Energy-Efficient Dynamic Server Provisioning in Server Farms[☆]

Burak Bartan, Nail Akar[*]

Electrical and Electronics Engineering Department
Bilkent University, Ankara, 06800 Turkey

Abstract

Energy management of a primary server farm for delay sensitive jobs is considered under non-zero setup times. A job request is said to overflow to the secondary server farm (to be served along with delay tolerant jobs) when no primary server is available at the time of request. We propose two algorithms (synchronous and asynchronous) for dynamic server provisioning for the primary server farm under overflow probability constraints. We also introduce queuing models for both algorithms when the arrival process is Poisson. Numerical examples are provided to validate the effectiveness of the proposed algorithms for actual traffic traces when the arrival process is a non-homogeneous Poisson process.

Keywords: server farms, energy efficiency, level-dependent QBD

1. Introduction

A server farm is a cluster of servers maintained by a single organization that provide computational resources to a very large number of users. A well-managed server farm ensures that its users receive an acceptable performance dictated by a Service Level Agreement (SLA) established between

[☆]This work is supported in part by the Science and Research Council of Turkey (Tübitak) under project grant EEEAG-115E360.

[*]Corresponding author

Email addresses: `burak.bartan@ug.bilkent.edu.tr` (Burak Bartan), `akar@ee.bilkent.edu.tr` (Nail Akar)

URL: `www.ee.bilkent.edu.tr/~akar` (Nail Akar)

the organization and its users. For web workloads and database applications, response time has been viewed as the most important performance metric; other performance metrics may also be defined for different applications [1]. Reducing the operational costs of server farms is key to profitability of such organizations. One effective means to reduce the operational expenses is to deploy systems and/or mechanisms to reduce the total power consumed by the servers [2]. One such mechanism is based on the concept of power-proportional servers that consume power in proportion with their utilization, which is enabled by mechanisms such as Dynamic Voltage and Frequency Scaling (DVFS) by which a server can be run at a lower voltage or frequency to consume less power in case of light load [3]. However, servers consume about 60-65% of their peak power when idle (see for example [4],[5]) which limits the effectiveness of DVFS. The current paper's focus will be on the alternative Dynamic Server Provisioning (DSP) mechanism which refers to the dynamic management of the number of *on* servers as a function of the workload [1], also called dynamic right-sizing [6]. In case of light loads, some of the servers can be turned off towards power reduction with DSP which may later be turned back on when the load picks up again. However, servers cannot be turned on instantaneously and they need a considerable setup time in the order of several minutes so as to start serving user requests [5]. Such setup times are known to limit the effectiveness of DSP systems. Moreover, servers consume a significant amount of power during setup while no useful service is being delivered [5].

DSP has attracted considerable research attention in the past few years. We now review the most related studies to that of the current paper in the context of multi-server queuing systems with the on servers offering service to the incoming jobs under the first-come-first-served (FCFS) discipline and a waiting room for jobs that cannot be served immediately. In [7], the servers can be turned on and off one at a time but only one server is allowed to be in the setup state, called an ON/OFF/STAG staggered queuing system in [5]. With these assumptions, [7] obtains numerically the steady-state distribution of the underlying CTMC (Continuous-Time Markov Chain) using matrix-analytic techniques. The authors consider in [8] a similar problem to that given in [7] in the context of inventory systems. The distribution of the response times for the ON/OFF/STAG queuing system is given in [5] which also proposes a strategy in which an idle server is turned off only if the total number of busy and idle servers exceeds a certain threshold. This strategy is evaluated in [5] by matrix-analytic methods. A queuing model is introduced in [9] where micro-managing individual servers is not allowed but rather a block of reserve servers is turned on altogether when the number of jobs in the system becomes sufficiently high. However, only one single block of reserves is allowed, limiting the effectiveness of the proposed approach. The work in [10] proposes a multi-server queuing model for activation/deactivation of servers with finite buffer capacity whose control is performed by a finite state machine (FSM) with various hysteresis operation modes. On-line right-sizing algorithms have been proposed and shown to work for a wide variety of workloads in [11],[12],[6] but without queuing-theoretic models.

It is important in DSP to exploit the heterogeneity of jobs in server farms which can be classified into delay sensitive and delay tolerant jobs [13]. In this paper, we envision a Primary Server Farm (PSF) infrastructure for delay sensitive job requests which can be considered high-priority. An incoming high-priority job is assigned to one of the idle primary servers in the PSF in case of availability; otherwise it is said to overflow from the PSF and directed towards the Secondary Server Farm (SSF). On the other hand, all delay tolerant (low-priority) job requests are assigned to the SSF servers in which jobs are concurrently served, e.g., processor sharing. A high-priority job accepted into the PSF encounters zero wait time. The focus of this paper is dynamic server provisioning for the PSF with the goal of power reduction under the constraint that the probability of overflow denoted by P_o experienced by high-priority requests when there are no PSF idle servers, is kept below a desired level ε. We do not delve into the performance study of the SSF in this paper in terms of response times. We assume a PSF with k servers, high-priority request arrival process modeled as a non-homogeneous Poisson process with rate $\lambda(t)$, exponentially distributed service times with parameter μ, and deterministic setup times T. We propose two algorithms: an asynchronous DSP, called ADSP, and a synchronous DSP, called SDSP, the latter deciding to activate and de-activate servers at integer multiples of the setup time T based on the transient solution of the $M/M/N/N$ system. On the other hand, ADSP decides to turn off (turn on) servers when the number of servers in setup or idle states is above (below) a certain upper (lower) threshold. For the case of $\lambda(t) = \lambda$, we provide a level-dependent QBD model for ADSP and present an offline method to obtain the optimum values of the upper and lower thresholds of ADSP leading to minimum overall consumed power under the constraint $P_o < \varepsilon$ for a given λ. For computational tractability, the QBD model approximates the deterministic setup time by an exponential distribution with the same mean T. For the general $\lambda(t)$ case, the ADSP algorithm estimates $\lambda(t)$ at integer multiples of the so-called estimation period T_e and subsequently employs the optimum thresholds calculated offline that are associated with the estimate $\hat{\lambda}(t)$ until the next estimation epoch.

The paper is organized as follows. The ADSP and SDSP algorithms are described in Section 2 with their queuing models. Numerical results are presented in Section 3. Finally, we conclude.

2. Dynamic Server Provisioning Algorithms

We consider a PSF with k servers offering service to incoming high-priority requests. A PSF server can be in any of the four states: busy, setup, idle or off, with the power consumptions of each state being P_{busy}, P_{setup}, P_{idle}, and P_{off}, respectively. A server is in the busy state when it is serving a job whereas it is in the off state when turned off. A server is idle if it is ready to serve a job. A server transitions from an off state to either busy or idle states through an additional setup state. While a server is in setup, it cannot serve any jobs. The off state may be associated with a sleep mode leading to non-zero P_{off}, but we will take $P_{off} = 0$ in all the numerical examples. It is also generally true that $P_{busy} = P_{setup} > P_{idle}$ [5]. An incoming job is assigned to one of the idle servers leading to a zero wait time. If there are no idle primary servers in the PSF, the job overflows from the PSF and is subsequently assigned to one of the SSF servers that are concurrently serving low-priority requests and additionally high-priority requests that had previously overflown from the PSF. In this paper, we are only interested in the performance of high-priority requests, and in particular their overflow probability P_o, and we are not interested in the response times encountered in the SSF. Therefore, we do not make any assumptions on the SSF infrastructure and the nature of low-priority traffic. The setup time is assumed to be deterministic [12] and is denoted by T. Next, we present the two algorithms we propose for dynamic server provisioning.

2.1. Asynchronous Dynamic Server Provisioning (ADSP)

In ADSP(U, L), the number of servers in the setup or idle states is kept between an upper and a lower threshold, denoted by U and L, respectively. Let us denote the number of servers in the setup state by s and the number of idle servers by l. If the sum $s + l$ exceeds U, one of the servers in the setup state is turned off. In particular, the server with the most remaining setup time will be selected. If, however, there is no server in the setup state, one of the idle servers is turned off. If the number $s+l$ drops below the lower limit L, one of the off servers is put into the setup mode, resulting in s increasing by one. In short, the proposed hysteretic policy aims at keeping the sum $s+l$ between U and L by turning servers on or off when the sum $s+l$ is not in the range. The threshold parameters U and L need to be chosen so that P_o is less than a desired overflow probability ε and the overall consumed power is minimized.

Next, we provide an approximate stochastic model for the hysteretic ADSP(U, L) policy for the case $\lambda(t) = \lambda$. For computational tractability, we approximate the deterministic setup time by an exponentially distributed random variable with parameter $\alpha = T^{-1}$, also called the setup rate. With similar reasoning, we assume that when one of the servers in the setup state is to be turned off, this selection is made randomly without paying attention to the remaining setup time. This is a reasonable assumption since in this case, the remaining service time may not even be known to the ADSP decision entity. Under these assumptions, the system is representable by a 3-dimensional Markov chain with the state space

$$\mathcal{S} = (i, s, l), 0 \leq i, s, l \leq k, i+s+l \leq k. \tag{1}$$

where i, s, l denote the number of servers in the busy, setup, and idle states, respectively. Obviously, $(k-i-s-l)$ gives the number of off servers. Let us assume the MC (Markov Chain) is in state (i, s, l). Related to new job requests, the MC will transition with rate λ to the following states:

i) to state $(i+1, s, l-1)$ if $l \geq 1, s+l-1 \geq L$. In this case, the job is assigned to one of the idle servers.

ii) to state $(i+1, s+1, l-1)$ if $s+l = L, l \geq 1, i+s+l < k$. In this case, the incoming job is first assigned to one of the idle servers. Therefore, the sum of the idle and setup states momentarily turns out to lag the lower threshold L by one. Subsequently, an off server is put into the setup mode.

iii) to state $(i+1, s, l-1)$ if $s+l \leq L, l \geq 1, i+s+l = k$. The situation is similar to above but there is no off server to put into the setup mode.

Note that when a new job request arrives when in state $(i, s, 0)$, the job overflows from the PSF to the SSF. Related to job service completions, the MC will transition from state (i, s, l) with rate $i\mu$ to the following states:

i) to state $(i-1, s, l+1)$ if $s+l+1 \le U$. In this case, one of the jobs gets to complete and the number of idle servers is increased by one.

ii) to state $(i-1, s-1, l+1)$ if $s \ge 1, s+l = U$. In this situation, the sum of the idle and setup states momentarily exceeds U and one of the servers in setup is turned off.

iii) to state $(i-1, s, l)$ if $s = 0, l = U$. A job completion gives rise to deactivating an idle server.

Finally, related to a setup completion, the MC will transition to $(i, s-1, l+1)$ with rate $s\alpha$ if $s \ge 1$. Based on this description, the MC can be shown to be a finite level-dependent QBD (LD-QBD) with block tri-diagonal generator Q in the form

$$Q = \begin{bmatrix} Q_{0,0} & Q_{0,1} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} \\ Q_{1,0} & Q_{1,1} & Q_{1,2} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & Q_{2,1} & Q_{2,2} & Q_{2,3} & & \mathbf{0} \\ \vdots & & \ddots & \ddots & \ddots & \vdots \\ \mathbf{0} & \cdots & & Q_{k-1,k-2} & Q_{k-1,k-1} & Q_{k-1,k} \\ \mathbf{0} & \cdots & & \mathbf{0} & Q_{k,k-1} & Q_{k,k} \end{bmatrix}, \tag{2}$$

where the number of busy servers is chosen as the level of the LD-QBD [14],[15]. Since the number of idle and setup servers cannot exceed U in ADSP, each block in the above generator is a square matrix of size b with

$$b = \frac{(U+1) \times (U+2)}{2}.$$

The steady-state probabilities of each state $(i, s, l) \in \mathcal{S}$, denoted by $\pi_{(i,s,l)}$, are obtained by solving for the left null vector of Q denoted by π satisfying

$$\pi Q = 0, \sum_{(i,s,l) \in S} \pi_{(i,s,l)} = 1 \tag{3}$$

using the probabilistic algorithm given in [14] or the block tri-diagonal LU decomposition algorithm given in [16] with computational complexity $\mathcal{O}(b^3 k)$. Overflow from the PSF occurs when there are no idle servers in which case we write the overflow probability as

$$P_o = \sum_{(i,s,0) \in S} \pi_{(i,s,l)}. \tag{4}$$

The average power consumed by the PSF (denoted by P) is then given by the following expression:

$$P = \sum_{(i,s,l)\in S} \pi_{(i,s,l)} \left(iP_{busy} + sP_{setup} + lP_{idle} + (k-i-s-l)P_{off}\right). \qquad (5)$$

For a given λ, we propose to perform off-line calculations for obtaining the optimum values for the upper and lower thresholds U and L, denoted by $U^*(\lambda)$ and $L^*(\lambda)$, respectively, with known system parameters k, α, and μ, and desired overflow probability ε. In particular, $U^*(\lambda)$ and $L^*(\lambda)$ are chosen so as to yield the lowest average power consumption P satisfying $P_o < \varepsilon$. The ADSP algorithm that uses these two optimum values for a given λ is then denoted by $\text{ADSP}_\lambda(U^*(\lambda), L^*(\lambda))$. Let us now focus on the more general case for ADSP when the job arrival process is characterized by a non-homogeneous Poisson process with rate $\lambda(t)$ that varies over time. Typically, this process is hard to predict in advance and requires on-line estimation. We propose to use the following estimator $\hat{\lambda}(t)$ in our numerical examples:

$$\hat{\lambda}(t) = \frac{\text{number of job arrivals in } [t-T_m, t)}{T_m}, \qquad (6)$$

where T_m refers to a measurement period of choice, selected long enough to produce non-noisy output. Other estimators are also possible but left outside the scope of the current research. Let $T_e < T_m$ be the estimation period. When $t = jT_e$ for some $j \in \mathcal{Z}$, the estimation procedure is triggered in ADSP and the estimate $\hat{\lambda} = \hat{\lambda}(jT_e)$ is produced. Subsequently, the algorithm

$$\text{A}DSP_{\hat{\lambda}}\left(U^*(\hat{\lambda}), L^*(\hat{\lambda})\right)$$

is run in the time interval $[jT_e, (j+1)T_e)$ until the next estimation epoch $(j+1)T_e$. This process repeats itself in ADSP. Note that ADSP algorithm described above can be run on any server farm with arbitrary setup time distributions but using the upper and lower thresholds of the algorithm calculated offline on the basis of the assumption regarding exponentially distributed setup times. This is the approach we take in the numerical examples involving deterministic setup times.

2.2. Synchronous Dynamic Server Provisioning

In SDSP, the decision to activate or de-activate servers is made at integer multiples of the deterministic setup time T. Let $\hat{\lambda} = \hat{\lambda}(jT)$ denote the estimate of the job arrival rate to be used in the decision to be made at $t = jT_e$ using the estimator (6). Let $P_o(t)$ denote the time-varying overflow probability at time t. Also let $P_o[a,b] = \frac{1}{b-a}\int_a^b P_o(\tau)\,d\tau$ denote the average overflow probability in the time interval $[a,b), a < b$. Let i_0 and l_0 denote the number of busy and idle servers at time jT. Obviously, the activated servers (if any) at time epoch $(j-1)T$ have become idle as of time jT and there are no servers in the setup state. The $k - i_0 - l_0$ servers are in the off state. The decision to be made is

i) whether some of the idle servers are to be put into the off state or not.

ii) whether some of the off servers to be activated so that they become idle at time $(j+1)T$.

In the first case i), we attempt to find the minimum number of on servers N satisfying $i_0 \le N \le i_0 + l_0$ that guarantees a desired overflow probability $P_o[jT,(j+1)T]$ throughout the time interval $[jT,(j+1)T)$ on the basis of the estimate $\hat{\lambda}$ of the arrival rate. In order to calculate the quantity $P_o[jT,(j+1)T]$ as a function of N, we propose the following procedure. Recall that the number of busy servers at time jT is i_0 with the overall number of on servers being N in the interval $[jT,(j+1)T)$. Let $x_n(t), n = 0,1,\ldots,N$ denote the probability that there are n busy servers at time $t, jT \le t < (j+1)T$ and let $x(t) = [x_0(t), x_1(t), \ldots, x_N(t)]$. We also define $z(t)$ such that

$$\frac{d}{dt}z(t) = x_N(t), jT \le t < (j+1)T.$$

It is not difficult to show that

$$\frac{d}{dt}\begin{bmatrix} x(t) & z(t) \end{bmatrix} = \begin{bmatrix} x(t) & z(t) \end{bmatrix} Q^{(N)}, jT \le t < (j+1)T, \tag{7}$$

where

$$Q^{(N)} = \begin{bmatrix} -\hat{\lambda} & \hat{\lambda} & & & & 0 \\ \mu & -(\hat{\lambda}+\mu) & \hat{\lambda} & & & 0 \\ & 2\mu & -(\hat{\lambda}+2\mu) & \hat{\lambda} & & 0 \\ & & \ddots & \ddots & \ddots & \vdots \\ & & (N-1)\mu & -(\hat{\lambda}+(N-1)\mu) & \hat{\lambda} & 0 \\ & & & N\mu & -N\mu & 1 \\ 0 & 0 & \cdots & 0 & 0 & 0 \end{bmatrix},$$

and the vector v defined by

$$v = \begin{bmatrix} x(jT) & z(jT) \end{bmatrix}$$

is a $1 \times (N+2)$ row vector of zeros except for a unity entry in the $(i_0+1)^{th}$ position. Also let e be a $(N+2) \times 1$ vector of zeros except for its last entry which is unity. Note that

$$z((j+1)T) = \int_{jT}^{(j+1)T} x_N(\tau) d\tau = T P_o[jT, (j+1)T].$$

The solution for the differential equation (7) yields the following expression for the desired quantity $P_o[jT, (j+1)T]$:

$$P_o[jT, (j+1)T] = \frac{1}{T} z((j+1)T) = \frac{1}{T} v \, \mathbf{exp}(Q^{(N)}T) \, e, \tag{8}$$

where the operator $\mathbf{exp}(\cdot)$ denotes the matrix exponential. Let us now assume the existence of the minimum possible integer N such that $i_0 \leq N \leq i_0 + l_0$ satisfying $P_o[jT, (j+1)T] < \varepsilon$. In this case, $i_0 + l_0 - N$ out of the l_0 idle servers will be turned off at time jT.

In case such an integer N with $i_0 \leq N \leq i_0 + l_0$ satisfying $P_o[jT, (j+1)T] < \varepsilon$ does not exist, then the second situation ii) arises which happens explicitly when $P_o[jT, (j+1)T] \geq \varepsilon$ even for the choice of $N = i_0 + l_0$. This situation leads us to turn on some of the off servers at time $t = jT$ for meeting the overflow probability requirement. However, turning a server on will not have an immediate impact in the time interval $[jT, (j+1)T)$ since all the activated servers would be in the setup state in this interval. Therefore, we propose to find the minimum integer $N, i_0 + l_0 < N \leq k$ such that $P_o[(j+1)T, (j+2)T] < \varepsilon$. We will use the estimate $\hat{\lambda} = \hat{\lambda}(jT)$ to be used in the time interval $[(j+1)T, (j+2)T)$. In this case, we start with i_0 busy and l_0 idle servers in the time interval $[jT, (j+1)T)$, $N - i_0 - l_0$ servers that were put into the setup state at $t = jT$ become idle at $t = (j+1)T$, and the system would continue to evolve with N servers in the time interval $[(j+1)T, (j+2)T)$. According to this description, it is not difficult to show that

$$P_o[(j+1)T, (j+2)T] = \frac{1}{T} u \, \mathbf{exp}(Q^{(N)}T) \, e, \tag{9}$$

where e is defined as before, and the row vector u is defined as

$$u = \begin{bmatrix} u_0 \, \mathbf{exp}(Q^{(i_0+l_0)}T) & \mathbf{0}_{1 \times N - i_0 - l_0} \end{bmatrix},$$

and u_0 is row vector of zeros of size i_0+l_0+1 except for a one in the $(i_0+1)^{th}$ position. In SDSP, when a server activation decision is to be made at $t = jT$, at time $t = (j+1)T$, the SDSP process will be run again as described above, without paying attention to what was decided at the previous decision epoch.

3. Numerical Examples

We first define several key parameters that will be used in the numerical examples. $E = \lambda_m/\mu$ denotes the maximum total offered traffic in Erlangs where

$$\lambda_m = \max_t \lambda(t), \tag{10}$$

assuming a-priori known $\max_t \lambda(t)$. We define the system load as

$$\rho = \frac{\lambda_m}{\mu k}.$$

We assume that a server in busy or setup states consumes 200 Watts [1] and a server in the idle state consumes 60% of its peak value [5], i.e., $P_{idle} = 120$ Watts. We compare the proposed algorithms ADSP and SDSP with respect to the *always on* benchmark policy which activates $N \leq k$ servers all the time where N is the minimum integer satisfying $B(E,N) < \varepsilon$ where the Erlang-B blocking probability $B(\cdot,\cdot)$ is defined as

$$B(E,N) = \frac{\frac{E^N}{N!}}{\sum_{i=0}^{N} \frac{E^i}{i!}} \tag{11}$$

The percentage power gain of a proposed algorithm with respect to the benchmark algorithm is denoted by G:

$$G = 100 \times \frac{P(\text{benchmark}) - P(\text{algorithm})}{P(\text{benchmark})}. \tag{12}$$

A key parameter that will be used in the sequel is the ratio r of the mean service time to the setup time:

$$r = \frac{1}{T\mu} = \frac{\alpha}{\mu}. \tag{13}$$

All the numerical results have been obtained using MATLAB R-2015B on a computer with 4 GB of RAM and Intel i5 processor running on 2.67 GHz.

Table 1: Comparison of Algorithms 1 and 2

		Algorithm 1 [16]		**Algorithm 2 [14]**	
k	U	exec. time (sec.)	error	exec. time (sec.)	error
25	5	0.07	7.58e-16	0.10	5.68e-16
	10	0.50	2.42e-16	0.72	2.10e-16
	15	1.40	4.51e-16	2.10	2.76e-16
50	5	0.15	6.10e-16	0.21	4.14e-16
	10	1.05	3.19e-16	1.55	4.02e-16
	15	3.76	7.06e-16	5.64	8.18e-16
75	5	0.25	21.61e-16	0.34	26.31e-16
	10	1.68	25.53e-16	2.45	24.74e-16
	15	6.19	20.78e-16	9.30	19.59e-16

3.1. Example I

The linear system in (3) where Q is given as in (2) needs to be solved using computationally efficient algorithms because of the relatively large matrix sizes arising with ADSP models. We have employed two such algorithms: Algorithm 1 uses block tridiagonal LU factorization and the details of the algorithm are provided in [16] and Algorithm 2 is a probabilistic algorithm used in the solutions of tri-diagonal linear systems arising in finite birth-and-death models [14]. Algorithms 1 and 2 are compared with respect to their execution times and numerical accuracy in Table 1 for a sample scenario when $\mu = 1$, $\alpha = 0.1$, $\rho = 0.3$, and $L = 2$ as a function of the parameters k and U. The error column gives the quantity $||\pi Q||_2$ which is indicative of the numerical accuracy of the corresponding algorithm. Table 1 illustrates that Algorithm 1 is slightly faster than algorithm 2 with similar numerical accuracies. For the remaining numerical examples, we use Algorithm 1 for solving the ADSP model in (3).

3.2. Example II

In this example, we show how one can choose the upper and lower thresholds U and L for the ADSP algorithm for overall power consumption minimization. Fig. 1 illustrates how the overall power consumption P behaves as a function of the varying thresholds U and L in a PSF with $k = 50$ servers where $\mu = 1$, $\alpha = 0.1$, and $\rho = 0.3$ when ADSP is run. The solid bars are indicative of the situations in which the overflow probability P_o is less than

$\varepsilon = 0.001$ and the grey bars indicate otherwise. Fig. 1 clearly shows that if the thresholds U and L are chosen to be relatively high, the overall power P increases because in that case, obviously, we tend to keep more servers on compared to the case where we choose U and L lower. For a given value of λ, the thresholds $U^*(\lambda)$ and $L^*(\lambda)$ are then selected from those corresponding to the solid bars in Fig. 1 so as to yield the lowest power consumption. Fig. 2

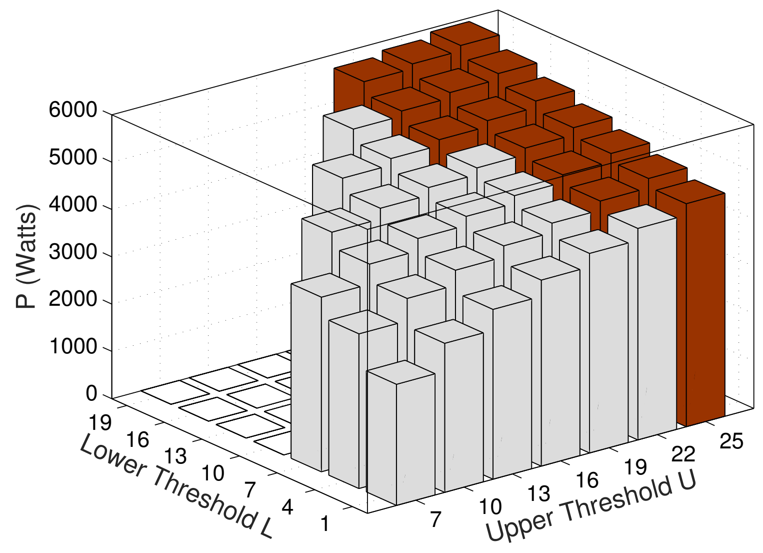

Figure 1: The overall consumed power P as a function of U and L for the PSF scenario with $k = 50$, $\mu = 1$, $\alpha = 0.1$, and $\rho = 0.3$.

shows how the optimal threshold values U^* and L^* behave for different setup rates and system loads assuming a system with $k = 50$ servers and $\mu = 1$. Fig. 2 illustrates that the upper threshold U^* tends to increase with increasing setup rates and higher loads. The lower threshold L^* usually takes values close or equal to 1 for this particular example.

3.3. Example III

In this example, we study the power gain G attained by ADSP when $\lambda(t) = \lambda$. In Fig. 3, the power gain G of the ADSP policy is depicted as a function of the parameters r and ρ for three different values of $k = 25$, 50, and 75, and for two different values of the tolerance parameter $\varepsilon = 0.001$, and 0.01. Fig. 3 reveals that the gain G increases with increased setup rate,

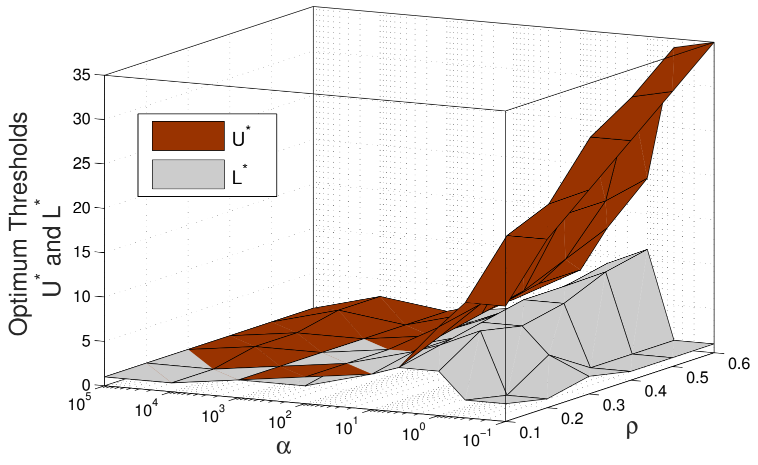

Figure 2: The optimum thresholds U^* and L^* as a function of the setup rate α and load ρ for a PSF with $k = 50$ servers and $\mu = 1$.

reduced loads, and relaxed overflow probability requirements. For low setup rates, the power gain G diminishes to zero, and even falls below zero for certain scenarios. This numerical example illustrates for the $\lambda(t) = \lambda$ case that ADSP can only be useful when the setup rate α is relatively higher. The potential benefits of ADSP for the general time-varying $\lambda(t)$ case are to be explored in the final two examples.

3.4. Example IV

In the remaining examples, both ADSP and SDSP policies are compared using simulations for which the parameter μ is set to 1 jobs/sec., the setup time T is assumed to be deterministic, and the tolerance parameter $\varepsilon = 0.001$. In this case, ADSP uses the values of the upper and lower thresholds from the off-line analytical tool based on the assumption of exponentially distributed setup times. For the current example, we set the capacity of the PSF to $k = 75$ and $\lambda(t) = \lambda$. The attained power gain G is depicted in Fig. 4 as a function of the parameter r. A notable power gain is observed only for relatively large values of the parameter r, i.e., $r > 1$, for which the setup time T is shorter than the average job service time for both policies, and also for relatively light loads. However, SDSP performs better than ADSP in the case of longer setup times and produces a positive gain for all values of r. We also note that, the always on benchmark policy for this example employs

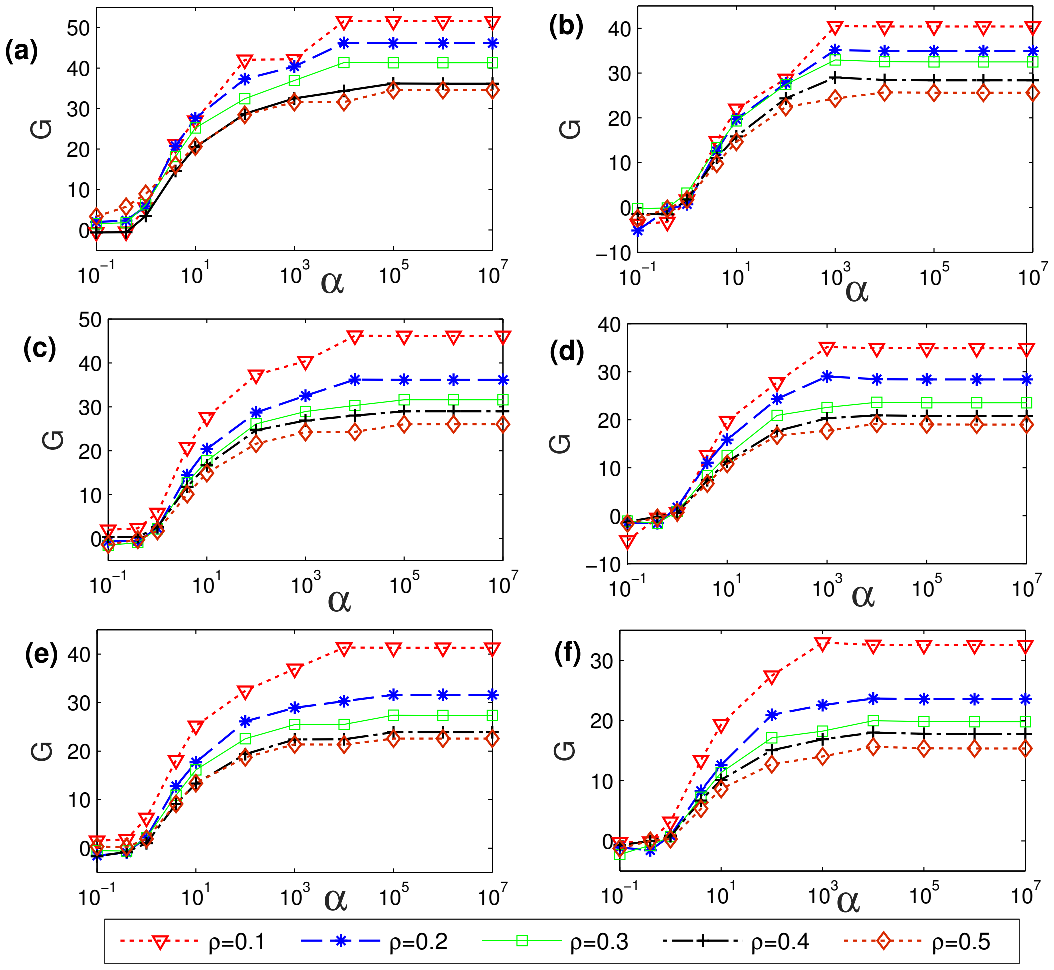

Figure 3: The power gain G attained as a function of the parameters r and ρ for six different scenarios: (a) $k = 25$, $\varepsilon = 0.001$, (b) $k = 25$, $\varepsilon = 0.01$, (c) $k = 50$, $\varepsilon = 0.001$, (d) $k = 50$, $\varepsilon = 0.01$, (e) $k = 75$, $\varepsilon = 0.001$, (f) $k = 75$, $\varepsilon = 0.01$

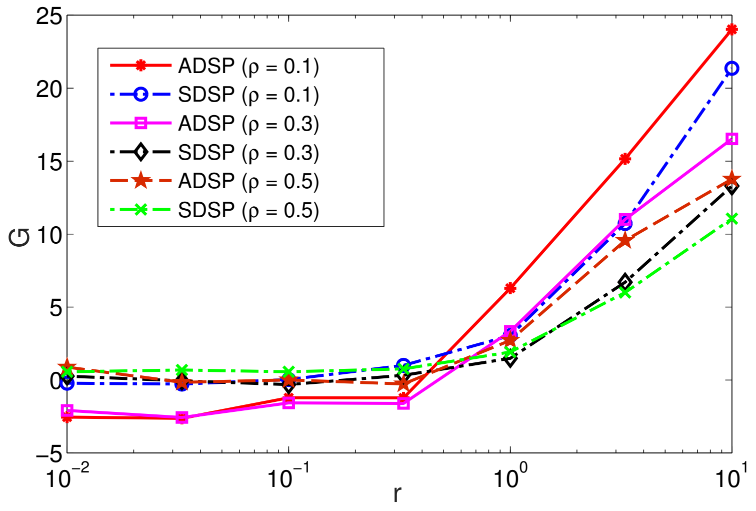

Figure 4: Power gain G as a function of the parameter r for the ADSP and SDSP policies for three different values of ρ for $\lambda(t) = \lambda$ case.

18, 38, and 57 servers, leading to overflow probabilities of 0.71e-3, 0.77e-3, and 0.68e-3, when the system load equals 0.1, 0.3, and 0.5, respectively.

3.5. Example V

The next two examples are devoted to the more general case of a non-homogeneous Poisson job arrival process with intensity $\lambda(t)$. We have used amplitude-scaled versions of the two different traces taken from [17] and also given in Fig. 5 for the time-varying job intensity in these two simulations. This scaling is done in this example so that ρ equals 0.63 and 0.61, for traces 1 and 2, respectively. We fix $k = 75$, $T_e = 10$ sec. and $T_m = 200$ sec. in the current example. Simulation results for the power gain G and P_o are provided for both traces as a function of the parameter r in Fig. 6 and Table 3. Recall that when $t = jT_e$ for some $j \in \mathcal{Z}$, the estimation procedure is triggered in ADSP and the estimate $\hat{\lambda} = \hat{\lambda}(jT_e)$ is produced and the algorithm $ADSP_{\hat{\lambda}}\left(U^*(\hat{\lambda}), L^*(\hat{\lambda})\right)$ is run in the time interval $[jT_e, (j+1)T_e)$ until the next estimation epoch $(j+1)T_e$. In the actual implementation of ADSP, the pair $\left(U^*(\hat{\lambda}), L^*(\hat{\lambda})\right)$ is only calculated for a finite set of λ values corresponding to uniformly sampled values of ρ. In ADSP^1 (ADSP^2), the average load samples are 0.1 (0.025) units apart, and the thresholds are used

Table 2: Overflow probability P_o obtained for the ADSP and SDSP policies for the $\lambda(t) = \lambda$ case for various values of r and ρ.

	$\rho = 0.1$		$\rho = 0.3$		$\rho = 0.5$	
r	ADSP	SDSP	ADSP	SDSP	ADSP	SDSP
0.01	0.68e-3	0.81e-3	1.06e-3	9.57e-3	5.23e-3	13.51e-3
0.03	0.64e-3	0.71e-3	0.65e-3	3.28e-3	2.45e-3	4.70e-3
0.1	0.71e-3	0.92e-3	0.67e-3	1.38e-3	1.61e-3	2.14e-3
0.3	0.71e-3	1.35e-3	0.61e-3	1.36e-3	1.06e-3	1.66e-3
1	1.85e-3	2.00e-3	2.10e-3	1.96e-3	2.16e-3	2.64e-3
3.3	1.85e-3	2.92e-3	2.68e-3	3.51e-3	4.32e-3	3.74e-3
10	2.35e-3	3.44e-3	2.39e-3	3.38e-3	2.63e-3	4.24e-3

corresponding to a sampling value in the sampling set which is closest to the estimate $\hat{\lambda}/(\mu k)$. We also study another variation of the ADSP policy, called ADSP^3, for which fixed upper and lower thresholds are employed throughout the simulation which are optimal as far as the actual job arrival rate equals the average job arrival rate$\bar{\lambda}$.

Table 3: Overflow probability P_o obtained with various server provisioning policies for various values of the parameter r for both traffic traces.

	Trace 1				Trace 2			
r	ADSP^1	ADSP^2	ADSP^3	SDSP	ADSP^1	ADSP^2	ADSP^3	SDSP
0.01	8.45e-3	5.45e-3	4.74e-3	1.59e-3	1.79e-3	1.47e-3	0.86e-3	1.30e-3
0.03	1.53e-3	1.15e-3	0.61e-3	1.11e-3	1.02e-3	0.80e-3	0.34e-3	1.06e-3
0.1	1.26e-3	1.00e-3	0.45e-3	1.12e-3	0.92e-3	0.69e-3	0.23e-3	1.05e-3
0.3	1.22e-3	1.04e-3	0.35e-3	1.43e-3	0.97e-3	0.81e-3	0.20e-3	1.40e-3
1	2.39e-3	2.10e-3	0.58e-3	2.49e-3	2.32e-3	2.13e-3	0.44e-3	2.31e-3
3.3	2.98e-3	2.34e-3	0.79e-3	3.60e-3	3.31e-3	2.49e-3	0.59e-3	3.52e-3
10	1.45e-3	1.23e-3	0.44e-3	3.51e-3	1.56e-3	1.35e-3	0.36e-3	3.55e-3

We observe that dynamic adjustment of the lower and upper thresholds on the basis of the estimation of the job arrival rate helps significantly in saving power since ADSP^1 and ADSP^2 consistently outperform ADSP^3 for all the studied values of the parameter r. The improvement obtained with finer adjustment with ADSP^2 compared to ADSP^1 is marginal which is indicative of a requirement for off-line calculations for these two thresholds for

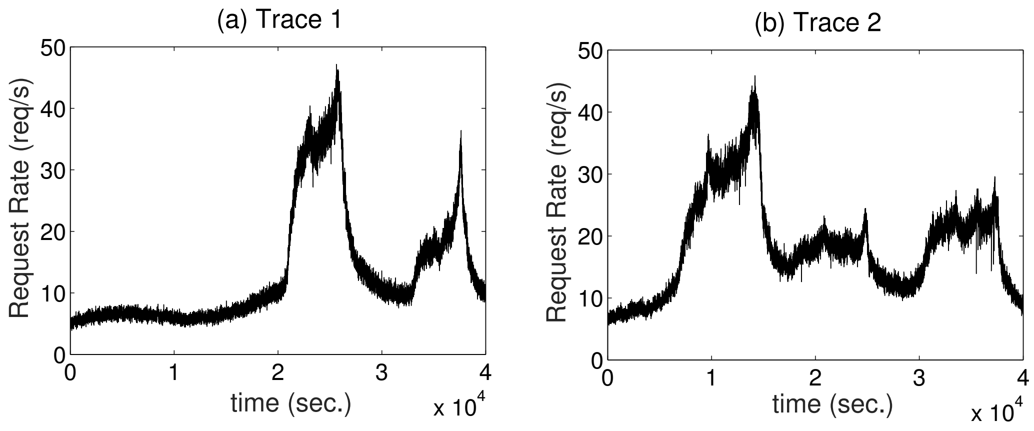

Figure 5: Traces used for the simulation of the general time-varying $\lambda(t)$ case (a) Trace 1 (b) Trace 2.

a relatively small set of λ values. Fig. 6 illustrates that the algorithms ADSP and SDSP perform similarly with respect to each other for the time-varying job arrival rate case as in the fixed intensity case. However, the attained power gains are much higher than those we had observed for fixed arrival rates. This observation clearly shows that the proposed algorithms adapt well to changes in $\lambda(t)$. Another key observation from Fig. 6 and Table 3 is that SDSP outperforms the ADSP policy in terms of power gain while not sacrificing much from overflow probabilities in the case of relatively longer setup times which is the case in actual systems. For reference, the always on benchmark policy keeps 68 and 67 servers on, leading to overflow probabilities of 4.36e-5 and 2.60e-5, respectively, for traces 1 and 2.

3.6. Example VI

For computational reasons, we have not been able to assess the performance of the ADSP policy via analysis for much larger PSF sizes than the ones studies in the previous examples. In the current example, we investigate the behavior of the SDSP policy only, for larger PSF sizes. Table 4 presents both the power gain G and overflow probability P_o for system loads of $\rho = 0.3$ and 0.5 for various PSF sizes with $k = 250$, 500, and 1000 for traces 1 and 2. In these simulations, we assume $T = 10$ seconds and $\mu = 1$ jobs/sec. Table 4 reveals that the power gain is higher with SDSP for a PSF with increased number of servers. However, with the power gain increasing, we also observe that the corresponding overflow probabilities increase. The key result here,

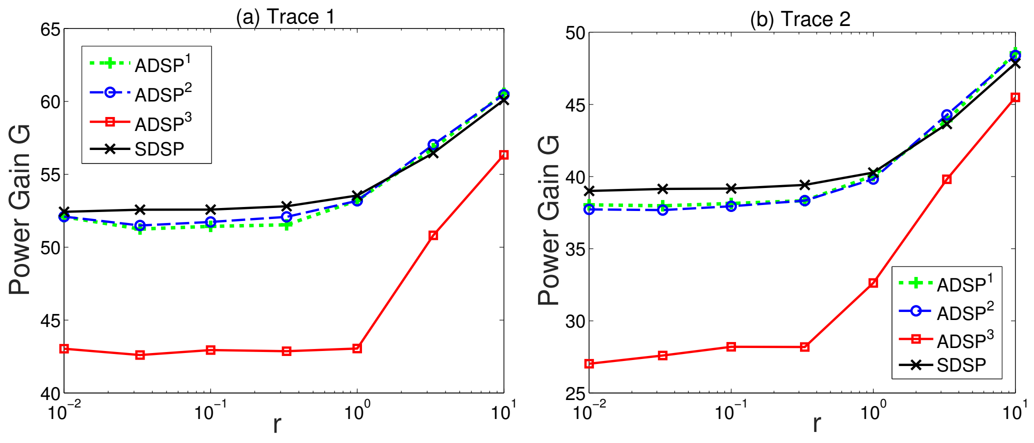

Figure 6: Power gain G as a function of the parameter r for various server provisioning policies (a) Trace 1 (b) Trace 2.

Table 4: Power gain G and overflow probability P_o with the SDSP policy for a PSF for various values of k and for two values of ρ.

		Trace 1		Trace 2	
k	ρ	P_o	G	P_o	G
250	0.3	1.42e-3	53.86	1.27e-3	40.34
	0.5	1.34e-3	54.86	1.89e-3	41.06
500	0.3	2.09e-3	55.02	1.74e-3	41.21
	0.5	3.35e-3	55.73	2.34e-3	41.76
1000	0.3	3.10e-3	55.89	2.83e-3	41.88
	0.5	4.74e-3	56.42	3.59e-3	42.32

though, is that for larger PSF as well, the SDSP policy is shown to attain high power gains and ensure overflow probabilities close to the desired value ε.

4. Conclusions

We propose two policies, namely the ADSP and SDSP policies, to dynamically provision the number of on servers in a primary server farm with deterministic server setup times to save power while maintaining a desired overflow probability. We show that power gains are minor for both policies in the case of fixed job arrival rates unless server setup times are short compared to mean job service times. However, we have shown through two traffic traces available in the literature that the attained power gains are significant for time-varying job arrival rates. Moreover, SDSP is devised with deterministic setup times in mind, and leads to higher power gains for relatively longer setup times when compared to ADSP.

5. References

[1] A. Gandhi, Dynamic server provisioning for data center power management, Ph.D. thesis, Computer Science Department, Carnegie Mellon University (2013).

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

[3] G. Semeraro, G. Magklis, R. Balasubramonian, D. Albonesi, S. Dwarkadas, M. Scott, Energy-efficient processor design using multiple clock domains with dynamic voltage and frequency scaling, in: High-Performance Computer Architecture, 2002. Proceedings. Eighth International Symposium on, 2002, pp. 29–40.

[4] A. Greenberg, J. Hamilton, D. A. Maltz, P. Patel, The cost of a cloud: Research problems in data center networks, SIGCOMM Comput. Commun. Rev. 39 (1) (2008) 68–73.

[5] A. Gandhi, M. Harchol-Balter, I. Adan, Server farms with setup costs, Performance Evaluation 67 (11) (2010) 1123 – 1138, performance 2010.

[6] M. Lin, A. Wierman, L. L. H. Andrew, E. Thereska, Dynamic right-sizing for power-proportional data centers, IEEE/ACM Trans. Netw. 21 (5) (2013) 1378–1391.

[7] J. R. Artalejo, A. Economou, M. J. Lopez-Herrero, Analysis of a multi-server queue with setup times, Queueing Systems 51 (1) (2005) 53–76.

[8] I. J. Adan, J. van der Wal, Combining make to order and make to stock, Operations-Research-Spektrum 20 (2) (1998) 73–81.

[9] I. Mitrani, Managing performance and power consumption in a server farm, Annals of Operations Research 202 (2013) 121–134.

[10] P. J. Kuehn, M. E. Mashaly, Performance of self-adapting power-saving algorithms for ICT systems, in: Proc. IFIP/IEEE International Symposium on Integrated Network, 2013, pp. 720–723.

[11] S. Nedevschi, L. Popa, G. Iannaccone, S. Ratnasamy, D. Wetherall, Reducing network energy consumption via sleeping and rate-adaptation, in: Proceedings of the 5th USENIX Symposium on Networked Systems Design and Implementation, NSDI'08, USENIX Association, Berkeley, CA, USA, 2008, pp. 323–336.

[12] G. Chen, W. He, J. Liu, S. Nath, L. Rigas, L. Xiao, 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, NSDI'08, USENIX Association, Berkeley, CA, USA, 2008, pp. 337–350.

[13] D. Xu, X. Liu, A. Vasilakos, Traffic-aware resource provisioning for distributed clouds, Cloud Computing, IEEE 2 (1) (2015) 30–39.

[14] D. P. Gaver, P. A. Jacobs, G. Latouche, Finite birth-and-death models in randomly changing environments, Adv. Appl. Prob. 16 (4) (1984) 715–731.

[15] L. Bright, P. Taylor, Calculating the equilibrium distribution in level dependent quasi-birth-and-death processes, Communications in Statistics. Stochastic Models 11 (3) (1995) 497–525.

[16] G. H. Golub, C. F. van Loan, Matrix Computations, The Johns Hopkins University Press, 1996.

[17] M. Arlitt, T. Jin, 1998 world cup web site access logs (1998).
URL `http://ita.ee.lbl.gov/html/contrib/WorldCup.html`.