large number of servers some classes may need to be hosted on only one server, (iii) develop an optimization framework to holistically right-size, speed scale, route and assign classes for energy efficiency, and (iv) extend to multi-server queues.

ACKNOWLEDGMENT

This material is based upon work partially supported by the AFOSR under Contract No.FA9550-13-1-0008. The authors thank the associate editor and five reviewers for their comments that resulted in significant improvement of the content and presentation of this work.

REFERENCES

[1] D. Menascé, V. Almeida, R. Riedi, F. Ribeiro, R. Fonseca, and W. Meira, Jr., "In search of invariants for e-business workloads," in *Proceedings of the 2nd ACM conference on Electronic commerce*, ser. EC '00. New York, NY, USA: ACM, 2000, pp. 56–65.

[2] "Report to congress on server and data center energy efficiency," *public law 109-431*, 2007.

[3] G. Chen, W. He, J. Liu, S. Nath, L. Rigas, L. Xiao, and F. Zhao, "Energy-aware server provisioning and load dispatching for connection-intensive internet services," in *Proceedings of the 5th USENIX Symposium on Networked Systems Design and Implementation*, vol. 5, 2008, pp. 337–350.

[4] J. Hamilton, "Cost of power in large-scale data centers," *Blog entry dated*, vol. 11, p. 28, 2008.

[5] J. Koomey, "Growth in data center electricity use 2005 to 2010," 2011.

[6] L. Barroso and U. Holzle, "The case for energy-proportional computing," *Computer*, vol. 40, no. 12, pp. 33–37, 2007.

[7] W. Vogels, "Beyond server consolidation," *Queue*, vol. 6, no. 1, pp. 20–26, 2008.

[8] D. Abts, M. R. Marty, P. M. Wells, P. Klausler, and H. Liu, "Energy proportional datacenter networks," in *ACM SIGARCH Computer Architecture News*, vol. 38, no. 3. ACM, 2010, pp. 338–347.

[9] M. Lin, A. Wierman, L. L. Andrew, and E. Thereska, "Dynamic right-sizing for power-proportional data centers," *Networking, IEEE/ACM Transactions on*, vol. 21, no. 5, pp. 1378–1391, 2013.

[10] E. Feller, L. Rilling, and C. Morin, "Energy-aware ant colony based workload placement in clouds," in *Proceedings of the 2011 IEEE/ACM 12th International Conference on Grid Computing*. IEEE Computer Society, 2011, pp. 26–33.

[11] A. Gandhi, M. Harchol-Balter, R. Raghunathan, and M. A. Kozuch, "Autoscale: Dynamic, robust capacity management for multi-tier data centers," *ACM Trans. Comput. Syst.*, vol. 30, no. 4, pp. 14:1–14:26, Nov. 2012. [Online]. Available: http://doi.acm.org/10.1145/2382553.2382556

[12] Y. C. Lee and A. Y. Zomaya, "Energy efficient utilization of resources in cloud computing systems," *The Journal of Supercomputing*, vol. 60, no. 2, pp. 268–280, 2012.

[13] K. Wang, M. Lin, F. Ciucua, A. Wierman, and C. Lin, "Characterizing the impact of the workload on the value of dynamic resizing in data centers," in *INFOCOM, 2013 Proceedings IEEE*. IEEE, 2013, pp. 515–519.

[14] R. Singh, U. Sharma, E. Cecchet, and P. Shenoy, "Autonomic mix-aware provisioning for non-stationary data center workloads," in *Proceedings of the 7th international conference on Autonomic computing*, ser. ICAC '10. New York, NY, USA: ACM, 2010, pp. 21–30.

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

[16] K. Wang, M. Lin, F. Ciucu, A. Wierman, and C. Lin, "Characterizing the impact of the workload on the value of dynamic resizing in data centers," in *Proceedings of the 12th ACM SIGMETRICS/PERFORMANCE joint international conference on Measurement and Modeling of Computer Systems*, ser. SIGMETRICS '12. New York, NY, USA: ACM, 2012, pp. 405–406.

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

[18] R. D. Foley, "Stationary poisson departure processes from non-stationary queues," *Journal of Applied Probability*, vol. 23, no. 1, pp. pp. 256–260, 1986.

[19] J. A. Barnes and R. Meili, "A stationary poisson departure process from a minimally delayed infinite server queue with non-stationary poisson arrivals," *Journal of Applied Probability*, vol. 34, no. 3, pp. pp. 767–772, 1997.

[20] Z. Feldman, A. Mandelbaum, W. A. Massey, and W. Whitt, "Staffing of time-varying queues to achieve time-stable performance," *Management Science*, vol. 54, no. 2, pp. 324–338, 2008.

[21] Y. Liu and W. Whitt, "Stabilizing customer abandonment in many-server queues with time-varying arrivals," *Operations Research*, vol. 60, no. 6, pp. 1551–1564, 2012.

[22] H.-L. Chen, J. R. Marden, and A. Wierman, "On the impact of heterogeneity and back-end scheduling in load balancing designs," in *INFOCOM 2009, IEEE*. IEEE, 2009, pp. 2267–2275.

[23] A. Gandhi, M. Harchol-Balter, R. Das, and C. Lefurgy, "Optimal power allocation in server farms," in *ACM SIGMETRICS Performance Evaluation Review*, vol. 37, no. 1. ACM, 2009, pp. 157–168.

[24] M. Harchol-Balter, A. Scheller-Wolf, and A. R. Young, "Surprising results on task assignment in server farms with high-variability workloads," in *Proceedings of the eleventh international joint conference on Measurement and modeling of computer systems*. ACM, 2009, pp. 287–298.

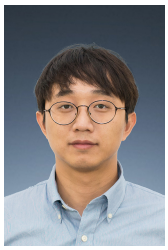
[25] D. Gmach, J. Rolia, L. Cherkasova, and A. Kemper, "Workload analysis and demand prediction of enterprise data center applications," in *Workload Characterization, 2007. IISWC 2007. IEEE 10th International Symposium on*, 2007, pp. 171–180.

[26] M. Lin, Z. Liu, A. Wierman, and L. L. Andrew, "Online algorithms for geographical load balancing," in *Green Computing Conference (IGCC), 2012 International*. IEEE, 2012, pp. 1–10.

[27] Z. Liu, A. Wierman, Y. Chen, B. Razon, and N. Chen, "Data center demand response: Avoiding the coincident peak via workload shifting and local generation," *Performance Evaluation*, vol. 70, no. 10, pp. 770–791, 2013.

[28] M. Harchol-Balter, M. E. Crovella, and C. D. Murta, "On choosing a task assignment policy for a distributed server system," *Journal of Parallel and Distributed Computing*, vol. 59, no. 2, pp. 204 – 228, 1999.

[29] N. Gautam, *Analysis of Queues: Methods and Applications*. CRC Press, 2012.



Soongeol Kwon received the B.S. and M.S. degrees in industrial engineering from Yonsei University, Seoul, Korea, in 2005 and 2007 respectively, and he is currently pursuing the Ph.D. degree in industrial and systems engineering at Texas A&M University, College Station, TX. His research interests include but are not limited to optimization, stochastic analysis and modeling and simulation, and research domains include energy-efficient data centers operations, optimal power system management, and energy-aware scheduling.

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 eight years. He received his M.S. and Ph.D. in Operations Research from the University of North Carolina at Chapel Hill, and his B.Tech. from 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.