Queueing Models and
Service Management
Vol. 3, No. 2, page 137-166, 2020

QMSM

Data Driven Server Allocation at Virtual Computing Labs

Siyun Yu[1], Nelson Lee[1], Vidayadhar G. Kulkarni[1,*] and Haipeng Shen[2]

[1]Department of Statistics and Operations Research
The University of North Carolina at Chapel Hill, Chapel Hill, NC 27599, USA
[2]Faculty of Business and Economics, The University of Hong Kong, Hong Kong
(Received November 2019 ; accepted March 2020)

Abstract: Virtual computing labs (VCL) are cloud computing platforms that provide users remote access to software applications. In this paper, we develop a data driven approach for server allocation in a VCL. The main challenge is to decide how many servers should be preloaded with which applications, and how many servers should be left flexible, to be loaded with the requested applications on demand. If a preloaded server with a desired application is available, the user gets immediate access. If not, the user gets delayed access after some extra loading time, if a flexible server is available. If no server (dedicated or flexible) is available, the user is blocked. We measure the service quality by the fractions of users who get immediate or delayed access, and the system cost by the number of on servers (that is, the sum of pre-loaded and flexible servers). We propose an implementable data driven policy that dynamically allocates servers in response to time-varying demand that minimizes the long-run system cost subject to a specified service quality. This policy combines Singular Value Decomposition (SVD) method for forecasting, Stationary Dependent Period by Period (SDPP) paradigm to address the time-varying nature of the system, and simple queueing models with robust Chebyshev bounds for ensuring that service quality constraints are satisfied. We evaluate several competing policies with discrete event simulations using three-year data from the VCL of NC State University and show that our recommended policy achieves the target service quality using less than half of the servers under current policy.

Keywords: Chebyshev inequality, data driven staffing, dynamic server allocation, multi-type demand, SDPP, stochastics and statistics, SVD, time-varying arrivals.

1. Introduction

It is common to model service systems as queueing systems with customers belonging to multiple classes and multiple servers having multiple skill sets. These models are used to design the systems so that the costs are minimized while meeting service quality levels. The analytical queueing models typically assume time-invariant parameters such as the arrival rates, and the design algorithms produce time-invariant staffing solutions. However, there are two main problems with this methodology.

First, in practice the system parameters show time varying behavior, with daily and weekly cycles, for example. In such cases, the stationary model analysis is not very useful. Second, the parameters for the system are not known in practice, and have to be inferred

*Corresponding author
Email : vkulkarn@email.unc.edu

from the past data.

Several approaches have been developed independently to address these two issues and we shall discuss them in Section 2. However, it has now become feasible to develop solutions that will simultaneously address both of these issues. Such an approach is called "Data Driven System Design". In the context of service systems, it involves using past data about arrivals and service times to develop dynamic staffing policies to optimize a given performance criterion subject to service quality constraints. In this paper we shall illustrate this approach by applying it to a Virtual Computing Laboratory (VCL).

VCL is a cloud computing service that provides users remote access to their desired set of software applications. There are usually hundreds or even thousands of software applications that the users may choose from. Some examples of applications are "Matlab on Windows 7", "Maple on Windows XP", "Matlab and MS Excel on Windows 7", "Arena and CPLEX OPL", etc. The VCL was first developed at North Carolina State University (NC State) and is now an open-source project at the Apache Software Foundation - *http://vcl.apache.org*. An increasing number of institutions are hosting VCL servers. For example, UNC-Chapel Hill and NC State currently have hundreds of such servers for students, faculty, and researchers. From the perspective of modeling, we do not distinguish between the terms - "virtual computer" and "virtual machine"; instead we use the generic term "server". Each user is granted full control of the assigned server. If two users request the same software application, they will need two different servers loaded with that application.

A server in the VCL may be preloaded with a specific application, or left flexible. A user gets immediate access to a server preloaded with the desired application if one is available. Otherwise, the user has to wait for several minutes until a flexible server is loaded with the desired application.

In this paper we are concerned with the issue of deciding how many servers should be preloaded with which applications, and how many servers should be left flexible. We call the preloaded and flexible servers the *on* servers, and the rest of the servers the *off* servers that are turned off for cost saving (both energy and management). The service quality is measured by the fraction of the users who get immediate access, and the fraction that get delayed access, while the system performance (cost) is measured by the number of on servers.

Although our research is motivated by the VCL application, the methodology developed here is of general applicability in any service center with sufficiently flexible servers. It is also applicable to *software as a Service* (SaaS) and *Desktop as a Service* (DaaS) in cloud computing, see Kibe, *et al.* [27] . In such large-scale computing environments, the system performance is measured by the energy consumption, since it is important both economically and environmentally. The service quality is measured by the delay

experienced by the clients in getting access to the required resources (hardware as well as software). Hence, it is important and relevant to dynamically allocate "the right number of on servers with the right capabilities".

In many current implementations of VCLs such as those at UNC-Chapel Hill and NC State, all servers are always on and there are no flexible servers. For example, the VCL at NC State has about 800 servers and 1600 applications; the current policy ranks the applications by the frequency with which an application is requested, and each of the 400 most popular applications are preloaded on two servers (each preloaded server has exactly one application).

In general, assume the VCL has a total of M servers and is capable of handling N types of application. A user who desires type n $(1 \le n \le N)$ application is called a type n user. A type n user arriving at the system receives instant service if a server preloaded with application n is available. Otherwise, the user is delayed and needs to wait until the system manager (an automated software) chooses a k -preloaded server $(k \ne n)$, removes application k and loads application n. If no server is available then this user is blocked (rejected). After a user finishes the session, the system manager wipes the server clean, and then reloads it with the same or another application.

A server allocation plan (SAP) decides how many servers should be pre-loaded with which applications, how many servers should be kept flexible, and how many should be turned off. An SAP is called dynamic if these numbers change with time; otherwise it is called static. It makes sense to consider dynamic SAPs to accommodate time-varying demand rates. Such time-varying demand is an inherent feature of the VCL system. This is caused by the seasonal demand induced by the semesters, the days of week, and the time of the day. In this paper, we propose a data driven dynamic SAP with the objective of minimizing the system cost while achieving the targeted service quality.

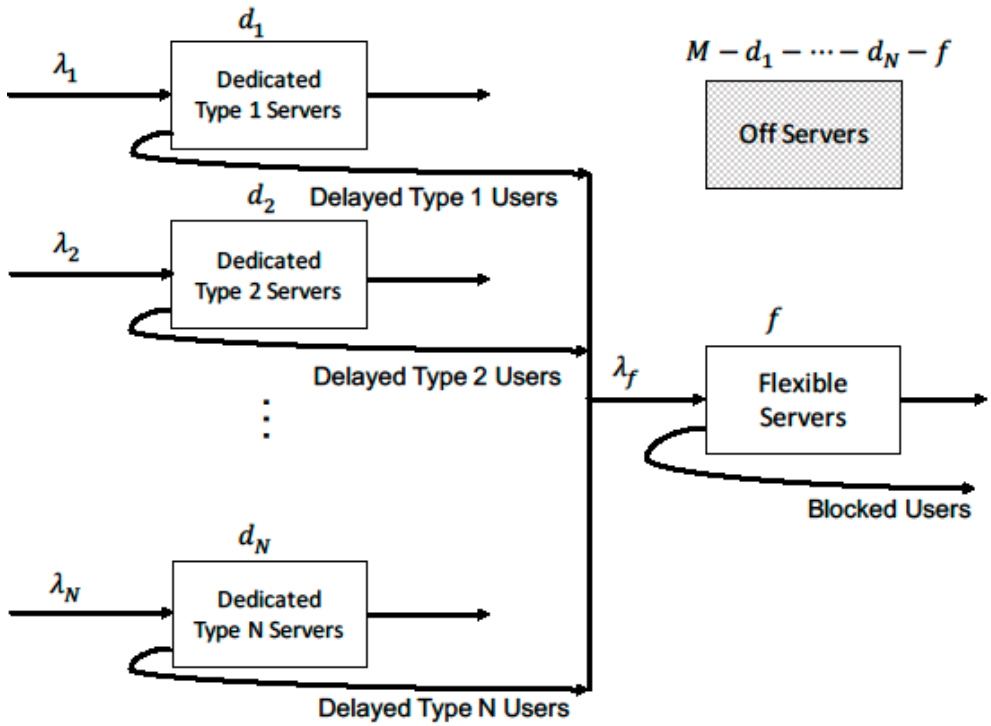

Figure 1. VCL Servers Network Diagram.

We shall begin with a static SAP assuming constant arrival rates. We keep d_n servers preloaded with application n (i.e. the dedicated type n server pool, $1 \le n \le N$), f servers flexible (i.e. the flexible pool), and $M - f - \sum_{n=1}^{N} d_n$ servers off. Figure 1 illustrates the server network diagram. When a type n arrival occurs (with rate λ_n), this user is given a server from the dedicated pool for application n, if one is available. Otherwise a server from the flexible pool is loaded with application n and assigned to this user. In the latter case the user needs to wait a few extra minutes for the loading operation. If no flexible servers are available, the user leaves the system without service, even if there are idle servers in the other dedicated pools. When a type n user finishes service from the dedicated pool, the released server is wiped clean and is reloaded with the same application to keep d_n a constant. Similarly, when a user finishes service from the flexible pool, the released server is left flexible to keep f a constant.

We then design a dynamic SAP based on the above static SAP. This is accomplished by dividing the whole time horizon into small periods, and implementing the static SAP over each period. Under dynamic SAP, the parameters d_n ($1 \le n \le N$) and f vary from period to period, but remain constant within each period. Thus the number of off servers will vary from period to period. We use probability bounds based on a stationary queueing model to bound the probabilities that an incoming user is delayed or blocked.

We compare two modeling approaches: the stationary independent period by period

(SIPP) approach, and our proposed stationary dependent period by period (SDPP) approach. These are described in Sections 4.3.1 and 4.3.2. It is clear that we need the forecasts of the arrival rates λ_n $(1 \le n \le N)$ for each period in order to execute our data driven dynamic SAP. We explore two statistical methods to forecast the future arrival rates: the moving average (MA) method and the singular value decomposition (SVD) method. They are described in Section 5. These arrival rate forecasts are then used to compute the bounds on the delay and blocking probabilities. In Section 6 we present the simulation results under different settings and recommend the best performing data driven dynamic SAP.

Our proposed policy uses a dynamic SAP under which at most 5% users are delayed and at most 0.5% of the service requests are blocked. The policy only uses a maximum of 391 on servers at any time, resulting in substantial energy savings. In contrast, Lee [30] shows that, under the current policy followed by the VCL at NC State, over 11% of the users are delayed and no users are blocked but all the available servers are always kept on. This makes our proposed policy extremely attractive.

The main contribution of this work is the development of a data driven server allocation procedure that results in an efficient allocation while meeting service criteria. It combines sophisticated forecasting, queueing analysis, robust bounds on service quality parameters, and a new method of addressing the non-stationarity and dependence, to devise an implementable algorithm.

The remainder of the paper is organized as follows. We provide a literature review on the related topics in Section 2. Section 3 introduces the structure of the data, and highlights the challenges posed by time-varying demands. We formulate our static SAP model in Section 4.1 and construct probability bounds in Section 4.2. Server allocation algorithms are developed there to ensure the service quality constraints. Section 4.3 explains our procedure of creating the data driven dynamic SAP based on the static SAP introduced in Section 4.2. Section 5 introduces two ways of forecasting future arrival rates, and Section 6 describes how we conduct discrete event simulations using real data and presents the results. It also makes recommendations about the best implementable policy, and discusses the managerial insights. Finally, Section 7 summarizes the paper and discusses how we can extend the current work.

2. Literature Review

One can think of the VCL as a server farm. There is a large literature on the topic of resource allocation in server farms. Gandhi *et al.* [13, 14, 15] defined four states of the servers: off, setup, idle, and on (busy). Comparing with our system, their idle servers are similar to our dedicated idle servers, where the user receives immediate service; their servers in setup state (switching from off to on) are similar to our flexible idle servers, where the

user has to wait for some extra setup time. Two performance measurements are usually considered in a server farm setting: waiting time and power consumption. In their work, Gandhi *et al.* used queueing models [15] and simulations [14] to derive these metrics. One of their conclusions is that keeping the servers idle is superior for reducing waiting time, and turning the servers off is superior for reducing power. Adan *et al.* [1] used a constant setup cost instead of a setup time to discourage switching between off and on, which simplified the state space and resulted in a switching-curve structure of the optimal policy.

Our model differs from the above server farm models in two important aspects. First, in the papers mentioned above, the users are homogeneous and the systems are stationary, while in our system, there are multiple types of users with time-varying arrival rates and service times. Second, the server farm literature deals with waiting times and power consumption, while our model focuses more on the fraction of delayed or blocking users. The use of dedicated servers is essential to reduce the fraction of delayed servers.

In this paper we address three main features of the VCL systems: (1) time-varying demands, (2) multi-type demand structure, and (3) availability of data to forecast future demand. We shall review the relevant literature below in each of these areas.

The phenomenon of time-dependent arrival is commonly seen in many service systems, and it is critical to staff them at appropriate levels to cope with this variation. There is a large literature dedicated to this problem. For an in-depth review on determining the staffing levels in the presence of time-varying demand, see Green *et al.* [19], Whitt [41], Liu and Whitt [31, 32, 33].

One approach to modeling the time-varying demand is to use stationary models in a non-stationary manner. It is achieved by dividing the working period (workday or workweek) into shifts, hours, quarter-hours, etc, and then applying a series of stationary queueing models over each planning period. This method is called the stationary independent period by period (SIPP) approach in Green *et al.* [18]. However its performance highly depends on the system parameters such as the arrival rate, the mean service time and the service quality, see Thompson [39] and Puhalskii and Reed [37]. As a counter example in Green *et al.* [18], when the Markovian model with sinusoidal arrival rates is considered in the simulation, the SIPP approach underestimates the staffing levels. Thompson [39] and Green *et al.* [18, 19] have addressed this issue and discussed several solutions such as a lagged SIPP, which essentially shifts the arrival rate curve to the right by a fixed amount.

In our paper, we apply approaches similar to SIPP to deal with the time-varying demand issue. Furthermore, we propose a modified stationary dependent period-by-period (SDPP) approach that takes into account the customers that remain in the system from the previous period. The SDPP approach outperforms the regular SIPP approach in the service quality while using fewer servers.

The second feature is the existence of the multiple types of users. This heterogeneity

in the sources of demands creates the critical issue of whether to use dedicated (specialized) or flexible resources. When the servers have sufficiently overlapping capabilities and work as a single super-server, the best possible performance can be achieved by the complete resource pooling strategy. See, for example, Harrison [20, 21]. Between the extremes of full-flexibility and full-specialization, different limited-flexibility structures can be constructed. Jordan and Graves [25] are the first to show that well-designed limited flexibility can be as good as full flexibility. These principles are further justified by Aksin and Karaesmen [3], Iravani *et al.* [23] and Bassamboo *et al.* [7]. They also propose methods to evaluate different flexibility structures. Our model considers the combination of dedicated server pools and a flexible server pool, in order to provide immediate service as much as possible while guaranteeing an overall service quality.

Next we address the statistical features of the VCL system. As pointed out by Chen and Henderson [11], where the staffing problem is studied under a Police Communication Center setting, designing a staffing level policy needs an accurate forecast of arrival rates. A handful of efficient forecasting approaches have been developed for call centers. For a comprehensive review, see Aksin *et al.* [2] and Ibrahim *et al.* [22]. Typically, arrival data in call centers are aggregated within each short time periods, such as 15-minute or 30-minute intervals, and the target of forecasting is implemented over such periods, see Jongbloed and Koole [24]. This is consistent with our SIPP and SDPP modeling approaches. Similar techniques are used in Weinberg *et al.* [40], where a multiplicative effects model is constructed to forecast Poisson arrival rates over intervals of 15, 30, or 60 minutes length, with a one-day lead time. More recently, Shen and Huang [38] propose a statistical model for forecasting call volumes within short time periods of a given day and also provide approaches to account for intraday forecast updating. Their singular value decomposition (SVD) based method outperforms existing forecasting methods. Our work adopts their SVD forecasting model to the VCL setting. Numerical experiments show that it leads to better service quality over the standard moving average (MA) method.

Finally we look at the (scant) literature on data driven staffing of service systems. Aktekin and Soyer [4] consider a Markovian queueing system with impatient customers with unknown arrival, service and impatient parameters. They develop a Bayesian procedure that updates the parameters as data becomes available and use stationary queueing models to do staffing. Bayesian procedures are typically slow, and require distributional assumptions that may not be justifiable. Recently, Gans *et al.* [16] have used SVD forecasting and stochastic programming to analyze the staffing problem. Bertsimas and Thiele [6] advocate a robust programming approach to handle the uncertainty in the parameter specification. Bassamboo and Zheevi [5] develop a two-stage stochastic programming approach to devise an algorithm for data driven staffing of a call center. They use fluid model of the service system to evaluate the performance measures.

This paper proposes an integrated solution that combines forecasting, queueing models, robust probability bounds, and a new approach to handle the non-stationarity and temporal dependence to devise an implementable solution to the server allocation problem.

3. Data

We had access to the VCL data set from NC State University containing information about all user requests from August 1, 2008 to July 31, 2011. In total there are 595,000 service requests for 1,643 different applications. For the ease of presentation, we sort the applications by their frequency of use in descending order. The usages vary considerably for different applications, where the top two applications account for 18.88%, the top ten applications account for 42.63%, and the top 400 applications account for 97.30% of the total requests. On the other hand, each of the bottom eight hundred applications is used no more than ten times over the three-year period that we consider. The detailed information is presented in Table 1. The VCL has around 700 to 900 servers. (The information about the exact number of servers is not given in our data set. The real time information about the number of on/off servers is given on the VCL website.) We present below some details of the arrival and service time data.

Table 1. Cumulative Relative Frequency of Arrivals.

Cumulative Frequency (%)	
Top 1	9.57%
Top 2	18.88%
Top 10	42.63%
Top 50	71.15%
Top 100	82.27%
Top 200	91.66%
Top 400	97.30%
Top 800	99.38%
Top 1643	100.00%

3.1. Arrivals

Figure 2 (a) plots the aggregated hourly arrivals from August 1, 2008 to July 31, 2011. To provide a better idea of arrival patterns in finer time scales, we use Panels (b) and (c) to show the average hourly arrivals in each hour of the week (starting with Sunday midnight) and each hour of the day (starting with midnight) respectively. We also present the same set of graphs for individual applications as comparison. For illustration, Figure 3 is shown here for Application 1, while Figures 11 and 12 (See Appendix A1) are for Applications 10 and

100.

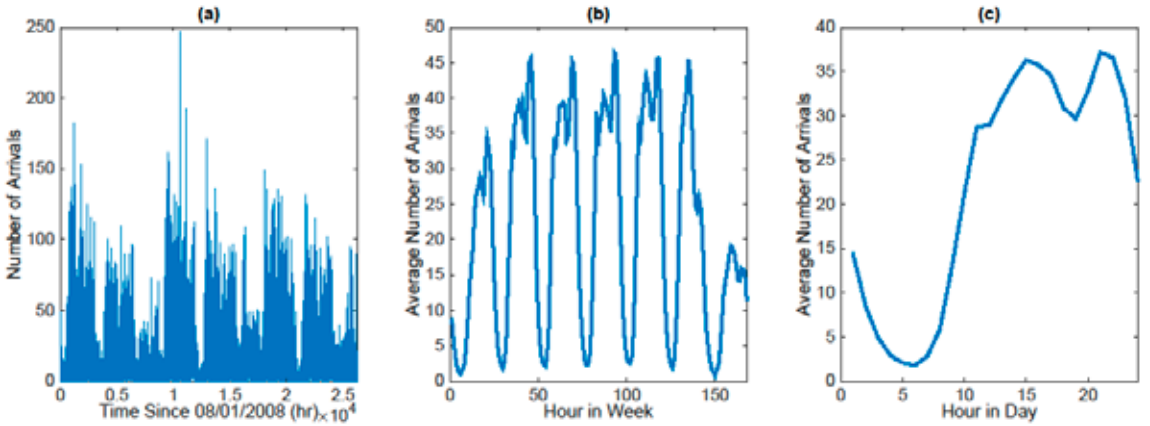

Figure 2. Aggregated Arrivals.

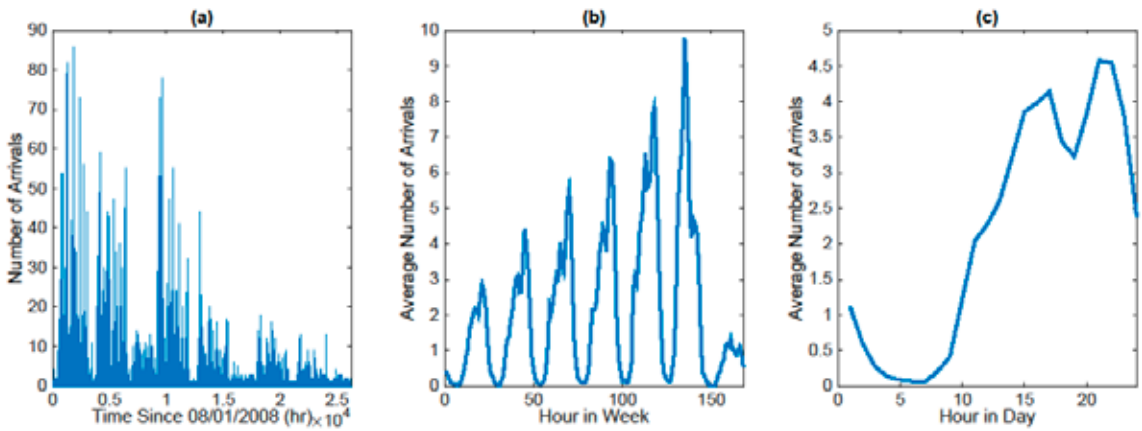

Figure 3. Arrivals of Application 1.

We observe that arrivals show a predictable, repeating pattern. The semesterly, weekly, and daily cycles are quite clear on both aggregated and individual application levels. In Figure 3(b) we see that for Application 1 the arrival volumes are low on Saturdays and high on Fridays. During the day, the first peak occurs around 3PM, followed by a second peak around 10PM. This is true in the aggregated case (Figure 3(c)) and for most of the applications. We observe that not all the applications were available in the VCL on the initial date of 08/01/2008. For example, Application 10 was not available until 08/18/2010 (Figure 11 in Appendix A1).

Using the aggregated arrival data we also plot the mean and standard deviation of the hourly arrivals against the time of day, grouped by the day of week, as shown in Figure 4. We observe the heteroscedasticity phenomenon - both the mean and the standard deviation depend on time. Besides, the magnitude of the standard deviation is almost at the same level of the mean; hence the variance exceeds the mean, which implies that the observations are

over-dispersed in comparison with a Poisson distribution.

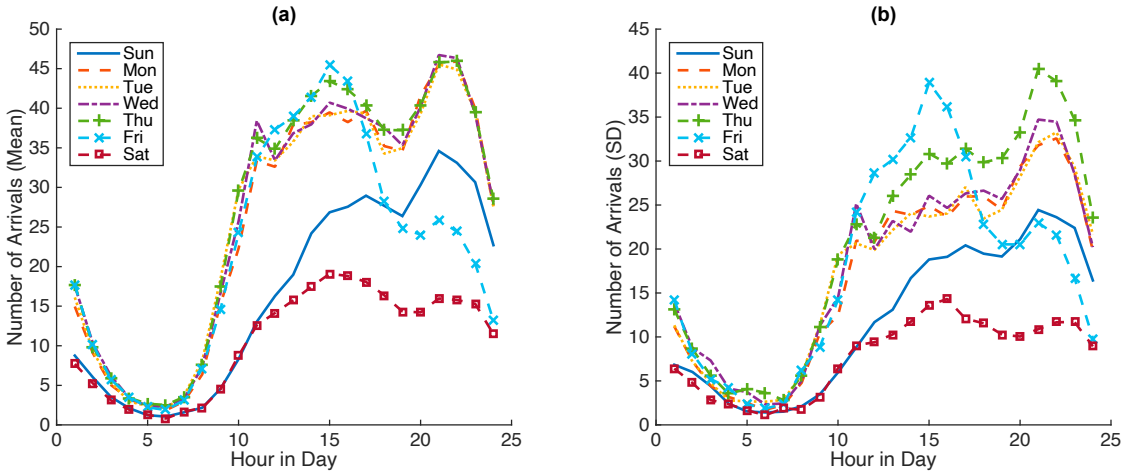

Figure 4. Mean and Standard Deviation of Aggregated Hourly Arrivals by Day of Week.

3.2. Service times

Running the analysis on the service time data, we find that about 3 % of the service times (i.e. 1327 arrivals) are less than 1 minute. The very short service times are questionable in the setting of software online service. One possible explanation is the accidental shut down of the system which would force the users to log off. Besides, there is a four-hour check initiated by the server, and if the user is not active, he is automatically logged off. If the user is active he can request extensions in two hour increments.

In Figure 5 we plot the empirical cumulative distribution function (CDF) of the service times for Applications 1, 10, 100, along with the corresponding CDF of the exponential distribution with the same mean. The plots suggest that an exponential assumption on the service time distribution is reasonable. (However, we do not need this exponential assumption in our queueing models.) Our exploration of the service time data showed that they do not vary with time to any meaningful level. So we shall take the service time distributions to be time-invariant.

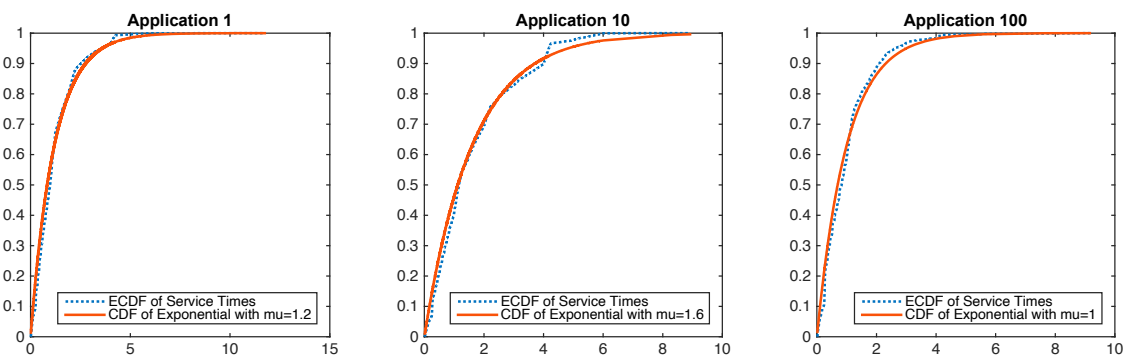

Figure 5. Empirical CDF of Service Times and Exponential CDF.

4. Allocation Policies

As described in Section 1, we consider a VCL with M servers, N applications, and N dedicated server pools (of sizes $d_1, d_2, \cdots, d_N$), one flexible server pool of size f, and the rest being off servers (Figure 1). In this section we develop server allocation plans (SAP) for the VCL. In Section 4.1, we first assume that the arrival rates for the applications are constant, and present the formulation of a static SAP. Then in Section 4.2, we quantify the service quality constraints for the VCL system and derive the "optimal" static SAP. In Section 4.3, we extend the static SAP to dynamic SAP using stationary independent period by period (SIPP) approach and the stationary dependent period by period (SDPP) approach.

4.1. Problem formulation-static SAP

Under the static problem we assume that the arrival processes of customers of different type are independent Poisson processes with fixed rate parameter, namely, λ_n for type n, and need independent and identically distributed (iid) service times with mean s_n. Under the static SAP, we are interested in a policy which assigns d_n dedicated servers preloaded with application n $(1 \le n \le N)$, allows f servers to be left flexible, and turns off the remaining $M - f - \sum_{n=1}^{N} d_n$ servers. The aim is to use as few servers (dedicated or flexible) as possible while satisfying the service quality constraints (to be quantified below).

Note that when a user gets a dedicated server, the wiping and reloading operations take place at the end of the service (after the user leaves the system); if a user gets a flexible server, the loading operation takes place before the service starts and the wiping operation takes place after the service finishes. Thus in both cases the service times are augmented by the wiping and loading operations.

Let the delay probability α be the fraction of the users who do not receive immediate service upon arrival, and the blocking probability β be the fraction of the users who leave the system without service. Our aim is to identify the smallest $d_1, \cdots, d_N$ and f that can satisfy the following service level constraints:

$$\alpha < \alpha^*, \tag{1}$$

$$\text{and } \beta < \beta^*, \tag{2}$$

where α^* and β^* are the design parameters. For example, we consider $\alpha^* = 0.05$ and $\beta^* = 0.005$. Below we describe a queueing model to quantify the above service quality constraints.

4.2. Service constraints under static SAP

In this section we derive an upper bound on the delay probability and the blocking probability. We use this close form upper bound to quantify the constraints given in (1) and (2), and derive an algorithm for the sizing problem that achieves the local optimal.

If a type n preloaded dedicated server is available when a type n user arrives, its

service starts immediately and lasts for an amount of time with mean s_n plus the wiping and loading time with mean u_n. Note that we do not need to make a specific assumption about the distribution of service time. If such a preloaded server is not available, then the user is delayed (or lost).

• *Dedicated Pool.*

We model the dedicated pool of type n as an $M/G/d_n/d_n$ queue with mean arrival rate λ_n and mean service time s_n+u_n. We know that in steady state the number in this system is bounded above by the number in an $M/G/\infty$ system with the same arrival and service parameters. Let X_n be the number in the $M/G/\infty$ system in steady state. We know that X_n is a Poisson random variable with mean (and variance) given by

$$a_n=\lambda_n(s_n+u_n).$$

Using Chebyshev's inequality, we obtain the following result for any $x\geq 0$:

$$\begin{aligned} P(X_n>x) &\leq \frac{Var(X_n)}{\left(x-E(X_n)\right)^2} \\ &= \frac{a_n}{(x-a_n)^2}. \end{aligned}$$

Clearly, we should use 1 as the upper bound if the bound above is greater than one. Hence the probability that an incoming arrival of type n is delayed is bounded above by

$$P_d(d_n,a_n)=\min\left(\frac{a_n}{(d_n-a_n)^2},1\right).$$

Now, the probability that an arrival is of type n is $\lambda_n/\sum_{k=1}^{N}\lambda_k$. Hence the upper bound of the overall probability of delay is given by

$$\frac{\sum_{n=1}^{N}\lambda_n P_d(d_n,a_n)}{\sum_{n=1}^{N}\lambda_n}, \tag{3}$$

which provides an upper bound on α. Hence we formulate the following optimization problem for the dedicated pools:

Problem P1

$$\text{minimize } \sum_{n=1}^{N} d_n$$

$$\text{subject to } \frac{\sum_{n=1}^{N} \lambda_n P_d(d_n, a_n)}{\sum_{n=1}^{N} \lambda_n} < \alpha^*, \quad d_n : \text{nonnegative integers.}$$

We first create a relaxation of **P1** by ignoring the integrality of the d_n's, and call this problem **P1R**:

Problem P1R

$$\text{minimize } \sum_{n=1}^{N} d_n$$

$$\text{subject to } \frac{\sum_{n=1}^{N} \lambda_n P_d(d_n, a_n)}{\sum_{n=1}^{N} \lambda_n} < \alpha^*, \quad d_n \geq 0.$$

Note that the expression in (3) is not a convex function in d_n, hence we introduce the following variation **P1R*** of **P1R**, and solve this **P1R*** using Lagrangian formulation, and then modify its solution to obtain the local optimal solution to **P1R**. For algebraic simplification, let

$$x_n = d_n - a_n, \quad f_n(x) = \frac{a_n}{x^2}, \quad n = 1, 2, \ldots, N.$$

Problem P1R*

$$\text{minimize } \sum_{n=1}^{N} x_n$$

$$\text{subject to } \frac{\sum_{n=1}^{N} \lambda_n f_n(x_n)}{\sum_{n=1}^{N} \lambda_n} < \alpha^*, \quad x_n \geq 0.$$

From Lagrangian formulation, we see that there exists a constant $c \geq 0$ such that the optimal x_n solves

$$\lambda_n f_n'(x_n) = -c, \quad n = 1, 2, \cdots, N, \tag{4}$$

which yields

$$x_n(c)=\sqrt[3]{\frac{2\lambda_n a_n}{c}}.$$

To account for the min operator, define

$$g_n(x)=\min(f_n(x),1),\quad n=1,2,\ldots,N.$$

Choose a c such that

$$G(c)=\frac{\sum_{n=1}^{N}\lambda_n g_n\left(x_n(c)\right)}{\sum_{n=1}^{N}\lambda_n}=\alpha^*.$$

Note that $G(\cdot)$ is an increasing function, and with

$$u=\max\{\frac{2\lambda_n}{\sqrt{a_n}}:\ 1\le n\le N\},$$

we have

$$G(0)=0,\quad G(u)=1.$$

Hence we can use bisection method to find a $c=c(\alpha^*)$ that solves

$$G(c)=\alpha^*.$$

Now define

$$x_n(\alpha^*)=\sqrt[3]{2\lambda_n a_n / c(\alpha^*)}.$$

and

$$y_n=\begin{cases}x_n(\alpha^*), & if\ \ a_n/x_n^2(\alpha^*)<1,\\ 0, & if\ \ a_n/x_n^2(\alpha)\ge 1.\end{cases}\tag{5}$$

This leads to

$$d_n=\begin{cases}a_n+x_n(\alpha^*), & if\ \ a_n/x_n^2(\alpha^*)<1,\\ a_n, & if\ \ a_n/x_n^2(\alpha)\ge 1.\end{cases}\tag{6}$$

Now we present the theorem to show the local optimality of the solution given in (6).

Theorem 1. *The allocation* $\{d_n,1\le n\le N\}$ *produced by (6) is a local optimal solution to the optimization problem* **P1R**.

Proof. Let $y=[y_n,1\le n\le N]$ be the allocation from (5), and let

$$Z=\{n:y_n=0\}.$$

Using $g_n(0)=1$, we have

$$H(y)=\sum_{n=1}^{N}\frac{\lambda_n}{\sum_{n=1}^{N}\lambda_n}g_n(y_n)=\sum_{n\in Z}\frac{\lambda_n}{\sum_{n=1}^{N}\lambda_n}+\sum_{n\notin Z}\frac{\lambda_n}{\sum_{n=1}^{N}\lambda_n}f_n(x_n)=\alpha^*.$$

Now consider a feasible $y'=y+h$, where h is a small deviation such that y' is a strict improvement over y, that is

$$\sum y'_n<\sum y_n.$$

This means we must have $h_n\geq 0$ for $n\in Z$ to maintain feasibility. Then, using (4), we get

$$H(y')-H(y)=\sum_{n\notin Z}\frac{\lambda_n}{\lambda}f_n'(x_n)h_n=-c\sum_{n\notin Z}h_n+o(h). \tag{7}$$

Now

$$0>\sum y'_n-\sum y_n=\sum h_n=\sum_{n\in Z}h_n+\sum_{n\notin Z}h_n.$$

This implies

$$\sum_{n\notin Z}h_n<-\sum_{n\in Z}h_n\leq 0.$$

However, in that case (7) implies

$$H(y')>H(y).$$

Thus y' cannot be feasible. This shows that y is a local optimum.

We believe that this is also a global optimal to **P1R**, but have not been able to show it. In practice we can round up or round the non-integer solution d_n given in (6) to get an integer solution. This may not be the optimal solution to the integer constrained problem **P1**, but will be reasonable for our purposes.

• *Flexible Pool.*

We see that the arrival process to the flexible pool is a superposition of the overflows from a large number of the dedicated pools. Hence, we choose to simply approximate the aggregate arrival process to the flexible pool by a Poisson Process (PP). Let λ_n^f be the rate of overflow from n th dedicated server pool and the arrival rate to the flexible pool be

$$\lambda^f=\sum_{n=1}^{N}\lambda_n^f.$$

The mean service time in the flexible pool is a weighted sum of the mean service times of each type n plus the mean loading time:

$$s_f=\frac{\sum_{n=1}^{N}\lambda_n^f(s_n+u_n)}{\sum_{n=1}^{N}\lambda_n^f}.$$

Then the offered load to the flexible pool is given by

$$a_f = \lambda^f s_f = \sum_{n=1}^{N} \lambda_n^f (s_n + u_n).$$

Again, assuming the system is in steady state, the probability of a user being blocked from the flexible pool can be bounded from above using Chebyshev's inequality:

$$P_b(f, a_f) = \frac{a_f}{(f - a_f)^2},$$

where f is the number of busy flexible servers. Note that the probability that a user leaves the system without service is equal to the probability that a user is blocked from the flexible pool. We call it the *blocking probability*. Hence, to satisfy (2), we solve the following optimization problem:

Problem P2

$$\begin{aligned} &\text{minimize } f \\ &\text{subject to } P_b(f, a_f) = \frac{a_f}{(f - a_f)^2} \le \beta^*. \end{aligned} \tag{8}$$

The above problem can be explicitly solved to get the optimal f as

$$f^* = a_f + \sqrt{a_f / \beta^*}.$$

We use the ceiling of f^* as the number of flexible servers to use.

4.3. Dynamic SAP

As observed in Section 3, the user arrival rates are highly time dependent. Hence we introduce a dynamic version of SAP, which adjusts the sizes of the server pools in response to changing arrival rates. The main idea is to divide the entire time horizon into small planning periods.

Let $\{\tau_i, i \ge 1\}$ be a given increasing sequence starting with $\tau_1 = 0$. For example we use $\tau_i = i - 1$ ($i \ge 1$, in hours). We call the interval $[\tau_i, \tau_{i+1})$ the i th period. We replace the notations such as λ_n, s_n, a_n with the notations $\lambda_{n,i}$, $s_{n,i}$, $a_{n,i}$, which represent the arrival rate, service time, and offered load respectively, of application n over the i th period. Then the sizes of the dedicated server pools $(d_{1,i}, \cdots, d_{N,i})$ are determined by (6) over the i th period. Similarly, for the flexible pool, we use notation $\lambda_{n,i}^f$, λ_i^f, $s_{f,i}$ and $a_{f,i}$ instead of λ_n^f, λ^f, s_f and a_f for the i th period, and we determine the size f_i period by period. The dynamic SAP is defined by $\{(f_i, d_{1,i}, \cdots, d_{N,i}), i \ge 1\}$.

There is one detail we have to take care of in the dynamic setting: transition from one period to the next. Let $B_n(t)$ be the number of type n users who are using servers from their dedicated pool at time t. Thus $B_n(\tau_i)$ is the number of dedicated servers serving type

n users at the beginning of the i th period. If $d_{n,i} \geq B_n(\tau_i)$, we allocate $d_{n,i} - B_n(\tau_i)$ additional servers to the dedicated pool of type n. If $B_n(\tau_i) > d_{n,i}$ no new arrivals of type n are allowed in the dedicated pool until $B_n(t)$ reduces to $d_{n,i}$. After that new arrivals of type n are accepted as long as $B_n(t)$ does not exceed $d_{n,i}$ over the i th interval. Thus the actual number of dedicated severs at time t is given by $\max\{B_n(t), d_{n,i}\}$ for $t \in [\tau_i, \tau_{i+1})$. We follow a similar transition procedure for the flexible pool.

One final question we need to resolve is how to estimate the arrival rates $\lambda_{n,i}$ and λ_i^f. We discuss two procedures below. The traditional stationary independent period-by-period (SIPP) methodology assumes that the smaller planning periods are independent, and the system is in steady state over each period (see Section 4.3.1). Due to the relatively longer service times (compared to the lengths of the periods) at the VCL, the SIPP approach may not be suitable. Hence we modify this approach to account for the dependence between the consecutive periods. We call this the Stationary Dependent Period by Period (SDPP) approach (see Section 4.3.2).

The overflow rates $\lambda_{n,i}^f$ and mean service time $s_{f,i}$ of the flexible pool are not part of the data. They depend on the data and the server allocation policy. Hence we estimate them by actually measuring the number of overflows under a given policy in our simulation. We discuss this in more detail in Section 6.

4.3.1. Stationary independent period by period (SIPP)

Let the $\lambda_n(t)$ be the arrival rates at time t for type n application. The SIPP approach approximates the arrival rate $\lambda_{n,i}$ of application n over the i th period by

$$\hat{\lambda}_{I(n,i)} = \frac{1}{\tau_{i+1} - \tau_i} \int_{\tau_i}^{\tau_{i+1}} \lambda_n(t) dt.$$

Since we assume that $\lambda_n(t)$ is a constant equal to $\lambda_{n,i}$ over $[\tau_i, \tau_{i+1})$, we have

$$\hat{\lambda}_{I(n,i)} = \lambda_{n,i}. \tag{9}$$

For the flexible pool, we forecast the overflow rates $\lambda_{n,i}^f$ on $[\tau_i, \tau_{i+1})$ from dedicated pool n and by assuming constant rate over this period we also have

$$\hat{\lambda}_{I(i)}^f = \sum_{n=1}^{N} \lambda_{n,i}^f. \tag{10}$$

4.3.2. Stationary dependent period by period (SDPP)

As discussed in Section 3, typical service times range from 30 minutes to four hours. Hence many of the arrivals in one interval would continue to be in the system over the next interval due to relatively long service times compared to the length of the interval. This makes the independent assumption of SIPP approach less effective (Green *et al.* [17]).

Hence, we propose the SDPP approach that accounts for the dependence between the consecutive intervals (see similar treatment in Massey and Whitt [34]).

Let S_n be the representative service time of a type n user. If a user arrives at the system during $[\tau_i - S_n, \tau_{i+1})$, the sojourn time in the system will overlap $[\tau_i, \tau_{i+1})$. Hence we adjust the arrival rate of type n over the i th period to

$$\begin{aligned}\lambda_{D(n,i)} &= \int_{\tau_i - S_n}^{\tau_{i+1}} \lambda_n(t)dt / (\tau_{i+1} - \tau_i + S_n) \\ &= \left[\int_{\tau_i - S_n}^{\tau_i} \lambda_n(t)dt + \int_{\tau_i}^{\tau_{i+1}} \lambda_n(t)dt\right] / (\tau_{i+1} - \tau_i + S_n) \\ &= \left[\int_{\tau_i - S_n}^{\tau_i} \lambda_n(t)dt + (\tau_{i+1} - \tau_i)\lambda_{n,i}\right] / (\tau_{i+1} - \tau_i + S_n)\end{aligned}$$

Since we do not assume any specific distribution of service times, even the expectation of the integral in the above equation

$$\int_{\tau_i - S_n}^{\tau_i} \lambda_n(t)dt \tag{11}$$

is intractable in general. However, we can interpret (11) as the total number of arrivals over $[\tau_i - S_n, \tau_i)$, and since S_n is a representative service time, all these arrivals are expected to be in the system at time τ_i. Let $F_n(t)$ be the number of type n users who are using servers from the flexible pool at time t. Thus $B_n(t) + F_n(t)$ is the number of type n users in the system at time t. Hence we approximate the integral by $B_n(\tau_i) + F_n(\tau_i)$, the actual number of type n users in the system at time τ_i. More specifically we approximate $\lambda_{D(n,i)}$ by

$$\hat{\lambda}_{D(n,i)} = [(\tau_{i+1} - \tau_i)\lambda_{n,i} + B_n(\tau_i) + F_n(\tau_i)] / (\tau_{i+1} - \tau_i + s_n), \tag{12}$$

where s_n is again the mean service time of Application n.

Similarly, for the flexible pool we have

$$\hat{\lambda}^f_{D(i)} = [(\tau_{i+1} - \tau_i)\lambda^f_i + F_n(\tau_i)] / (\tau_{i+1} - \tau_i + s^f_i), \tag{13}$$

Note that $\lambda_{n,i}$ in (12) and λ^f_i in (13) are forecasts. However, the quantities $B_n(\tau_i)$ and $F_n(\tau_i)$ depend on the actual evolution of the system. This is what makes the estimates $\{\hat{\lambda}_{D(n,i)}, i \geq 1\}$ and $\{\hat{\lambda}^f_{D(i)}, i \geq 1\}$ dependent. This justifies the name *Stationary Dependent Period by Period* for this dynamic procedure.

We now use these estimates to compute the server allocation $\{(f_i, d_{1,i}, \cdots, d_{N,i}), i \geq 1\}$ as described in the Section 4.2.

5. Forecasting Future Arrival Demand

To implement the dynamic SAP in practice, we need to forecast future arrival demand, denoted as $\lambda_{n,i}$ in the above sections. Here we focus on two forecasting methods that will

be compared in our numerical study section. The baseline method is the moving average (MA) method, and the more sophisticated one is the singular value decomposition (SVD) method introduced by Shen and Huang [38].

We introduce the common notation used in this section. Let T be the number of days. Each day is divided into P periods. For example, we use $P=24$ hourly periods in our numerical experiments. Let $x_{t,i}$ be the number of arrivals during the i-th period of day t, $t=1,\cdots,T$, $i=1,\cdots,P$. The vector $x_t=[x_{t1},\cdots,x_{tP}]$ records the hourly arrival volumes on day t. Given the historical data $x_1,\cdots,x_T$, the following sections discuss the two methods to forecast the next-day demand vector x_{T+1}.

5.1. Moving average forecasting

Let w be the rolling horizon, which is the number of historical days used for forecasting. Specifically, considering the daily patterns in our data, we forecast the arrival demand over each i th period of day $T+1$ as the average of the same time periods of the previous w days. That is

$$\hat{x}^{(1)}_{T+1,i}=\frac{1}{w}\sum_{t=T-w+1}^{T}x_{t,i},\quad i=1,\cdots,P,\quad T>w.$$

One important issue in the MA method is to decide the size of the rolling horizon w. The larger the window, the less influence the short term daily fluctuation will have, and more clearly we can see the long term effects. However, a larger w would make the MA method less sensitive to the non-stationary phenomenon. Our numerical studies use $w=30$ days.

5.2. SVD forecasting

Here we apply the inter-day forecasting method introduced in Shen and Huang [38]. To be consistent with the MA method, we also use w historical days. Let $X=[x_{T-w+1}^{\top},\cdots,x_T^{\top}]^{\top}$ be the $w\times P$ historical data matrix used in the forecasting of arrivals on day $T+1$ (here the sign $^{\top}$ stands for transpose). The singular value decomposition (SVD) of the matrix X can be expressed as $X=USV^{\top}$, where U is a $w\times P$ matrix that records the row information (the daily (inter-day) pattern of the original X), and V is a $P\times P$ matrix that records the column information (the time of day (intra-day) pattern of the arrivals). We write the three decomposition matrices in the form of column vectors and diagonal elements: $U=(u_1,\cdots,u_P)$, $V=(v_1,\cdots,v_P)$, $S=diag(s_1,\cdots,s_P)$, where $s_1\geq s_2\geq\cdots\geq s_P$. Then it follows

$$x_t=(s_1u_{t,1})v_1^{\top}+\cdots+(s_Pu_{t,P})v_P^{\top},\quad t=T-w+1,\cdots,T.$$

To summarize the inter-day features while reducing the dimension of data profiles, we extract the first K singular vectors from V, which is similar to the techniques used in the principle component analysis. By setting $s_ku_{t,k}=\beta_{t,k}$, we get the following approximation

$$x_t \cong (s_1 u_{t,1}) v_1^\top + \cdots + (s_K u_{t,K}) v_K^\top$$
$$= \beta_{t,1} v_1^\top + \cdots + \beta_{t,K} v_K^\top, \quad t = T - w + 1, \cdots, T.$$

The last step is to forecast arrivals on day $T+1$. Since there are obvious day of week patterns shown in the data, we include $z_i = 1, \cdots, 7$ as a categorical covariate controlling for the day of week effects in the AR(1) time series model to obtain $\beta_{T+1,k}$, $1 \le k \le K$:

$$\beta_{T+1,k} = a_k(z_{T+1}) + b_k \beta_{T,k} + \epsilon_{T+1,k}.$$

Next, the arrival vector on day $T+1$ is modeled as

$$x_{T+1} = \beta_{T+1,1} v_1^\top + \cdots + \beta_{T+1,K} v_K^\top + \epsilon_{T+1},$$

whose mean can be estimated by

$$\hat{x}_{T+1}^{(2)} = \beta_{T+1,1} v_1^\top + \cdots + \beta_{T+1,K} v_K^\top.$$

We use the first semester arrival rates of application 1, 3 and 4 as test data to obtain the scree plots (Shen and Huang [38]) in Figure 6, and find that $K =$ 2 or 3 gives good forecasting results.

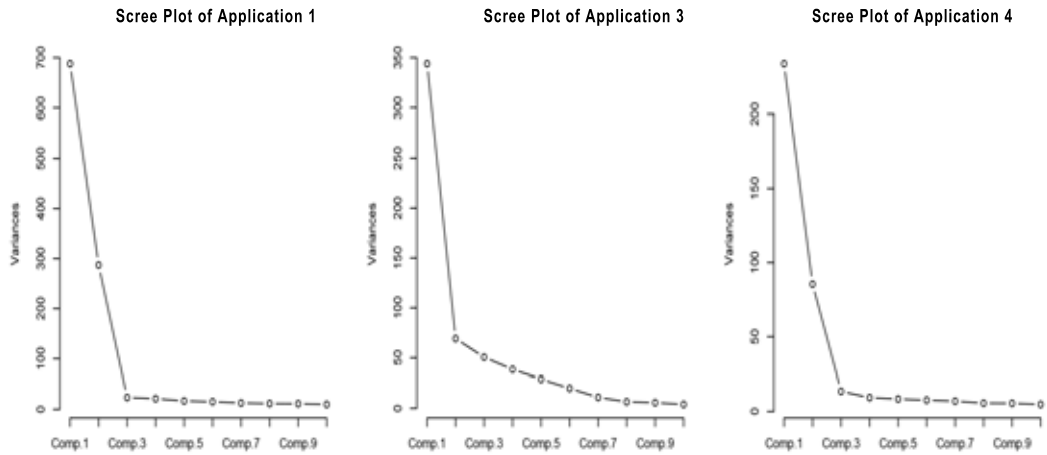

Figure 6. Scree Plots.

Note that the SVD method involves regression on the decomposed data matrix, which is not efficient when the historical arrival volumes are close to zero. This happens to be the case for applications ranked 100 and lower in terms of total requests, whose hourly arrival rates are usually less than one. We therefore pool the lower ranked applications into few groups, and forecast the aggregated arrival rates within each group. We then split the group total arrival forecast using the Bernoulli splitting rule, and obtain the hourly arrival rates for each individual application.

In the rest of the paper, we use $\lambda_{n,i}^{(j)}$ to denote the forecasting arrival rate of application n over the i th period using the j th forecasting method illustrated above, and $\lambda_{n,i}^{f(j)}$ as the forecast for the overflow into the dedicated pool ($j=1,2$ as in the notations $\hat{x}_{T+1}^{(j)}$).

It is possible to compute the prediction intervals for both methods. We do not discuss it here since we do not use it.

6. Numerical Experiment and Managerial Insights

In this section, we report the numerical experiments conducted to implement and compare the performance of the dynamic SAP method. The queueing model ($M/G/\infty$) assumes that the arrival process is Poisson over each period. We use statistical tests from Brown *et al.* [8] to validate this assumption, and show that overall there is no evidence to reject the null hypothesis that the arrival process is NHPP with piecewise constant (PC) arrival rates. See the Appendix A2 for details.

To implement the dynamic SAP, we forecast arrival rates $\lambda_{n,i}$ using the methods introduced in Section 5. We assume that the mean service times $s_{n,i}$ are time-independent and simply take the sample mean s_n for application n as the estimate of $s_{n,i}$ for all the periods.

We simulate the system with the trace data from the VCL to compare the efficacy of all the methods introduced earlier. We run four simulation setups:

1. SVD + SIPP: Use SVD forecasting to generate the arrival rate forecasts. Use SIPP to compute the allocation vectors for each period.

2. SVD + SDPP: Use SVD forecasting to generate the arrival rate forecasts. Use SDPP to compute the allocation vectors for each period.

3. MA + SIPP: Use MA forecasting to generate the arrival rate forecasts. Use SIPP to compute the allocation vectors for each period.

4. MA + SDPP: Use MA forecasting to generate the arrival rate forecasts. Use SDPP to compute the allocation vectors for each period.

As a general parameter setting, we assume the targeted global probability of delay α^* is 5%, and the targeted global blocking probability β^* is 0.5%. There are 400 dedicated pools and one flexible pool. To forecast the arrival rates, we choose the length of rolling horizon $w=30$ days, and assume that there are $P=24$ hourly periods in a day. Applying methods described in Section 5 for each dedicated pool, we obtain two different forecasts: $\lambda_{n,i}^{(1)}$ using the MA method, and $\lambda_{n,i}^{(2)}$ using the SVD approach. Besides, we set loading time $u_n=5$ mins for all the applications.

To determine the number of dedicated servers assigned over the i th interval, we let $\tau_i=i-1$ ($i\geq 1$, in hours), and record the number of type-n users using the dedicated servers ($B_n(i)$) and the flexible servers ($F_n(i)$) at the beginning of the i th period (same as at the

end of the previous period). Then the SIPP method and the SDPP method yield the following estimates ($j=1,2$ for the MA and SVD, respectively):

$$\hat{\lambda}_{I(n,i)} = \lambda_{n,i}^{(j)},$$

$$\hat{\lambda}_{D(n,i)} = [\lambda_{n,i}^{(j)} + B_n(i) + F_n(i)]/(1+s_n).$$

Similarly, for the flexible pool we collect the overflow data of past 30 days generated in the simulation, and use the corresponding forecasting and estimation approaches as the ones for the dedicated pool to obtain $\hat{\lambda}_{I(i)}^{f}$, $\hat{\lambda}_{D(i)}^{f}$ as described in Sections 4.3 and 5.

Finally we compute the allocation vector $\{(f_i, d_{1,i}, \cdots, d_{N,i}), i \geq 1\}$ as described in Section 4. The performance under the 4 simulation setups is shown in the figures below. We present the average probability of delay and blocking on the left, and the average number of on servers on the right, both of which are plotted against the hour in a day.

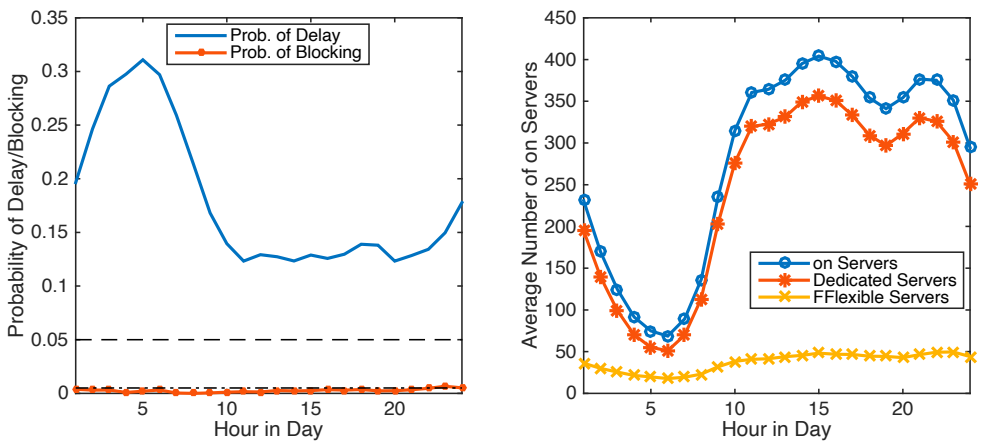

Figure 7. Simulation Results with MA + SIPP.

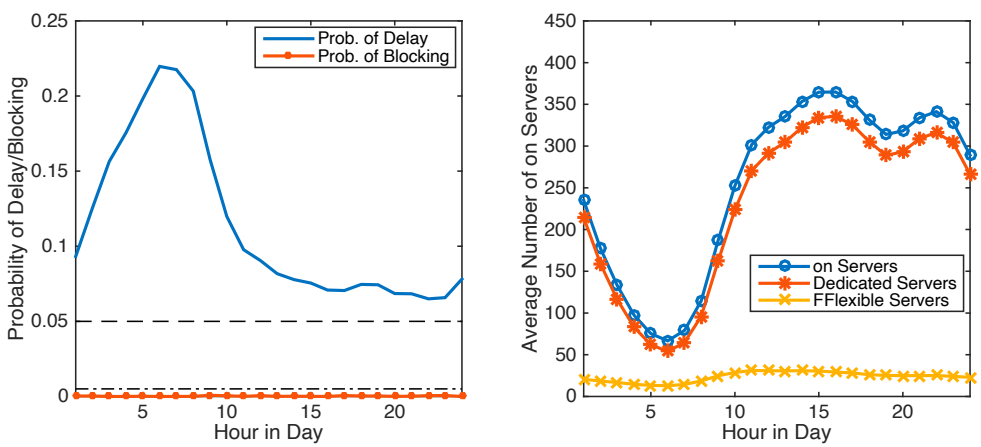

Figure 8. Simulation Results with MA + SDPP.

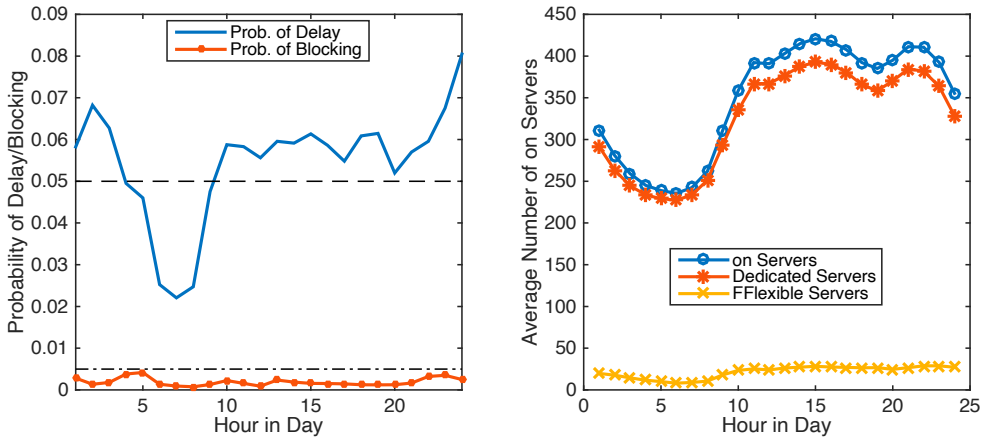

Figure 9. Simulation Results with SVD + SIPP.

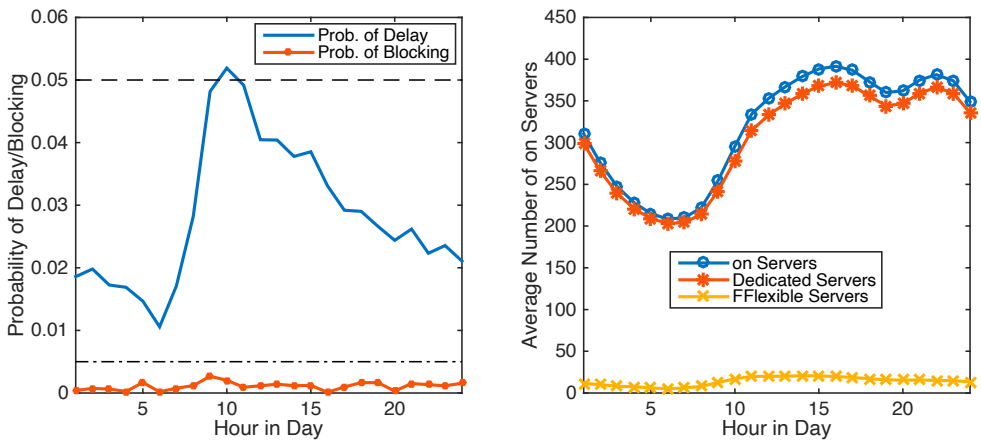

Figure. 10. Simulation Results with SVD + SDPP.

We first compare the performance between the SIPP and SDPP approaches, that is, we compare Figure 7 with Figure 8, and Figure 9 with Figure 10. Under both MA and SVD forecasting settings, we find that SDPP generally allocates less servers than SIPP and at the same time achieves better service quality, that is, significantly lower probabilities of delay and blocking.

Among the four settings, only the SVD+SDPP combination (Figure 10) closely achieves the target 5% probability of delay and 0.5% blocking probability. On the other hand, the SVD+SIPP (Figure 9) approach induces delay probability of as high as 8% towards the end of the day. The performance of blocking probability under the SVD+SIPP approach is comparable to that of the SVD+SDPP approach. Besides, SDPP uses less servers during both peak and off-peak hours than SIPP. The average hourly on servers under SDPP is 318, and the number is 347 under SIPP. The maximum number of servers used during a day is 391 for the SDPP and 421 for the SIPP. This is achieved by considering the

dependence over periods, which is also reflected in the smoother and right shifted curve in the right panel in Figure 10 comparing with the right panel in Figure 9. The number of on servers is significantly less than the number of available servers (700 to 900), implying a significant reduction in terms of energy consumption and management cost over the current operating procedure.

We next compare the performance of the MA and SVD forecasting methods, that is to compare Figure 7 with Figure 9, and Figure 8 with Figure 10. Under both SIPP and SDPP settings, the SVD approach far outperforms the MA forecasting. For example, in the MA+SDPP setting, the MA approach leads to severe under-allocation of servers, hence the serious downgrading of service quality. The delay probabilities experienced by users approach 20%. This leads to overestimates of overflow rates and over-allocation to the flexible pool, hence the near zero blocking probabilities. The number of on servers at the peak is only marginally smaller than the one under the SVD+SDPP combination. It is clear that under the MA approach, VCL is understaffed over the off-peak hour from 3am-10am, which is why the delay probability significantly exceeds the target 5%. In comparison, the SVD approach does a great job in forecasting arrival rates, since it takes into account inter-day dependence, as well as day-of-week effects.

In summary, MA performs much worse compared to SVD, and SDPP performs better than SIPP. Hence we recommend the SVD+SIPP combination in practice. To be precise, we recommend the use of the data driven dynamic SAP with the following steps:

1. Use SVD forecasting with a rolling window of thirty days,
2. Use SDPP to estimate the arrival rate parameters,
3. Use the allocation algorithms of Section 4 to decide the server allocations.

Managerial Implications: Although the above study was carried out using the VCL dataset from NC State over 2008-2011, our analysis yields several important managerial implications for the optimal operation of a service system with multiple flexible servers and multiple classes of customers with time varying arrival rates. First, it is important to accurately forecast the demand rates using statistical tools that account for the cyclic patterns and trends in the demand. For example, we concluded that SVD works better than MA. It is equally important to make the server allocation decisions dynamically, in response to the current trajectory of the system. For example, we discovered that SDPP works better than SIPP. Ignoring these two aspects can lead to serious degradation of performance.

7. Summary and Extensions

Motivated by the server allocation problem in the VCL, we consider a system using two types of server pools – the dedicated pools where pre-determined types of applications

are preloaded on certain dedicated servers, and the flexible pool where different applications are loaded on demand. We formulate an optimization problem to minimize the number of on servers, subject to pre-specified service level constraints. The service level includes the probabilities of a user being delayed or blocked from the system. We derive a method to quantify the service quality constraints, and develop an algorithm to identify the corresponding static SAP which is further extended to a dynamic SAP. We extend the SIPP approach and propose the SDPP approach to handle the time-varying demand and long service times.

We then consider two methods – MA and SVD – to forecast future arrival rates given historical data. We evaluate the performance of the dynamic SAP by conducting discrete event simulation experiments using real data. We run statistical tests to justify the assumption of a Poisson arrival process with the piecewise constant arrival rate function over the planning horizon.

Overall, our recommended dynamic SAP keeps no more than 400 servers on during the peak hours and less than 210 servers on during the off-peak hours, which is a significant saving over the 700-900 servers currently being kept on by the VCL using its policy. Furthermore, under our policy, at least 95% of the users receive immediate service from the dedicated server pools, and 99.5% of them are guaranteed a service from the flexible pool with no more than 5 extra minutes of waiting.

For future work, we are interested in modeling the VCL system using Markovian decision processes, where decisions on the allocation of the servers and the admission of the users can be chosen to optimize the long run cost. In that case, we can allow switching between different types of dedicated servers, or between dedicated servers and flexible servers; we can also allow rejecting a user even when servers are available. Costs of switching servers and rejecting different types of users can also be incorporated. This is a rather involved project and will be undertaken at a later date.

Acknowledgement

This research is partially supported by Ministry of Scienceand Technology Major Project of China 2017YFC1310903, University of Hong Kong (HKU) Stanley Ho Alumni Challenge Fund, and HKU BRC Fund.

References

[1] Adan, I. J. B. F., Kulkarni, V. G., & Van Wijk, A.C.C. (2013). Optimal control of a server farm. *INFOR*, 51, 241-252.

[2] Aksin, O. Z., Armony, M., & Mehrotra, V. (2007). The modern call center: A multi-disciplinary perspective on operations management research. *Production, Operations Management*, 16, 665-688.

[3] Aksin, O. Z., & Karaesmen, F. (2007). Characterizing the performance of process flexibility structures. *Operations Research Letters*, 35, 477-484.

[4] Aktekin, S., & Soyer, R. (2012). Bayesian analysis of queues with impatient customers: Applications to call centers. *Naval Research Logistics,* 59, 441-456.

[5] Bassamboo A., & Zheevi, A. (2009). On a data-driven method for staffing large call centers. *Operations Research*, 57, 714-726.

[6] Bertsimas, D., & Thiele, A. (2014). Robust and data-driven optimization: Modern decision making under uncertainty. *INFORMS Tutorials in Operations Research*, 95-122.

[7] Bassamboo, A., Randhawa, R. S., & Van-Mieghem, J. A. (2012). A little flexibility is all you need: Asymptotic optimality of tailored chaining and pairing in queuing systems. *Operations Research*, 60, 1423-1435.

[8] Brown, L., Gans, N., Mandelbaum, A., Sakov, A., Shen, H., Zeltyn, S., & Zhao, L. (2005). Statistical analysis of a telephone call center: A queueing-science perspective. *Journal of the American Statistical Association*, 100, 36-50.

[9] Cameron, A. C., & Trivedi, P. K. (1998). Regression Analysis of Count Data. Cambridge University Press, Cambridge, MA.

[10] Chaudhry, M. L., & Templeton, J. G .C. (1983). A First Course in Bulk Queues. Wiley, New York.

[11] Chen, B. P., & Henderson, S. G. (2002). Two issues in setting call centre staffing levels. *Annals of Operations Research*, 108, 175-192.

[12] Cooper, R. B. (1972). Introduction to Queueing Theory. Macmillan, New York.

[13] Gandhi, A., Harchol-Balter, M., Das, R., & Lefurgy, C. (2009). Optimal power allocation in server farms, Proceedings of ACM SIGMETRICS Conference on Measurement and Modeling of Computer Systems. Seattle, WA, June 2009.

[14] Gandhi, A., Harchol-Balter, M., Raghunathan, R., & Kozuch, M. (2011). Distributed, robust autoscaling policies for power management in compute intensive server farms. Open Cirrus Summit, Georgia Tech, Atlanta, GA, October 2011 .

[15] Gandhi, A., Harchol-Balter, M., & Adan, I. J. B. F. (2010). Server farms with setup costs. *Performance Evaluation*, 67, 1123-1138.

[16] Gans, N., Shen, H., Zhou, Y.-Z., Korolev, N., McCord, A., & Ristock, H. (2015). Parametric forecasting and stochastic programming models for call-center workforce scheduling. *Manufacturing & Service Operations Management*, 17, 571-588.

[17] Green, L. V., Kolesar, P. J., & Svoronos, A. (1991). Some effects of nonstationarity on multiserver Markovian queuing systems. *Operation Research*, 39, 502-511.

[18] Green, L. V., Kolesar, P. J., & Soares, J. (2001). Improving the SIPP approach for staffing service systems that have cyclic demands. *Operations Research*, 49, 549-564.

[19] Green, L. V., Kolesar, P. J., & Whitt, W. (2007). Coping with time-varying demand when setting staffing requirements for a service system. *Production and Operations Management*, 16, 13-39.

[20] Harrison, J. M. (1988). Brownian models of queueing networks with heterogeneous customer populations. *Stochastic Differential Systems, Stochastic Control Theory, Applications*. Springer, New York. 147-186.

[21] Harrison, J. M. (2000). Brownian models of open processing networks: Canonical representation of workload. *Annual of Applied Probability*, 10, 75-103.

[22] Ibrahim, R., Ye, H., L'Ecuyer, P., & Shen, H. (2016). Modeling and forecasting call center arrivals: A literature survey and a case study. *International Journal of Forecasting*. 32, 865-874.

[23] Iravani, S. M. R., Kolfal, B., & Oyen, M. P. V. (2007). Call center labor cross-training: It's a small world after all. *Management Science*, 53, 1102-1112.

[24] Jongbloed, G., & Koole, G. M. (2001). Managing uncertainty in call centers using Poisson mixtures. *Applied Stochastic Models in Business and Industry*, 17, 307-318.

[25] Jordan, W. C., & Graves, S. C. (1995). Principles on the benefits of manufacturing process flexibility. *Management Science*, 41, 577-594.

[26] Khintchine, A. Y. (1960). Mathematical Methods in The Theory of Queueing. *Charles, Co., London*.

[27] Kibe, S., Koyama, T., & Uehara, M. (2012). The evaluations of desktop as a service in an educational cloud, *15th International Conference on Network-Based Information Systems, Melbourne, VIC,* 621-626.

[28] Kim, S.-H., & Whitt, W. (2014). Choosing arrival process models for service systems: tests of a nonhomogeneous Poisson process. *Naval Research Logistics*, 61, 66-90.

[29] Kuczura, A. (1973). The interrupted Poisson process as an overflow process. *The Bell System Technical Journal*, 437-448.

[30] Lee, N. (2013). Design and Control of Service Centers. (Doctoral dissertation). Retrieved from https://cdr.lib.unc.edu/.

[31] Liu, Y., & Whitt, W. (2011). A network of time-varying many-server fluid queues with customer abandonment. *Operations Research*, 59, 835-846.

[32] Liu, Y., & Whitt, W. (2012). The $G_t / GI / s_t + GI$ many-server fluid queue. *Queueing Systems*, 71, 405-444.

[33] Liu, Y., & Whitt, W. (2014). Many-server heavy-traffic limit for queues with time-varying parameters. *The Annals of Applied Probability*, 24, 378–421.

[34] Massey, W. A., & Whitt, W. (1994). An analysis of the modified offered-load approximation for the nonstationary Erlang loss model. *The Annals of Applied Probability*, 4, 1145-1160.

[35] Messerli, E. J., 1972. Proof of A Convexity Property of The Erlang-B Formula. *The Bell System Technical Journal*, 51, 951-953.

[36] Palm, C. (1988). Intensity Variations in Telephone Traffic. North-Holland, Amsterdam.

[37] Puhalskii, A. A., & Reed, J. E. (2010). On many-server queues in heavy traffic. *The Annals of Applied Probability*, 20, 129-195.

[38] Shen, H., & Huang, J. Z. (2008). Interday forecasting, intraday updating of call center arrivals. *Manufacturing, Service Operations Management*, 10, 391-410.

[39] Thompson, G. M. (1993). Accounting for the multi-period impact of service when determining employee requirements for labor scheduling. *Journal of Operations Management*, 11, 269-287.

[40] Weinberg. J., Brown, L. D., & Stroud, J. R. (2007). Bayesian forecasting of an inhomogeneous Poisson process with applications to call center data. *Journal of the American Statistical Association*, 102, 1185-1199.

[41] Whitt, W. (2007). What you should know about queueing models to set staffing requirements in service systems. *Naval Research Logistics*, 54, 476-484.

Appendix:

A1. Additional plots for sample applications

Arrival patterns for applications 10 and 100, in comparison to Figures 2 and 3 in the main paper.

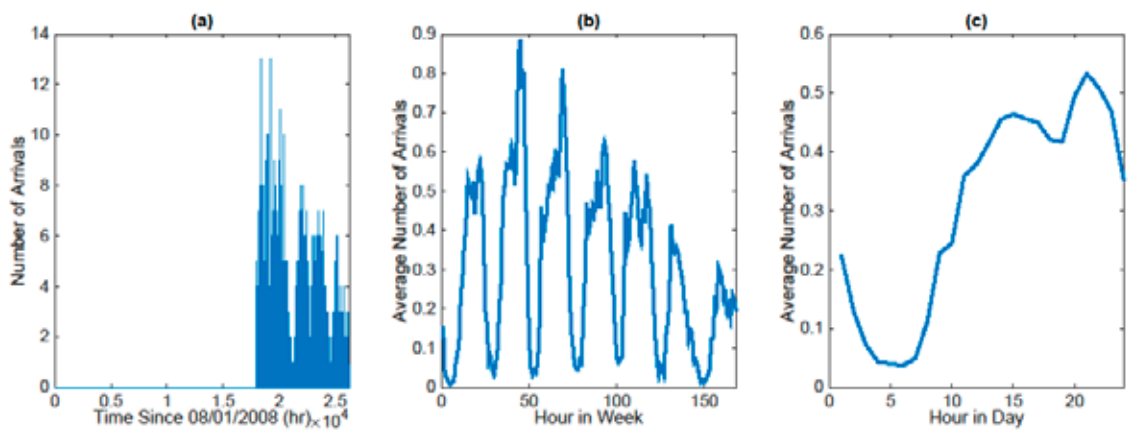

Figure 11. Arrivals of Application 10.

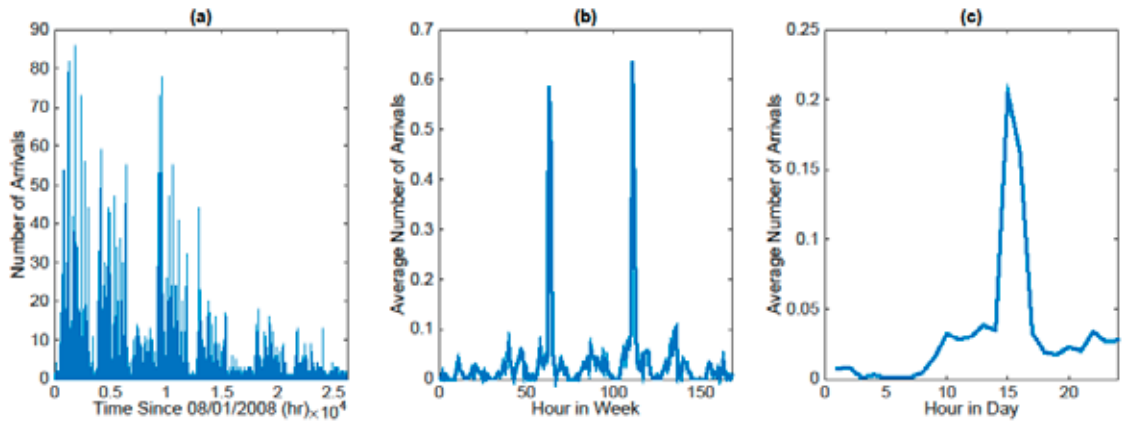

Figure 12. Arrivals of Application 100.

A2. NHPP tests on the arrival data

Using the results of Brown *et al.* [8] we construct the Kolmogorov-Smirnov test for the null hypothesis that arrivals of given types of application form a NHPP. Their approach addresses the time-varying arrival rate by converting the problem into a standard statistical test to determine whether inter-arrival times data can be regarded as a sample from a sequence of independent and identically distributed (i.i.d.) random variables with a specified distribution (see Kim and Whitt [28]). The first step is to approximate the NHPP by a piecewise-constant (PC) NHPP, by assuming that the arrival rates are constant on each interval. The second step is to apply the conditional-uniform (CU) transformation to transform the PC NHPP into a sequence of i.i.d. random variables uniformly distributed on $[0,1]$. Because of the assumption of PC, the NHPP now can be regarded as a homogeneous Poisson process (PP) over each interval. For a PP on $[0,T]$, conditioned on the total number of arrivals in that interval, the arrival times divided by T are distributed as the order statistics of i.i.d. random variables uniformly distributed on $[0,1]$. Finally, following Brown *et al.* [8] we use a scaled logarithmic transformation of the data, which under the Poisson null hypothesis produces a sequence of i.i.d. mean-one exponential random variables. Then we apply the KS test with $F(x)=1-e^{-x}$.

Using our data, the first test example includes all the arrivals for application 1 arriving on every Monday from 14:00 to 14:59, August 1, 2008 to July 31, 2011. In total we have 150 such Monday one-hour intervals, and 470 arrivals. The respective Kolmogorov-Smirnov statistic has a value of $K=0.0374$ (p-value = 0.3811). The second example includes 278 arrivals requesting for type 2 application on Wednesday Oct 14th, 2009. The interval length is half hour. For this case we have Kolmogorov-Smirnov statistic $K=0.0648$ (p-value = 0.18). These results are typical of those we have obtained from various selections of intervals of the various types of requests. Thus, overall there is no evidence in this data

set to reject the null hypothesis that the arrival process of application requests is NHPP with PC arrival rates. We also apply the root-unroot method from Brown *et al.* [8] to stabilize the variance.