# Provisioning Servers in the Application Tier for E-commerce Systems

Daniel Villela*, Prashant Pradhan† and Dan Rubenstein*
*Dept. of Electrical Engineering
Columbia University
500 W. 120th St., New York, NY 10027
{dvillela,danr}@ee.columbia.edu
†Network Software and Services
IBM T. J. Watson Research Center
Hawthorne, NY 10532
ppradhan@us.ibm.com

***Abstract*— Server providers that support e-commerce applications as a service to multiple e–commerce websites traditionally use a tiered server architecture. This architecture includes an application tier to process requests that require dynamically generated content. How this tier is provisioned can significantly impact a provider's profit margin. In this paper, we study methods to provision servers in the application serving tier to increase a server provider's profits. First, we examine actual traces of request arrivals to the application tier of e-commerce sites, and show that the arrival process is effectively Poisson. Next, we construct an optimization problem in the context of a set of application servers modeled as $M/G/1/PS$ queueing systems, and derive three simple methods to approximate the allocation that maximizes profits. Simulation results demonstrate that our approximation methods achieve profits that are close to optimal and are significantly higher than those achieved via simple heuristics.**

## I. Introduction

Companies that offer e-commerce applications often contract a third-party web service provider to provide and manage the necessary computing infrastructure. To be profitable, these providers simultaneously service multiple customer companies, maintaining a separate service level agreement (SLA) with each customer. The stochastic nature of request arrivals and service times makes it impossible for the provider to meet the conditions of the SLA for every request it hosts. Hence, as part of the SLA agreement, the provider is charged for each *service miss*: a request whose service does not meet the requirements specified in the SLA. A financially sound strategy for the provider is to provision its resources among its set of customers in such a way that its profits are maximized, which translates to minimizing the charges accrued as a result of server misses.

The web servicing architecture used by providers typically consists of three tiers, each of which is provisioned independently. The *front-end serving tier* handles all simple, static web transactions, composed of HTTP (HTTPS) requests. The *application tier* handles more complex, dynamic queries that might involve the execution of java servlets or scripts. The *database tier* handles requests that involve the lookup of specific, non-cached data, such as ones' personal banking records.

The amount of work that must be performed by the servers that support the front-end tier is low enough that overprovisioning is a cheap solution to meet SLA requirements. It has been shown in McWerther *et al.* [13] that prioritized scheduling can improve the database tier's ability to meet its imposed SLA requirements. With methods to maximize profits in these two tiers, a provider's profits then depend on how well it provisions its application tier resources to service requests to that tier. If not configured properly, a customer can suffer numerous service misses which the provider winds up paying for. Surprisingly, there has been little work that assists providers in how to configure their application tier.

In this paper, we explore how a provider should allocate its application tier serving resources among its set of customers with whom it has established SLAs. A desirable allocation is one that maximizes profits by minimizing the cost accrued as a result of service misses. Along this front, we make three major contributions. First, we identify an appropriate model for the servicing system of the application tier. Second, we formalize the problem of allocating serving resources to multiple customers to maximize provider profits as an optimization problem. Third, we derive three efficient approximation methods that allocate resources to customers, and show through simulation that the allocations produced achieve profits that are close to optimal and are significantly higher than the profits achieved via simple heuristics.

To identify an appropriate model for the application tier servicing system, we analyze traces of requests for dynamic content to a department store e-commerce website from 2001. Using these traces, we characterize the arrival process of requests to the application tier. We find that, for the horizon time of interest, the arrival process is adequately described by a Poisson process.

Our formulation of the optimization problem models each of the provider's servers as an independent M/G/1/PS queue. The solution minimizes an objective function that describes the cost resulting from service misses. We compute allocations by deriving three approximation algorithms, each of whose computational complexity is linear in the number of customers that the service provider hosts. The first approximation assumes that the average servicing time of jobs for each customer is known. The second approximation requires both the first and second moments of the servicing time distribution. The third approximation method utilizes known bounds for the class

This material was supported in part by the National Science Foundation under Grant No. ANI-0117738 and CAREER Award No. ANI-0133829, and by a gift from Lucent Technologies. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation. Daniel Villela was partially supported by a CNPq-Brazil scholarship (Ref. Number 200168/98-3). Daniel Villela performed part of this work during his internship at IBM T. J. Watson Research Center.

of Exponential Bounded Burstiness processes, to which the Poisson process belongs. In special circumstances, a provider is able to estimate the adequate size of its total serving resources via a simple equation.

Results from experiments using event-driven simulation show that our approximation methods come close to minimizing service miss costs. We also compare the costs computed by the approximation methods to costs computed by naive heuristics. The results show that, by using our approximation methods, service providers can increase their profits by a significant margin.

This paper is organized as follows. In Section II we overview related work. Section III describes our model of the service provider and poses the optimization problem. In Section IV we demonstrate that, for timescales of interest, the arrival process of requests to the application tier is effectively Poisson. Section V describes our approximation algorithms, and Section VI shows results from simulations that demonstrate the performance of our algorithms. We present our concluding remarks in Section VII.

## II. Related Work

Previous work that investigates e–commerce workloads differs from our characterization study here in that previous work do not address the application–tier workload. Instead, studies in [14], [15] focus on the workload at the front-end tier of a website. There has been little progress in characterizing the e–commerce website workloads imposed specifically by jobs whose content is generated dynamically. Studies that investigate dynamic content workloads analyze sites other than e–commerce websites such as a sporting event website [3], [19] or analyze measurements from an instrumented client, rather than a server workload [18].

There have been numerous performance studies focusing on the problem of allocating a fixed set of resources in order to balance the load across the set of resources. Game-theoretic models for resource allocation have been proposed (refer to [10] and therein for additional references) under the assumption that the resources to be pooled are controlled by providers that do not cooperate. In contrast, our work focuses on a single provider that controls the allocation of its serving resources. Federgruen and Groenvelt [6] take an algorithmic approach to find conditions for the optimization formulation of resource allocation problem where resources are given in discrete units. This approach was subsequently generalized by Wolf and Yu [24]. Tantawi and Towsley [22] explored a similar problem via a graph-theoretic formulation, solving a resource-allocation optimization problem. These authors further extended this work in [23] to the case where there are multiple classes of resources. Our work is different in that our goal is not to balance load, but is to maximize profits. In particular, when customers' charges differ for service misses, a simple load balancing strategy is less favorable than a strategy which diverts resources to meet the load of higher-cost service misses at the expense of missing lower-cost service misses. In [17] Sairamesh *et al.* explore the problem of allocating network link capacity to different traffic classes in order to minimize costs. Their formulation, however, does not map easily to the provisioning problem at the application serving tier. They utilize an $M/M/1/B$ queueing model with a finite buffer of size $B$ for which the blocking probability is used as a utility function of the link capacity. Such a model does not accurately capture e–commerce systems. Furthermore, service level agreements are usually specified as function of response time and not as a function of the blocking probability.

A small body of work uses analytic models to evaluate e–commerce serving systems. Almeida *et al.* [1] propose a scheme of priority levels as function of potential revenue estimated for different types of requests to a single website, rather than customer differentiation for multiple websites as in our work. The works by Liu *et al.* [11], [12] and Farias *et al.* [4] are closest to our approach; both use a general concept of a cost model given by a product between customer request rate and fraction of requests that violate service level agreements for the front-end serving tier. There are several aspects of application-tier serving systems that are captured by our work that are not captured in these previous studies. For instance, each provider's server is dedicated to a particular customer, and may not be shared among multiple customers (as is assumed in previous work). Our methods permit us to find allocations via a pair of equations or, in special circumstances, a single–equation solution, both of which are simpler than the methods from [4], [11], [12]. This is of great importance in practical settings where a provider can adjust its serving infrastructure when loads fluctuate over time.

## III. Model for the Application Serving System

Our model consists of two classes of participants: a single e-business provider and $m$ customers which we number 1 through $m$. Each customer $i$ individually establishes a service level agreement (SLA) with the provider, the details of which are described below. The provider has at its discretion $n$ servers that it uses to service the requests of the $m$ customers. As is typically done in practice, the provider selects the number of servers that it will dedicate to each customer such that no two customers' requests are serviced by the same server. Each customer, who then offers services to clients (users), can independently arrange a certain level of service to each client. Our study focuses on the SLA between a server provider and its customers, and the study of arrangements between customers and clients lies outside the scope of this paper.

The SLA for each customer $i$ includes a charge, $p_i$, that is paid by the customer to the provider for each request the provider services. It also includes a refund $c_i$ per request in case the service provider exceeds a response time requirement. As part of the response time requirement, the SLA defines, for each customer $i$, a series of $s_i$ service demand levels, where the $k$–th demand level has associated with it a maximum tolerated response time $d_{i,k}$, $1 \leq k \leq s_i$ where $d_{i,k} < d_{i,k+1}$ for all $1 \leq k < s_i$. The service demand time, $w$, of an instance of a transaction for customer $i$, $1 \leq i \leq m$, is the amount of time it

would take to process the transaction when a single application server is dedicated to processing only this transaction. We let $B_i$ be a random variable that describes the service demand time of a transaction, and define $S_i$ to be a function that maps the transaction demand $B_i$ to the service demand level $k$, $1 \leq k \leq s_i$, i.e., given that $B_i = w$, $S_i(w) = k$. We describe the time that the servicing system actually takes to process a transaction of service demand $w$ as $D_i(w)$. The probability of violating the response time requirement is $\sum_{k=1}^{s_i} P(D_i(w) > d_{i,k}|S_i(w) = k)P(S_i(w) = k)$.

We assume that all demands for customer $i$ that fall into the same service demand level have identical service demand times, i.e., that there is a constant estimator, $w_{i,k}$ for $B_i$ whenever $S_i(w) = k$. In this case, the probability of violating the response time requirement can be rewritten as $\sum_{k=1}^{s_i} P(D_i(w_{i,k}) > d_{i,k})P(S_i(w) = k)$. The SLA must then also contain a mapping from estimators $w_{i,k}$ of a service demand level to a response time $d_{i,k}$. In the case that $S_i$ is a non–decreasing "quantization" function of $B_i$, the set $\{\ell_1, \ell_2, ...\ell_k\}$ comprise the "jump points" of the quantization function such that $S_i(w) = k$, for $\ell_{i,k-1} < B_i \leq \ell_{i,k}$. A possible choice for $w_{i,k}$ is the average given by $\int_{\ell_{k-1}}^{\ell_k} w d\hat{B}_i(w)$, where $\hat{B}_i$ is the distribution function of $B_i$. When only one level is used ($s_i = 1$), then $w_i = \int_0^\infty w d\hat{B}_i(w) = E[B_i]$. The option to specify a set of demand levels covers the case of service demands with multimodal distributions, whereas a simple specification based only on the average $E[B]$ can often be limiting. Also, in practice, an SLA may be specified only in terms of worst-case assumptions, i.e. a maximum tolerated response time for the larger envisioned service demand expressed as a single demand level. Our model permits this simplification. An allocation based on a worst-case assumption, however, may result in underutilized servers when compared to an allocation based on a multiple–level service demand SLA.

We let $\Lambda_i$ be the arrival rate of requests from customer $i$. We assume that the $D_i(w_{i,k})$, $1 \leq i \leq m$, are identically distributed random variables. A provider's profits can then be written as:

$$\sum_{i=1}^{m} \Lambda_i (p_i - c_i \sum_{k=1}^{s_i} P(D_i(w_{i,k}) > d_{i,k})P(S_i = k))$$

and the provider's objective is to maximize this quantity. Note that the manner in which the provider chooses to provision its servers affects only $P(D_i(w_{i,k}) > d_{i,k})$. Hence, the provisioning problem is equivalent to one in which the goal is to provision the servers among customers such that $\sum_{i=1}^{m} \sum_{k=1}^{s_i} \Lambda_i P(D_i(w_{i,k}) > d_{i,k})\ P(S_i = k)c_i$ is minimized.

We assume that each server is work conserving, and splits its processing power evenly among all of its simultaneously active jobs. The actual time, $D_i(w)$, spent servicing the transaction depends heavily on the number of other jobs being serviced by the system. An application server typically relies on a round-robin mechanism implemented in its operating system that concurrently services multiple requests. This mechanism assures that an equal time interval ("quantum" of computation) is assigned to each simultaneous job. After this quantum of computation, the job must wait for the other concurrent jobs to each receive a quantum. This process repeats for a job until its servicing is finished. In theory, as the quantum is made very small, in the limit this servicing mechanism becomes a *processor sharing* system. In practice, the quanta are small enough such that a processor sharing system provides a fairly accurate model of serving systems. Also, in application servers, a server's servicing resources are finite, such that the number of jobs that can be serviced simultaneously is bounded. Since SLAs incorporate delay guarantees, it is unlikely that the number of jobs in the system reaches this upper bound. Thus, it is safe to assume for simplicity of analysis that there is no limit on the number of simultaneous jobs. This allows us to model the application server as an unbounded Processor Sharing (PS) queueing system.

The parameters described above are assumed to be available as inputs to the provider such that it may decide the number of servers, $\eta_i$, to allocate to each customer $i$ such that $\sum_{i=1}^{m} \eta_i = n$. We assume that once a subset of servers is dedicated to a particular customer, that customer's processing load is equitably balanced among that subset of servers. Hence, customer $i$'s requests, $1 \leq i \leq m$, arriving at rate $\Lambda_i$ are divided evenly among its $\eta_i$ dedicated servers, such that the arrival rate to each of the $\eta_i$ servers receives requests at rate $\lambda_i = \Lambda_i/\eta_i$.

Our goal is formally stated as follows: Find the allocation $(\lambda_1, \lambda_2, ..., \lambda_m)$ that minimizes

$$\sum_{i=1}^{m} \sum_{k=1}^{s_i} \Lambda_i c_i P(D_i(w_{i,k}) > d_{i,k})P(S_i = k) \tag{1}$$

such that

$$\Lambda_i/\lambda_i \geq 1, \forall 1 \leq i \leq m \tag{2}$$

$$\sum_{i=1}^{m} \Lambda_i/\lambda_i = n, \tag{3}$$

$$0 < \lambda_i E[B_i] < 1, \forall 1 \leq i \leq m \tag{4}$$

Note that because $P(D_i(w_{i,k}) > d_{i,k})$ is a function of $\lambda_i$, the objective function depends on the values of $\lambda_1, \lambda_2, ..., \lambda_m$. The first set of constraints (Equation 2) requires that the number of servers allocated to each customer is at least one. The second constraint given by Equation 3 says that the sum of allocations must be equal to the number of servers that the provider owns. The third constraint describes a condition, $\lambda_i E[B_i] < 1$ (Equation 4), necessary for finite response times in the stationary regime of a single processor queueing system.

## IV. Application–tier Workload Characterization

In this section, we demonstrate that the arrival process to an e–commerce website's application tier can be characterized over small to moderate timescales as a Poisson process. This analysis lays the foundation that allows us to model the serving system as a set of $M/G/1/PS$ queueing systems.

### A. Methodology

We analyze actual traces of requests for dynamic content. We apply a procedure that has been used previously by Sriram and Whitt in [20] to analyze the superposition of voice and data sources. There, they consider a random variable $V_k$ which is the sum of a consecutive sequence of $k$ interarrival times, such that $V_k = X_1 + X_2 + ... + X_k$ where $X_i$ is the time between the $i$th and $i+1$st request arrivals. The *index of dispersion for intervals (IDI)* is defined to be

$$c_k^2 = \frac{k\sigma_k^2}{(E[V_k])^2}, \text{ where } \sigma_k^2 = E[V_k^2] - (E[V_k])^2.$$

The IDI is an estimator of inter-dependence within the arrival process. One interpretation of its values is the degree of correlation exhibited by the inter-arrival times. Another interpretation is an estimation of the level of "burstiness" that is exhibited by a process. For a Poisson process the IDI equals 1; for a renewal processes the IDI is invariant with respect to the number of samples, $k$ (for further discussion refer to [20]). In addition, the IDI analysis can be used to observe the behavior of the measured process over multiple time-scales by varying $k$, the number of consecutive samples.

The use of the IDI can easily be extended to analyze the behavior of the arrival process of HTTP requests. We have traces of a department store website. A typical HTTP trace stores a record of every HTTP transaction performed by the website that generated the trace, where each record contains arrival timestamps, the requested URL, and the size (in bytes) of the object requested. The record, however, does not contain the time necessary (or time taken) for servicing the request. Using an HTTP trace, we partition the request records into contiguous intervals of equal duration, and the IDI values are computed within these limited-time intervals in order to increase the likelihood that the arrival process is stationary within the measurement interval. We compute the IDI for the arrival process of HTTP requests of dynamic content at the department store's website using numerous daily traces from each of them. We also measure the rate of requests of dynamic content.

### B. Trace Analysis

We present results of our experiments using traces from June 14th, 2001 from a department store's e-commerce website as a representative of e-commerce services. We identify requests for dynamic content (and hence requests served by the application tier) as those that contain the character ? ("question mark") in the requested URL.

We partition the logging of requests over a 24-hour period (midnight to midnight) into intervals of 30 minutes and compute the IDI for each of the 30-minute intervals. We then construct non-overlapping sequences of $k$ consecutive inter-arrival intervals (although it is also permitted to use overlapping intervals to compute the IDI). Figure 1(a) plots the IDI ($y$-axis, left–hand side) and load (arrival rate) of the arrival process (along with $y$–axis, right–hand side) over a 24-hour period when $k = 20$. The time of day in hours is varied along the $x$–axis, where 0 corresponds to 12AM (midnight). The sum of lengths of $k = 20$ consecutive inter-arrival intervals yields a time interval for computing IDI in a range from 5 to 25 seconds. The curve labeled "load" depicts the load in requests per minute. We note that the IDI values are close to one except at times when the load is low, and that the IDI remains small even during periods where the load is rapidly increasing.

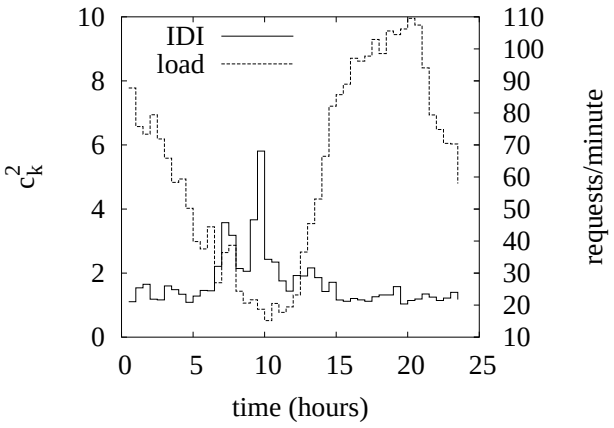



Fig. 1. IDI values and load over a 24-hour period ($k = 20$).

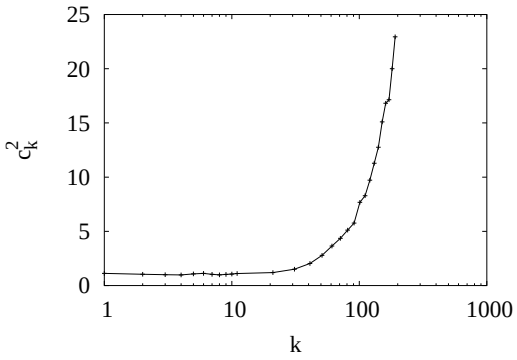



Fig. 2. The IDI test over different timescales using traces of a dept. store website.

There is an intuitive explanation as to why the arrival process of requests for dynamic content presents a different behavior than the arrival process of requests static content, whose arrivals are generally bursty. Queries for static content from a single user are often batched. The canonical example involves a web page that contains several objects, such as images and text. Hence, the interarrival times of adjacent requests to a server for static content often come from the same user and their transmission times are heavily correlated. On the other hand, the observation of low levels of correlations within the arrival process at the application tier can be explained intuitively by regarding the arrival process as a superposition of multiple users placing queries, where numerous additional

queries are placed by other users between a user's pair of queries for dynamically generated content. In contrast with an arrival process of requests of static content, a request of dynamic content (to the application serving tier) typically requires processing of a single job instead of a batch of transactions. Therefore, in timescales (seconds) on the order of lengths of interval between user actions, we expect the incoming requests to come from different users. Since each user's actions are independent from the actions of the other users, there is little correlation in the superposition process of all arrivals.

Figure 2(a) plots, using a logarithmic scale, the IDI values as $k$ is varied along the $x$-axis. The value of the IDI is close to one for values of $k$ up to 30, and then increases rapidly with further increasing $k$. We conclude that over small to moderate timescales, the arrival process behaves like a Poisson process, but that for larger timescales, there is a high degree of correlation among arrivals. The observations presented here were also observed from traces taken on different days.

The intuition delineated above also explains why the IDI peaks during low-load periods. During such periods, requests come from a small population of users and the inter-arrival times of two or more requests from a same user will be highly correlated. Therefore, correlations among arrivals are expected to increase in low–load periods because of small number of sessions. For provisioning purposes, however, the behavior process during low–load periods is not of much concern, since the arrival rate $\Lambda_i$ used to provision servers will overestimate the arrival rate during these periods.

The justification that enables us to concern ourselves with correlations only up to a certain timescale comes from the theories in two independent studies developed by Grossglauser and Bolot [7] and Ryu and Elwalid [16]. These works show that for a queueing system, the timescales over which correlations exist are delimited by an upper bound termed the *Critical Time Scale* in [16]. As a result, any model that accurately captures the correlation structure, including Markovian models [7] up to the Critical Time Scale will closely approximate the behavior of the queueing system. In our case, e-commerce website transactions at the application tier are expected to complete on the order of a fraction of a second. The computation of IDI values on the timescale of tens of seconds is expected to measure correlations that well exceed the Critical Time Scale. Since we measure IDI values close to one for timescales of up to tens of seconds during high-utilization periods, a Poisson process adequately models request arrivals in the timescales of interest.

Since the arrival traffic is effectively Poisson, an $M/G/1/PS$ queueing system is a fairly accurate model of the behavior of an application server. We utilize this model to derive mechanisms that allocate servers to e–commerce websites (customers). In general, when a number of servers is allocated to a customer we model each server as an $M/G/1/PS$ queueing system. Assuming requests to this customer website are placed at each queueing system with equal probability, the arrival rate of requests is effectively equally split among the servers allocated to that customer. Hence, on average the same amount of work is placed upon each queueing system.

## V. Server Provisioning

In this section, we derive three methods that allocate servers to customers. An exact solution requires knowledge of the distribution of response times in an M/G/1/PS queue. The solutions generated by these methods approximate the optimal solution of the optimization problem posed in Section III. Approximation methods are necessary for this problem since the distribution of response times in an M/G/1/PS queue in exact form remains an open problem (for most known results refer to [27]).

### A. General Solution Procedure

In our original optimization problem (Equations 1, 2, 3, and 4), we replace $P(D_i(w_{i,k}) > d_{i,k})$ with a bounding function, $\psi_{i,k}(\lambda_i, w_{i,k},\ d_{i,k})$. The difference in the three methods is the used bounding function. We seek to minimize the objective function $\sum_{i=1}^{m}\sum_{k=1}^{s_i}\Lambda_i c_i \psi_{i,k}(\lambda_i, w_{i,k}, d_{i,k})$. Using the Karesh-Kuhn-Tucker (KKT) [2] conditions, this optimization problem can be solved via Lagrange multipliers.

First, we take the partial derivatives of the Lagrangian function for customers $i$ and $j$, each with variables $\lambda_i$ and $\lambda_j$ respectively. For the partial derivative with respect to variable $\lambda_i$ (to variable $\lambda_j$) we are able to express the Lagrange multiplier related to constraint given by Equation 3 as a function of customer $i$'s (customer $j$'s) parameters. After equating the function of customer $i$'s parameters with the one of customer $j$'s parameters we determine $\lambda_i$ as function of $\lambda_j$: $\sum_{k=1}^{s_i} c_i\lambda_i^2 \frac{\partial \psi_{i,k}}{\partial \lambda_i}(\lambda_i, w_{i,k}, d_{i,k})P(S_i = k) =$

$$\sum_{v=1}^{s_j} c_j\lambda_j^2 \frac{\partial \psi_{j,v}}{\partial \lambda_j}(\lambda_j, w_{j,v}, d_{j,v})P(S_j = v) \qquad (5)$$

We can use the above equation to solve for $\lambda_j$ in terms of $c_j, w_{j,v}, d_{j,v},\ c_i, w_{i,k}, d_{i,k},$ and $\lambda_i$. We refer to this solution as the *comparative equation*. A property of the comparative equation used in the three methods presented in this Section guarantees that $\rho_j = \lambda_j E[B_j] < 1$. Hence, Equation 4 is always satisfied. A specific $\lambda_i$ value is obtained by applying the comparative equation for each $j$ in terms of $i$, excluding exceptional cases noted below, to the equality given by Equation 3, resulting in a *first-allocation equation*.

One complication, however, is that the solution from the comparative equation may determine some set of customer's arrival rates to be greater than the actual arrival rate (for customer $i, \lambda_i > \Lambda_i$). Such a solution is of course not practical (unacceptable given the constraints), so a constraint from either Equation 2 or 4 is imposed whereby $\lambda_i = \min\{\Lambda_i, 1/E[B_i]\}$.

Thus, assuming the result of the comparative equation for customer $i$ is $\lambda_i^*$, the $\lambda_i$ value to be taken is $\lambda_i = \min\{\lambda_i^*, \Lambda_i, 1/E[B_i]\}$. Since for our methods $\lambda_i E[B_i] < 1$, we further simplify $\lambda_i = \min\{\lambda_i^*, \Lambda_i\}$. This solution determines the first-allocation equation to be derived applying $\lambda_i = \min\{\lambda_i^*, \Lambda_i\}$, $1 \le i \le m$, to Equation 3.

Still, in order to derive a first–allocation equation, it is useful to define a set $\Phi$ to contain all customers such that the comparative equation yields an acceptable solution. The cardinality of $\Phi$ is measured by $l$.

The number of servers per customer $i$, $\eta_i$, $1 \leq i \leq m$ is finally computed using a rounding procedure applied to the ratio $\Lambda_i/\lambda_i$ (details discussed in Section V-E).

We remark that, when a server provider has a sufficient number of servers to guarantee that $l = m$, the first–allocation equation can be used to derive from the comparative equation a single equation that applied to each customer results in the final allocations. This is of practical importance for a server provider has a mechanism to re-allocate servers via a single equation when total demands fluctuate.

### *B. Costs as function of average response times*

The first method, called *Average–based method*, requires knowledge of the average servicing times of every customer. This parameter is usually known or can be reasonably estimated by the provider. For a PS queue with utilization $\rho$, $0 < \rho < 1$, a job requiring $w$ units of time is expected to be completed, in average, after $w/(1-\rho)$ units of time, due to the simultaneous processing of other jobs. The slow down factor is known to converge to $1/(1-\rho)$ for large $w$ under any work-conserving discipline [8].

We approximate the number of service misses as a function of averages of response time. We apply the Markov inequality to find a bound to the *ccdf* of response time $D(w)$ distribution

$$P(D(w) \geq d) \leq \frac{E[D(w)]}{d} = \frac{w/d}{1-\rho}. \tag{6}$$

Therefore the number of service misses for a customer $i$ at demand level $k$ is approximated as $\Lambda_i P(S_i = k)\frac{w_{i,k}/d_{i,k}}{1-\rho_i}$. The optimization problem is to minimize the loss of revenue $\sum_{i=1}^{m}\sum_{k=1}^{s_i} c_i\Lambda_i\frac{w_{i,k}P(S_i=k)/d_{i,k}}{1-\rho_i}$, as described in Section III, subject to the constraints given by Equations 2, 3, and 4.

We derive the comparative equation $\lambda_i = \frac{\rho_j\gamma_{j,i}E[B_i]}{1-\rho_j+\rho_j\gamma_{j,i}}$, where $\gamma_{i,j} = \sqrt{\frac{c_iE[B_j]}{c_jE[B_i]}\frac{\sum_{k=1}^{s_i} w_{i,k}P(S_i=k)/d_{i,k}}{\sum_{v=1}^{s_j} w_{j,v}P(S_j=v)/d_{j,v}}}$. The first–allocation equation is subsequently derived:

$$\lambda_j = \frac{(1/E[B_j])\sum_{k\in\Phi}\bar{\rho}_k\gamma_{k,j}}{n-m+l-\sum_{k\in\Phi}\bar{\rho}_k(1-\gamma_{k,j})} \tag{7}$$

where $l$ is the number of customers for which the comparative equation is valid and $\bar{\rho}_k = \Lambda_k E[B_k]$. The constraint $0 < \lambda_i E[B_i] < 1$, $1 \leq i \leq m$ (Equation 4) is always satisfied, since, rewriting the result of the comparative equation, $\rho_i = \frac{\rho_j\gamma_{j,i}}{1-\rho_j+\rho_j\gamma_{j,i}}$, we find a fraction whose numerator is smaller than its denominator. Hence, the arrival rate for any $i$ is derived to be

$$\lambda_i = \min\{\frac{\rho_j\gamma_{j,i}/E[B_i]}{1-\rho_j+\rho_j\gamma_{j,i}}, \Lambda_i\}.$$

### *C. Costs as function of variance of response times*

Our second method requires knowledge of both the first and second moments of the servicing time distribution. Again, these parameters are easily estimated in practice. We call this method the *Variance–based* method, because the bound for response time distribution in this method applies Chebyshev's inequality to bound the variance of response times. An exact expression for the variance of response times involves an integration term [26], making an exact derivation difficult. We derive another bound of the complementary distribution function using an upper bound of the variance. Consider a PS queue with average demand time $E[B]$, second moment of general demand time $E[B^2]$ and utilization $\rho$. Again let an arriving request require a job demand $w$. From the exact known result for the variance of response times in a PS queue [26], we can show that $\text{var}[D(w)] \leq w\frac{\rho}{1-\rho}\frac{E[B^2]}{E[B]}\frac{1}{(1-\rho)^2}$. Zwart and Boxma [28] find an expression for the second moment of the response times in an M/G/1/PS queue in which this same expression appears as an asymptotic limit when job sizes grow large. Therefore, the bound is tighter for large size jobs. We derive a second bound for the complementary distribution of response times (*ccdf*) in an M/G/1/PS queue. First, we apply Chebyshev's inequality,$P(D(w) - E[D] \geq d) \leq \frac{\text{var}[D(w)]}{d^2}$. Using the necessary condition for stability $\rho < 1$, we find:

$$P(D(w) - E[D] \geq d) \leq \frac{E[B^2]}{E[B]}\frac{w}{(1-\rho)^3 d^2}. \tag{8}$$

*1) Optimization:* Using Equation 8, the cost per customer is approximated using $\psi_{i,k}(\lambda_i, w_{i,k}, d_{i,k}) = \frac{w_{i,k}E[B_i^2]}{d_{i,k}^2(1-\rho_i)^3E[B_i]}$, such that the overall cost for the provider is

$$\sum_{i=1}^{m}\sum_{k=1}^{s_i} c_i\Lambda_i\frac{w_{i,k}E[B_i^2]}{d_{i,k}^2(1-\rho_i)^3E[B_i]}P(S_i = k).$$

The optimization via Lagrange multipliers results in allocations $\lambda_i$, $1 \leq i \leq m$ such that $\frac{\rho_i^2\beta_i^2}{(1-\rho_i)^4} = \frac{\rho_j^2\beta_j^2}{(1-\rho_j)^4}$, where $\beta_u = \frac{c_uE[B_u^2]}{(E[B_u])^2}\sum_{k=1}^{s_u} w_{u,k}P(S_u = k)/d_{u,k}^2$. After algebraic manipulation, we obtain the comparative equation in this method:

$$\lambda_i = \frac{1}{E[B_i]}\frac{\sqrt{1+4\frac{\beta_j\rho_j}{\beta_i(1-\rho_j)^2}}-1}{\sqrt{1+4\frac{\beta_j\rho_j}{\beta_i(1-\rho_j)^2}}+1}. \tag{9}$$

The right-hand side of Equation 9 is written as a product of two fractions. The second fraction is smaller than one. Furthermore, the second fraction equals $\rho_i = \lambda_i E[B_i]$. Hence, the comparative equation for the Variance–based method yields results that satisfy the constraint given by Equation 4.

Equation 3 in this case yields a radical equation containing only one customer rate to be used as a first allocation as follows. The first allocation equation for the Variance–based method is derived from Equations 9 and 3:

$$\sum_{i\in\Phi}\left(\Lambda_i\frac{\sqrt{1+4\frac{\beta_j\rho_j}{\beta_i(1-\rho_j)^2}}+1}{\sqrt{1+4\frac{\beta_j\rho_j}{\beta_i(1-\rho_j)^2}}-1}\right) = n-m+l \tag{10}$$

It is hard to isolate the first rate variable, $\lambda_j$, to derive a first-allocation Equation. Thus, a shortcoming of this method is the difficulty of finding the first allocation. Simple numerical solution procedures such as the Bisection method [5] can be applied in this case.

### D. The EBB–based Method

Our third method utilizes bounds derived for a class of processes called Exponential Bounded Burstiness processes as defined in [25] by Yaron and Sidi, and later generalized in [21]. Zhang *et al.* find statistical guarantees for a generalized processor sharing discipline in [9] when the arrival process belongs to the class of EBB processes. From earlier results from [25], it is relatively simple to prove (shown in this section) that the Poisson process belongs to the class of E.B.B. processes. We can therefore leverage the results of Zhang *et al.* and establish a bound on the *ccdf* of response times in our $M/G/1/PS$ model.

The theory in [25] defines the class of E.B.B. processes to contain any process $X(t)$ such that there exist parameters $(\nu, \phi, \theta)$ satisfying the condition $P(\int_s^t dX(\tau) > \nu(t-s) + \sigma) \leq \phi e^{-\theta\sigma}$. We let $X(t)$ be a Poisson process with rate $\lambda$, and deduce from Proposition 3 (regarding a Bernoulli random process) in [25] that an $\alpha > 0$ can be found satisfying $P(\int_s^t dX(\tau) > (\lambda+\epsilon)(t-s) + \sigma) \leq e^{-\alpha\sigma}$, where $\epsilon > 0$. Thus, a Poisson process with rate $\lambda$ is an E.B.B. process with parameters $(\lambda+\epsilon, 1, \alpha)$.

Since a Poisson process is time–invariant, we substitute $N(t) = \int_s^\tau dX(u)$ in the previous inequality, where $t = \tau - s$, and $N(t)$ remains of the same nature of $X(t)$, i.e. Poisson process with rate $\lambda$. Here, we are able to apply a Chernoff bound, $P(N(t) > (\lambda+\epsilon)t + \sigma) \leq \frac{E[z^{N(t)}]}{z^{(\lambda+\epsilon)t+\sigma}}$. By using the $z$–transform of the Poisson process, we find $\alpha$ given an $\epsilon > 0$:

$$P(N(t) > (\lambda+\epsilon)t + \sigma) \leq \frac{e^{-\lambda t(1-z) - log(z)(\lambda+\epsilon)t}}{e^{log(z)\sigma}}. \quad (11)$$

Mapping parameters of Equation 11 to the ones of $P(\int_s^t dX(\tau) > (\lambda+\epsilon)(t-s) + \sigma) \leq e^{-\alpha\sigma}$ yields $P(N(t) > (\lambda+\epsilon)t + \sigma) \leq e^{-log(z)\sigma}$, with the necessary condition $\lambda(1-z) + (\lambda+\epsilon)log(z) \geq 0$. Furthermore given $\epsilon > 0$, $z > 1$ so that the right-hand side is a decreasing function. In this case the decay parameter is given by $log(z)$. We pick $z_m$, as a value for $z$, such that the bound decays quickly. But the condition given by $\rho(1-z_m) + (\rho+\epsilon)log(z_m) \geq 0$ must be satisfied, and, increasing $\epsilon$, we find larger values of $z_m$ satisfying this condition. However, increasing $\epsilon$, also relaxes the tightness of the bound, unless the desired range of response times is much farther than the mean value $\lambda t$. Thus, a tradeoff exists for the selection of $\epsilon$.

For convenience, we define $\lambda^* = \lambda + \epsilon$ and $\rho^* = (\lambda + \epsilon)E[B] = \rho + \epsilon E[B]$. Applying the upper rate parameter, $\lambda + \epsilon$, and the decay parameter, $log(z_m)$, to one of Zhang et al. results we find that the bound of probability of delay $D$ (Eq. 36 of [9]): $P(D \geq d) \leq \phi^* e^{-\alpha g d}$, where $\phi^* = \frac{\phi e^{\alpha\rho^*\delta}}{1-e^{-\alpha(g-\rho^*)\delta}}$, $g$ is the fraction of processing rate for a job, and $0 < \delta < \frac{log(\phi+1)}{\alpha(g-\rho^*)}$. In our model all jobs receive equal treatment, and we use the fraction of processing rate $g = 1/\bar{n}$, where $\bar{n}$ is the number of concurrent jobs under processing. Therefore, $P(D \geq d) \leq \phi^* e^{-\alpha g d}$ is expanded to

$$P(D \geq d) \leq \frac{e^{log(z_m)\rho^*\delta}}{1 - e^{-log(z_m)(g-\rho^*)\delta}} e^{-log(z_m) g d} \quad (12)$$

Since $\phi^*$ is found in terms of $\delta$, among other parameters, we may write $\phi^*$ as a function of $\delta$. A suitable value for $\delta$ is the minimum value for the function $\phi^*(\delta)$. We select the value $\hat{\delta}$ such that the derivative $\phi^{*'}(\hat{\delta}) = 0$, $\hat{\delta} = \frac{log(\rho^*) - log(g)}{\alpha(\rho^* - g)}$.

By applying this result to Equation 12, we find a third bound for the *ccdf* of response times to be applied in our framework:

$$P(D > d) \leq \frac{\gamma^{\frac{\gamma}{\gamma-1}}}{1-\gamma} e^{-log(z_m)\frac{1-\rho}{\rho} d}, \quad (13)$$

where $\gamma = \rho^*/g$.

*1) Optimization:* We derive here the comparative and first–allocation equations for the method based on the EBB model. A drawback of the EBB method is that it only permits a single mapping of $d_i$ to $w_i$. A natural value of $w_i$ is $E[B_i]$, as discussed in Section III. Here we further bound the *ccdf* of the response times to a function $\psi_i(\lambda_i, w_i, d_i)$ that yields a more conservative cost function whose solution is tractable. In order to find $\psi_i(\lambda_i, w_i, d_i)$ we derive the inequality departing from the bound in Equation 13

$$\begin{aligned} \frac{\gamma^{\frac{\gamma}{\gamma-1}}}{1-\gamma} e^{-log(z_m)\frac{1-\rho_i}{\rho_i} d_i} &< \frac{e}{1-\gamma} e^{-log(z_m)\frac{1-\rho_i}{\rho_i} d_i} \\ &< (1+\epsilon') e^{1-log(z_m)\frac{1-\rho_i}{\rho_i} d_i} \end{aligned} \quad (14)$$

which results in $\psi_i(\lambda_i, w_i, d_i)$ defined for the EBB–based method as in the last term of the above inequality. Thus, $\psi_i(\lambda_i, w_i, d_i) = (1+\epsilon') e^{1-log(z_m)\frac{1-\rho_i}{\rho_i} d_i}$. The inequality is valid only if $\rho_i < \hat{\rho}_i$, where $\hat{\rho}_i$ is any $\rho_i$ such that $1/(1-\gamma) < 1 + \epsilon'$, $\epsilon' > 0$. Thus, here we add an extra constraint to the original problem, but most likely a system can be overall provisioned such that this condition holds true.

We solve the problem of allocation, as defined in our framework, minimizing the total cost function $\sum_{i=1}^m c_i \Lambda_i \psi_i(\lambda_i, w_i, d_i) P(S_i = k)$. Since the constant $\epsilon'$ is used across all customers as a multiplicative constant, we need to solve the optimization problem as formulated in our general model (Section III) with the function $c_i \Lambda_i e^{1-log(z_m)\frac{1-\rho_i}{\rho_i} d_i}$.

We find the comparative equation for pairs of variables $\lambda_i$ and $\lambda_j$, $1 \leq i < j \leq 1$,

$$\lambda_j = \lambda_i \frac{d_j E[B_i]/E[B_j]}{d_i + \lambda_i E[B_i] \left( \frac{log(\frac{c_j d_j E[B_i]}{c_i d_i E[B_j]})}{log(z_m)} + d_j - d_i \right)}.$$

The first–allocation equation is obtained using Equation 3 for $\lambda_i$:

$$\lambda_i = \frac{(d_i/E[B_i]) \sum_{j\in\Phi} \Lambda_j E[B_j]/d_j}{n + l - m - \sum_{j\in\Phi} \frac{\Lambda_j E[B_j]}{d_j} \left( \frac{log(\frac{c_j d_j E[B_i]}{c_i d_i E[B_j]})}{log(z_m)} + d_j - d_i \right)}.$$

### E. Solving allocations via comparative and first–allocation equations

The typical procedure to find allocations involves first to construct the first–allocation equation and determine the value

of one of the rates, say $\lambda_i$. This equation make use of a parameter for $l$, which is the number of customers for which the comparative equation is ultimately valid. However, $l$ can only be identified once the values for $\lambda_j$, $1 \leq j \leq m$ are known. Here, we present an algorithm that iterates over candidate values of $\lambda_j$, $1 \leq j \leq m$, and $l$ until an appropriate solution is found.

Let us assume a first allocation is computed using a value of $l$ equal to $\hat{l}$, yet to be found. If after all comparative equations are computed and the resulting $l$ equals $\hat{l}$, then the first allocation equation is computed such that the general conditions of the problem are satisfied. Therefore, the procedure to find values for $\lambda_i$ must find a $\hat{l}$ value.

A natural first choice is to set $l = m$ and $\Phi = \{1, 2, ..., m\}$ for the first-allocation equation. Using the $\lambda_j$ value found via the first-allocation equation, each $\lambda_i$ is computed since the comparative equation gives $\lambda_i$ in terms of customer $i$'s and $j$'s parameters. The algorithm keeps a customer $i$ in $\Phi$, if $\Lambda_i/\lambda_i \geq 1$. If $\Lambda_i/\lambda_i < 1$, the customer is placed out of $\Phi$ and its $\lambda_i$ is set to $\Lambda_i$. The value of $l$ is re-evaluated by the cardinality of $\Phi$. If $l$ is re-evaluated equal to its previous value, the allocations satisfy the constraints of the problem. In this case the algorithm stops and outputs the set $(\lambda_1, ...\lambda_m)$. Otherwise, the sequence of computations of first-allocation equation and comparative equations is repeated with the new value of $l$ and the re-arranged set $\Phi$. In the worst–case, the sequence of first–allocation plus comparative equations is repeated at most $m$ times.

The selection of the index $j$ for a $\lambda_j$ value to be obtained in computations of first–allocation equation does not affect the result, if an extra procedure is taken, since from the theory $\lambda_j$ is a solution of a set of $l$ equations and $l$ variables. The extra procedure involves assuring that the index $j$ is such that the comparative equation is valid for $\lambda_j$ in the final allocation. Hence an index $j$ must be selected for a first allocation such that $\Lambda_j/\lambda_j \geq 1$ is likely.

An intuitive idea to find such index is to choose the index whose partial cost $\sum_{k=1}^{s_i} c_i \Lambda_i \psi_{i,k}(\lambda_i, w_{i,k}, d_{i,k}) P(S_i = k)$ is maximum among partial costs of all customers, since a high partial cost implies high likelihood that $\Lambda_j/\lambda_j \geq 1$.

The number of servers for a customer $i$ is determined from the value $\Lambda_i/\lambda_i$. This ratio does not necessarily result an integer value, making a rounding procedure necessary. We find the number $y = \sum_{i=1}^{m} \Lambda_i/\lambda_i - \sum_{i=1}^{m} \lfloor \Lambda_i/\lambda_i \rfloor$, where $\lfloor x \rfloor$ denotes the maximum integer smaller than $x$, and to construct the set $\Psi(y)$ containing the $y$ customers whose fractional portions of their costs are largest when estimated via $\sum_{k=1}^{s_i} c_i \Lambda_i \psi_{i,k}(\lambda_i, w_{i,k}, d_{i,k})$. Finally, we determine the number of servers $\eta_i$ to be $\eta_i = \lfloor \Lambda_i/\lambda_i \rfloor + 1$, if $i \in \Psi(y)$, and $\lfloor \Lambda_i/\lambda_i \rfloor$, otherwise.

## VI. Experiments

We perform a series of experiments via discrete–event simulation to evaluate costs for a variety of partitioning configurations of a provider's servers. For our simulations we utilize the model with Poisson arrivals and processor sharing queues in order to describe application servers. We resort to synthetic workloads for a number of customers, since we had traces of only one e-commerce website, and no knowledge of processing times for the requests contained in the traces. By using synthetic workloads we have the ability to fully characterize customers as a function of their sensitivity to response time and intensity of the workload. We compare the costs that result from application of the Average, Variance and EBB-based methods developed in the previous section to the costs obtained by simple heuristic approaches. A first heuristic sets the number of servers in proportion to a customer arrival rate times the charge per request and divided by the tolerated response time. We name this heuristic "naive". A second heuristic, termed "uniform", divides the servers evenly among existing customers. The costs produced by these methods and heuristics is compared to an estimate of the minimum cost obtained via Monte-Carlo simulation. For each experiment, an iteration of the Monte-Carlo simulation assigns servers at random among the customers that assures that each customer is assigned at least one server. We perform one million iterations per experiment, and return the lowest cost obtained.

To our knowledge, there is no study that parameterizes the time necessary to execute applications in an operating system of an application server. However, the standard SPECWeb99 used to evaluate Web server performance uses a lognormal distribution for file sizes [15]. We select a lognormal distribution to describe the distributions of $B_i$, $1 \leq i \leq m$, for this reason, and also because it permits one to vary both first and second moments.

We compare the various methods and heuristics over a suite of specific configurations of customers. Each customer $i, 1 \leq i \leq m$ can either be tolerant (labeled 'T'), when the SLA specifies a required response time for a query of less than $8E[B_i]$. A customer's SLA is severe (labeled 'S') when its required response time is $2E[B_i]$. In addition, each customer's intensity, $\bar{\rho}_i \equiv \Lambda_i E[B_i]$ can be high (labeled 'H') such that we set $\bar{\rho}_i = n/m$, or can be low (labeled 'L') such that we set $\bar{\rho}_i = 0.2n/m$. We set the standard deviation of servicing times equal to the average size $E[B_i]$, thus the variance is $(E[B_i])^2$. Hence, a customer $i$ belongs to one of four classes: TL, TH, SL, or SH. We describe a configuration as a vector containing customer description labels: $(e_1, ..., e_m)$, where $e_i \in \{$TL, TH, SL, SH$\}, 1 \leq i \leq m$. We assume only one response time level per customer, i.e. $\forall i, s_i = 1$. In this case, the demand $w_i = E[B_i]$. We assume the charges $c_i$, $1 \leq i \leq m$, are 1 for simplicity. In the absence of a first-rate allocation equation for the variance method, we solve the root for the radical equation via the Bisection method [5].

### A. Experimental Results

In our first set of experiments, we compare the costs yielded by the various methods and heuristics for five different customer configurations for a provider supporting $m = 4$ customers with $n = 20$ servers. The configurations are: A = (SH, SL, SL, SL); B= (SH, SH, SL, SL); C=(SH, TL, TH, SL); D=(SH, TH, SL, SL); E= (SL, SL, SL, SL). Figure

3 plots along the $y$-axis the costs incurred for the various allocations under these five different customer configurations. The bars labeled "average", "variance", "EBB", "uniform", "naive", and "MC" (Monte-Carlo) indicate costs incurred for the server allocation selected by the respective method (average, variance, and EBB), heuristic (uniform and naive), or Monte-Carlo estimated minimum (MC).

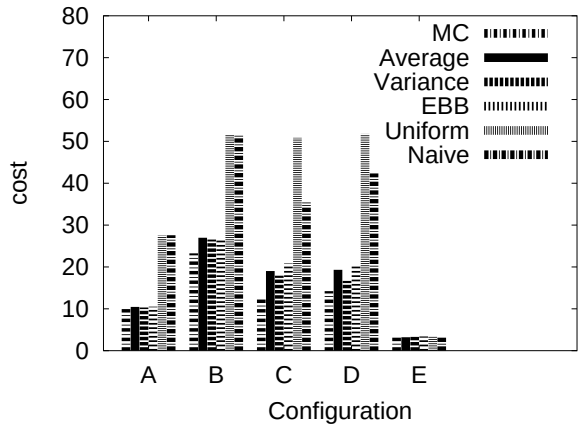
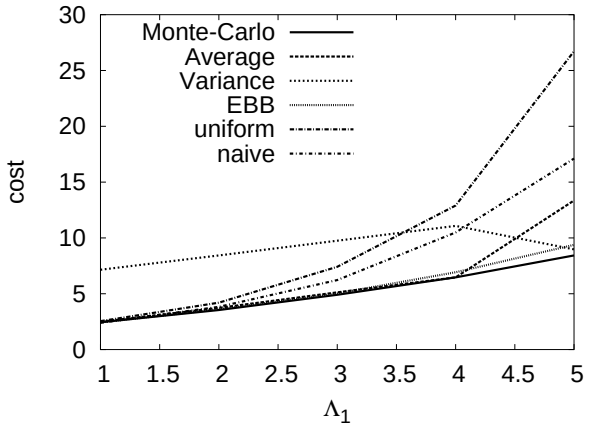



Fig. 3. Different configurations under different provisioning schemes.

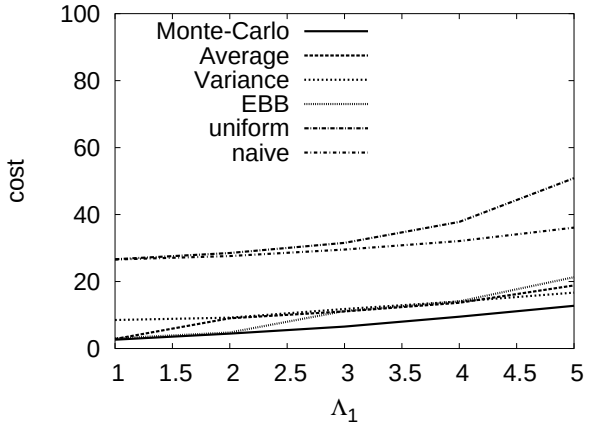
The results demonstrate that our methods achieve allocations whose costs are close to the minimum possible cost (with high likelihood), while the simple heuristic approaches generally incur significantly higher costs. The one exception is case E, where the four customers are identical. It is not surprising that in this case, the heuristics also work well. In general, the variance method comes closest to the minimum cost, with the average method and the EBB method usually yielding a slightly higher cost.

In our second set of experiments, we select two base–configurations and vary the arrival rate of a single customer in a system with $n = 20$ servers. Results are shown in Figure 4. The overall arrival rate for the first customer, $\Lambda_i$, is varied along the $x$-axis. The cost values appear along the $y$-axis. The configuration labeled "Sx" denotes that the overall arrival rate of the associated customer varies along the $x$-axis and is not simply classified as high or low. The curve labeled "Monte-Carlo" depicts the minimum cost estimated via Monte-Carlo simulation. The curves labeled "Average", "Variance", "EBB" depict results for the Average–based, Variance-based, EBB-based methods, respectively, and the curves labeled "uniform" and "naive" depict the cost for the heuristics' methods.

In Figure 4(a) the base–configuration is (Sx, SL, SL, TL). We find that the Average–based method and the EBB–based method yield configurations whose costs are remarkably close to those achieved from the configuration chosen by the Monte-Carlo simulation, and the variance method's resulting cost relatively close to the cost obtained via Monte-Carlo simulation. The most important result, however, is that cost under all three methods increases slowly with additional load (increasing arrival rate), whereas for the uniform and naive heuristics, costs increase rapidly with higher loads.




(a) Configuration (Sx, SL, SL, TL) as base.




(b) Configuration (Sx, TL, TH, SL) as base.

Fig. 4. Costs incurred when varying only one customer (1) arrival rate ($\Lambda_1$).

In Figure 4(b), the base–configuration is (Sx, TL, TH, SL). Figure 4(b) demonstrates the increase in cost that a provider will incur if the servers are allocated to customers using simple heuristics. The costs that result from applying the Average–, Variance–, and EBB–based methods are close to the estimated minimum cost.

In Figure 5, we analyze the cost generated by configurations selected via the methods and heuristics as the number of customers is varied along the $x$-axis. The costs are shown along the $y$–axis. Here, the provider offers 60 servers, where, in each experiment, a customer's configuration is set at random to TL, TH, SL, and SH with respective probabilities 0.6, 0.15, 0.15, and 0.1. We see that, irrespective of the number of customers, the costs achieved by the method-produced configurations are near-optimal, whereas the costs achieved by the heuristic-produced configurations are significantly larger.

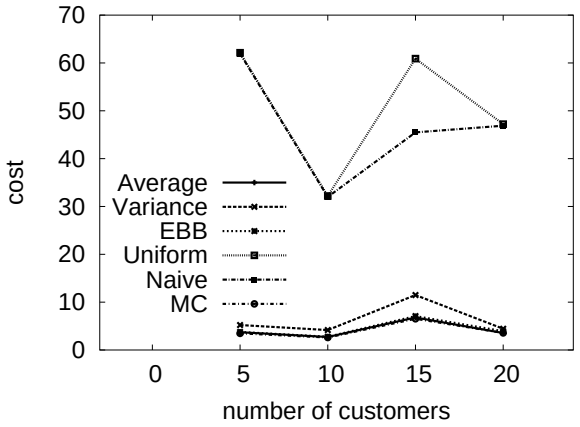



Fig. 5. Costs obtained when varying the number of customers.

## VII. Conclusion

We study methods to provision an e-commerce service provider's application-tier servers among a set of customer companies to maximize the provider's profit. We use as our setting the standard business model in which the provider forms a separate service level agreements (SLA) with each customer that indicates the profit per transaction, bound on delivery time, and the charge penalty for failing to meet the delivery time bound. We devised three methods for allocating a fixed number of servers among an arbitrary set of customers with a variety of traffic demands and different SLA configurations. The first method uses only an estimation of average response time to evaluate costs for the provider. The second method utilizes an estimation of variance of response times to evaluate costs for the provider. The third method utilizes a Poisson process description under the E.B.B. model.

Analysis of traces revealed that it is reasonable to treat the arrival process of requests from a customer to the provider's application tier as a Poisson process. This allowed us to setup an optimization problem in the context of a set of $M/G/1/PS$ queueing systems.

Via Monte-carlo simulation, we showed that the costs of the derived methods are near-optimal, and are significantly lower than more naive heuristic methods. We conclude that application of these methods by a provider to provision its application tier serving resources offers a low complexity solution that can significantly increase profits.

## Acknowledgements

We thank Sambit Sahu and Erich Nahum for their valuable comments on this work.

## References

[1] V. Almeida, R. Fonseca, M. A. Mendes, and D. Menasce. Resource management policies for e-commerce servers. *Performance Evaluation Review*, 27(4), Mar. 2000.

[2] M. Carter. *Foundations of Mathematical Economics*. MIT Press, Cambridge, MA, 2001.

[3] J. Challenger, P. Dantzig, A. Iyengar, M. Squillante, and L. Zhang. Efficiently serving dynamic data at highly accessed web sites. *IEEE/ACM Transactions on Networking*, to appear.

[4] D. de Farias, A. King, and M. Squillante. Dynamic control of web server farms. In *INFORMS Reveneue Management Section Conf.*, June 2002.

[5] J. F. Epperson. *An Introduction to Numerical Methods and Analysis*. J. Wiley, New York, NY, Aug. 2001.

[6] A. Federgruen and H. Groenvelt. The greedy procedure for resource allocation problems: necessary and sufficient conditions for optimality. *Operations Research*, 34:908–918, 1986.

[7] M. Grossglauser and J.-C. Bolot. On the long range dependence in network traffic. *IEEE/ACM Trans. on Networking*, 7(5):629–640, Oct. 1999.

[8] M. Harchol-Balter, K. Sigman, and A. Wierman. Understanding the slowdown of large jobs. *Perf. Eval. Review*, 30(3):9–11, 2002.

[9] Z. li Zhang, J. Kurose, and D. Towsley. Statistical analysis of generalized processor sharing scheduling discipline. *IEEE Journal on Selected Areas in Communications*, 13(6):1071–1080, Aug. 1995.

[10] L. Libman and A. Orda. The designer's perspective to atomic noncooperative networks. *IEEE/ACM Transactions on Networking*, 1999.

[11] Z. Liu, M. Squillante, and J. Wolf. On maximizing service-level-agreement profits. In *Proc. of ACM Conference on Electronic Commerce*, pages 213–223, Oct. 2001.

[12] Z. Liu, M. Squillante, and J. Wolf. Optimal control of resource allocation in e-business environments with strict quality-of-service performance guarantees. Technical report, IBM Research Division, 2001.

[13] D. McWherter, B. Schroeder, N. Ailamaki, and M. Harchol-Balter. Priority mechanisms for OLTP and transactional web applications. In *Proc. of International Conference on Data Engineering (ICDE 2004)*, Boston, MA, Apr. 2004.

[14] D. Menasce, V. Almeida, R. Riedi, R. Fonseca, and W. M. Jr. In search of invariants for e-business workloads. In *Proc. of ACM conference on Electronic Commerce*, pages 56–65, Minneapolis, MN, 2000.

[15] E. Nahum. Deconstructing specweb99. In *Proc. of WCW'99*, Boulder, CO, Aug. 2002.

[16] B. Ryu and A. Elwalid. The importance of long-range dependence of VBR video traffic in ATM traffic engineering: myths and realities. In *Proc. of ACM SIGCOMM'96*, pages 3–14, Palo Alto, CA, Aug. 1996.

[17] J. Sairamesh, D. Ferguson, and Y. Yemini. An approach to pricing, optimal allocation and quality of service. In *Proc. of INFOCOM'95*, pages 1111–1119, 1995.

[18] W. Shi, E. Collins, and V. Karamcheti. Modeling object characteristics of dynamic web content. *Journal of Parallel and Distributed Computing*, Sept. 2003.

[19] M. Squillante, B. Woo, and L. Zhang. Analysis of queues under correlated arrivals with applications to web server performance. *Performance Evaluation Review*, 28(4):41–43, Mar. 2001.

[20] K. Sriram and W. Whitt. Characterizing superposition arrival processes in packet multiplexers for voice and data. *IEEE Journal on Selected Areas in Communications*, 4(6):833–846, Sept. 1986.

[21] D. Starobinski and M. Sidi. Stochastically bounded burstiness for communication networks. *IEEE Transaction on Information Theory*, 46(1):206–212, Jan. 2000.

[22] A. Tantawi and D. Towsley. Optimal static load balancing in distributed computer systems. *Journal of the ACM*, 32(2):445–465, 1985.

[23] A. Tantawi, J. Wolf, and D. Towsley. Optimal allocation of multiple class resources in computers systems. In *Proc. of Sigmetrics*, pages 253–260, 1988.

[24] J. Wolf and P. Yu. On balancing the load in a clustered web farm. *ACM Transactions on Internet Technology*, 1(2):231–261, Nov. 2001.

[25] O. Yaron and M. Sidi. Performance and stability of communication networks via robust exponential bounds. *IEEE/ACM Transaction on Networking*, 1(3):372–385, 1993.

[26] S. Yashkov. A derivation of response time distribution for a $M/G/1$ processor sharing queue. *Problems of Control and Information Theory*, 12:133–148, 1983.

[27] S. F. Yashkov. Processor-sharing queues: Some progress in analysis. *Queueing Systems*, 2:1–17, 1987.

[28] B. Zwart and O. Boxma. Sojourn time asymptotics in the $M/G/1$ processor sharing queue. *Queueing Systems*, 35:141–166, 2000.