Provisioning Servers in the Application Tier for E-Commerce Systems

DANIEL VILLELA
Centro de Pesquisas de Energia Elétrica
PRASHANT PRADHAN
IBM T. J. Watson Research Center
and
DAN RUBENSTEIN
Columbia University

Server providers that support e-commerce applications as a service for multiple e-commerce Web sites traditionally use a tiered server architecture. This architecture includes an application tier to process requests for dynamically generated content. How this tier is provisioned can significantly impact a provider's profit margin. In this article we study methods to provision servers in the application serving tier that increase a server provider's profits. First, we examine actual traces of request arrivals to the application tier of an e-commerce site, 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 that 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.

Categories and Subject Descriptors: C.4 [**Performance of Systems**]—*Modeling techniques*; H.3.5 [**Information Storage and Retrieval**]: Online Information Services—*Commercial Services*; K.6.2 [**Management of Computing and Information Systems**]: Installation Management—*Pricing and resource allocation*

This material was supported in part by the National Science Foundation under Grant No. ANI-0615126 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 supported by a CNPq-Brazil scholarship (Ref. Number 200168/98-3) for his PhD program at Columbia University.

Authors' addresses: D. Villela, Centro de Pesquisas de Energia Elétrica, Cidade Universitária, Rio de Janeiro, RJ 21941-911, Brazil; email: maciel@cepel.br; P. Pradhan, IBM T. J. Watson Research Center, Hawthorne, NY 10532; email: ppradhan@us.ibm.com; D. Rubenstein, Columbia University, Department of Electrical Engineering, 500 W. 120th St. New York, NY 10025; email: danr@ee.columbia.edu.

General Terms: Performance, Management

Additional Key Words and Phrases: Eletronic commerce, server provisioning

ACM Reference Format:

Villela, D., Pradhan, P., and Rubenstein, D. 2007. Provisioning servers in the application tier for e-commerce systems. ACM Trans. Intern. Tech. 7, 1, Article 7 (February 2007), 23 pages. DOI = 10.1145/1189740.1189747 http://doi.acm.org/10.1145/1189740.1189747

1. INTRODUCTION

Companies that offer e-commerce applications often contract a third-party Web service provider to manage their 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, noncached data, such as one's personal banking records.

The amount of work performed by servers that support the front-end tier is low enough that overprovisioning is a cheap solution to meet SLA requirements. It has been shown by McWherter et al. [2004] that prioritized scheduling can improve the database tier's ability to meet its imposed SLA requirements. A provider's profits depend to a great extent on how well it provisions its application tier resources to service requests for that tier. If not configured properly, a customer can suffer numerous service misses for which the provider then pays. Surprisingly, there has been little work that assists providers in configuring their application tier.

In this article, 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:

—We identify an appropriate model for the servicing system of the application tier.

—We formalize the allocation of serving resources to multiple customers to maximize provider profits as an optimization problem.

—We derive three efficient approximation methods that allocate resources to customers, and show through simulation that the allocations 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 e–commerce Web site (department store) traces of requests for dynamic content 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 that results 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 service time of jobs for each customer is known. The second approximation requires both the first and second moments of the service time distribution. The third approximation method utilizes known bounds for the class 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 comparing the results of our approximation methods to the ones obtained via naive heuristics, service providers can decrease their costs by 60%, hence increasing their profits by a significant margin.

This article is organized as follows. In Section 2 we overview related work. Section 3 describes our model of the service provider and poses the optimization problem. In Section 4 we demonstrate that, for timescales of interest, the arrival process of requests to the application tier is effectively Poisson. Section 5 describes our approximation algorithms, and Section 6 shows results from simulations that demonstrate the performance of our algorithms. We discuss how the optimization framework enables one to determine the necessary number of servers in Section 7. We present our concluding remarks in Section 8.

2. RELATED WORK

Previous work that investigates e-commerce workloads differs from our characterization study here in that it does not address the application–tier workload. Instead, studies in Menasce et al. [2000] and Nahum [2002] focus on the workload at the front-end tier of a Web site. There has been little progress in characterizing e-commerce Web site workloads imposed specifically by jobs whose content is generated dynamically. There are studies [Challenger et al. 2004; Squillante et al. 2001; Shi et al. 2003] that investigate dynamic content but either do not analyze server workloads or do not investigate e-commerce

sites in particular. Shi et al. [2003] use an instrumented client to collect measurements, but parameters of relevance to our study, such as rate of request arrivals at a server could not be measured because measurements were taken on the client side, rather than on the server side. In Challenger et al. [2004] and Squillante et al. [2001], traces of a sporting event Web site are analyzed. The characterization of these workloads can differ from e-commerce website workloads as we show later (e.g., the distribution of time between request arrivals exhibits different properties).

Numerous performance studies focus 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 Libman and Orda [1999] 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 [1986] use an algorithmic approach to find conditions to optimize a resource allocation problem where resources are given in discrete units. This approach was subsequently generalized by Wolf and Yu [2001]. Tantawi and Towsley [1985] explored a similar problem via a graph-theoretic formulation, solving a resource allocation optimization problem. These authors further extended this work in Tantawi et al. [1988] to the case where there are multiple classes of resources. Our work is different in that our goal is not to balance load, but to maximize profits. In particular, when customers' charges differ for service misses, a simple load balancing strategy is less favorable than a strategy that diverts resources to meet the load of higher-cost service misses at the expense of missing lower-cost service misses. Sairamesh et al. [1995] 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. Furthermore, service level agreements are usually specified as a 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. [2000] propose a scheme of priority levels as a function of potential revenue estimated for different types of requests to a single Web site (front end tier). The works by Liu et al. [2001a, 2001b], de Farias et al. [2002], and Urgaonkar et al. [2005] are closest to our approach; both Liu et al. [2001a, 2001b] and de Farias et al. [2002] 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, whereas Urgaonkar et al. [2005] addresses provisioning in all tiers, using a queueing network model. There are several aspects of practical application-tier serving systems that are captured by our work and 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. 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 of de Farias et al. [2002] and Liu et al. [2001a, 2001b]. This is of great importance in practical settings where a provider can adjust its serving infrastructure when loads fluctuate over time.

3. 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 we describe. The provider has at its discretion n, $n > m$, 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. The study of arrangements between customers and their clients lies outside the scope of this article.

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 of an instance of a transaction for a customer is the amount of time it would take to process the transaction when a single application server is dedicated to processing only this transaction. Let S_i be a random variable that describes demand levels from 1 to k. Also, let B_i be the service demand time for a request for customer i. We assume that all demands for customer i that fall into the same service demand level have identical service demand times, that is, there is a constant estimator, $w_{i,k}$ for B_i whenever $S_i = k$. In this case, the probability of violating the response time requirement can be written as $\sum_{k=1}^{s_i} P(D_i(w_{i,k}) > d_{i,k})P(S_i = k)$. Hence, the SLA must then also contain a mapping from estimators $w_{i,k}$ of a service demand level to a response time $d_{i,k}$. A recommended 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 a single level is used ($s_i = 1$), then $w_i = \int_0^\infty w d\hat{B}_i(w) = E[B_i]$. A single level per customer is indeed expected to be required in practice, using the average service demand time as estimator. Our formulation permits a more general approach that could be used, for instance, when the service demand time is described by a multimodal distribution. Also, an SLA may be specified only in terms of worst-case assumptions: a maximum tolerated response time for the larger envisioned service demand expressed as a single demand level. Our model permits this type of specification. 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 Λ_i be the arrival rate of requests from customer i. Requests are always accepted into the system. A provider's profits can then be written as:

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

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, $D_i(w_{i,k})$, spent servicing the transaction depends heavily on the number of other jobs being serviced by the system. We assume that the response times $D_i(w_{i,k})$, $1 \leq i \leq m$, are identically distributed random variables. An application server typically relies on a round-robin mechanism implemented in its operating system, which concurrently services multiple requests. This mechanism ensures 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 an application server as an unbounded processor sharing (PS) queueing system.

These parameters are available as inputs to the provider such that it may decide the number of servers, η_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 Λ_i are divided evenly among its η_i dedicated servers, such that the arrival rate to each of the η_i servers is $\lambda_i = \Lambda_i / \eta_i$.

The arrival of requests at the application tier is assumed to be described by a Poisson distribution. This assumption is validated in the next section by analyzing traces taken from an e-commerce website.

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

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

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

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

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

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

4. 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.

4.1 Methodology

We analyze actual traces of requests for dynamic content. We apply a procedure that has been used previously by Sriram and Whitt [1986] 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 $V_k = X_1 + X_2 + \cdots + X_k$ where X_i is the time between the ith 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 interdependence within the arrival process. One interpretation of its values is the degree of correlation exhibited by the interarrival times. Another interpretation is an estimation of the level of "burstiness" that is exhibited by a process. The IDI of a Poisson process equals 1. The IDI of a renewal process is invariant with respect to the number of samples, k (for further discussion refer to Sriram and Whitt [1986]). 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. In this work, we use the IDI to analyze traces of a department store Web site. 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. Intrumented servers are not an option for collecting the service time, because we did not have access to the website and its server when we did this study. Rather, the traces are only available for *postmortem* analysis. Using an HTTP trace, we partition the

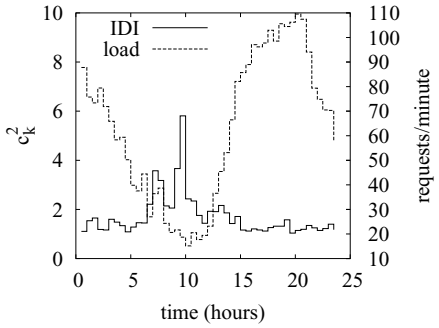


Fig. 1. IDI values and load over a 24-hour period ($k = 20$).

request records into contiguous intervals of equal duration; 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. We also measure the rate of requests of dynamic content.

4.2 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 nonoverlapping sequences of k consecutive interarrival intervals (although it is also permitted to use overlapping intervals to compute the IDI). Figure 1 plots the IDI (y-axis, left-hand side) and load (arrival rate) of the arrival process (along the y-axis, right-hand side) over a 24-hour period when $k = 20$, for a trace of requests from June 14th, 2000. The time of day in hours is varied along the x-axis, where 0 corresponds to 12AM (midnight). The sum of lengths of k consecutive interarrival intervals, where k equals 20, yields a time interval for computing the IDI in a range of 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.

Figure 2 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 results presented here were also observed from traces taken on different days.

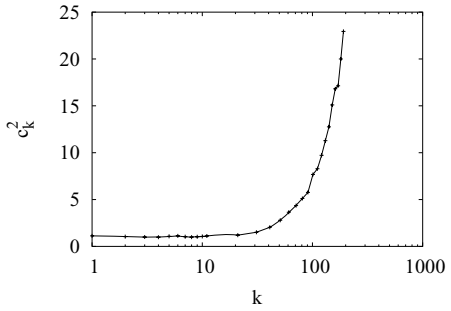


Fig. 2. The IDI test over different timescales using traces of a department store website.

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 for 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 in the same "click"—many requests in parallel—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 of the actions of the other users, there is little correlation in the superposition process of all arrivals within a short time period.

The intuition presented here also explains why the IDI peaks during low-load periods. During such periods, requests come from a small population of users and the interarrival times of two or more requests from a single user will be highly correlated. Therefore, correlations among arrivals are expected to increase in low-load periods because of the small number of sessions. For provisioning purposes, however, the behavior process during low-load periods is not of much concern, since the arrival rate Λ_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 of two independent studies developed by Grossglauser and Bolot [1999] and Ryu and Elwalid [1996]. These works show that for queueing systems, the timescales over which correlations exist are delimited by an upper bound, named the *Critical Time Scale* in Ryu and Elwalid [1996]. As a result, any model that accurately captures the correlation structure, including Markov models [Grossglauser and Bolot 1999] up to the Critical Time Scale will closely approximate the behavior of the queueing

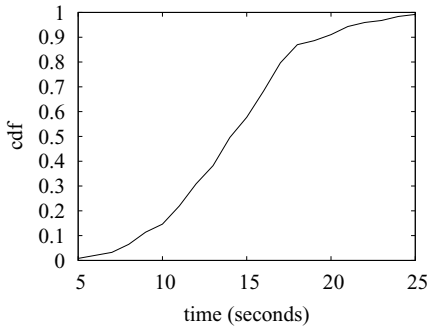


Fig. 3. Distribution of length of sum of k consecutive intervals.

system. In our case, e-commerce Web site 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.

In Figure 3, we plot the cumulative distribution function (*cdf*) of the time required to receive 20 arrivals, where the samples are drawn from a 30-minute interval, selected arbitrarily from one of the peak hours, whose load is approximately 100 requests per minute. The *cdf* values are shown along the y-axis as a function of time (seconds) which is varied along the x-axis. We see that almost always, the time required to receive 20 requests is on the order of several seconds. For instance, approximately 90% of the samples last longer than 10 seconds. Since the IDI is approximately 1 when the number of samples used per interval is 20, and since a time of several seconds intuitively lies beyond a Critical Time Scale, we find that the arrival process is adequately modeled by a Poisson process.

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.[1] 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 Web site 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 assigned to the same customer.

5. 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 3. Approximation

[1]It is interesting to note that since the process of request arrivals at the application tier exhibits different degrees of correlation when varying the timescale of interest, this process does not maintain its characteristics for a a wide range of timescales, which would demonstrate a self-similar property.

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 (refer to a survey of known results in Yashkov [1987]).

5.1 General Solution Procedure

In our original optimization problem given by the formulation in (1), (2), (3), and (4), let $\psi_{i,k}(\lambda_i, w_{i,k}, d_{i,k})$ be a bounding function for $P(D_i(w_{i,k}) > d_{i,k})$. Such a bounding function permits us to derive an approximation of the cost function. Then, the solution to our problem can be approximated by the solution to the problem using the cost approximation. Therefore, for the approximation, we seek to minimize the objective function $\sum_{i=1}^{m}\sum_{k=1}^{s_i}\Lambda_i c_i \psi_{i,k}(\lambda_i, w_{i,k}, d_{i,k})$, given the constraints in (2), (3), and (4).

It is first necessary to define the Lagrangian function $L(\lambda_1, \ldots, \lambda_m, \upsilon, \tau_1, \ldots, \tau_m) = \sum_{i=1}^{m}(\sum_{k=1}^{s_i}\Lambda_i c_i \psi_{i,k}(\lambda_i, w_{i,k}, d_{i,k}) - \tau_i(\Lambda_i/\lambda_i - 1)) - \upsilon(\sum_{i=1}^{m}\Lambda_i/\lambda_i - n)$, where υ and τ_i, $1 \le i \le m$, are Lagrange multipliers.

The Karesh-Kuhn-Tucker conditions permit us to find the solution to the problem using the Lagrangian function. These conditions are given by

$$\frac{\partial L(\lambda_1, \ldots, \lambda_m, \upsilon)}{\partial \lambda_i} = 0, 1 \le i \le m,$$

$$\frac{\partial L(\lambda_1, \ldots, \lambda_m, \upsilon)}{\partial \upsilon} = 0,$$

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

$$\tau_i \ge 0, \Lambda_i/\lambda_i \ge 1, \text{ and } \tau_i(\Lambda_i/\lambda_i - 1) = 0, 1 \le i \le m.$$

Hence, in this framework, we take the partial derivatives of the Lagrangian function for customers i and j, each with variables λ_i and λ_j respectively. In most cases, using only the Lagrange multiplier υ, we are able to express the partial derivative with respect to variable λ_i (to variable λ_j) as a function of customer i's (customer j's) parameters. After equating the multiplier υ as a function of customer i's parameters to the same multiplier υ as a function of customer j's parameters, we determine λ_i as function of λ_j. This permits us to derive λ_j, $1 \le j \le m$, $j \ne i$, as functions of an arbitrary λ_i as follows.

$$\sum_{k=1}^{s_i} c_i \lambda_i^2 \frac{\partial \psi_{i,k}}{\partial \lambda_i}(\lambda_i, w_{i,k}, d_{i,k}) P(S_i = k) =$$

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

We can use the above equation to solve for λ_j in terms of $c_j, w_{j,v}, d_{j,v}$, $c_i, w_{i,k}, d_{i,k}$, and λ_i. We refer to this formula as the *comparative equation*.

Once we have λ_j as function of an arbitrary λ_i for $1 \le j \le m$, $j \ne i$, we can use the condition $\sum_{i=1}^{m}\Lambda_i/\lambda_i = n$ to derive λ_i. This derivation permits us to find a formula for λ_i which is called the *first–allocation equation*. Since this formula utilizes only inputs to the problem, we can indeed find λ_i. We can subsequently

use this result to derive λ_j using the result from the derivation for λ_i in the first-allocation equation.

We mentioned that in most cases we can find the comparative equation using the Lagrange multiplier υ, but in some cases it is necessary to make $\lambda_i = \Lambda_i$ for a given customer i. This enforces that the arrival rate per server for a given customer cannot exceed the total arrival rate of a customer. In these cases, the derivation that required only the Lagrange multiplier υ will then require knowledge of other Lagranger multipliers from the set $\{\tau_1, \ldots, \tau_m\}$. It becomes difficult then to relate λ_j to another λ_i using only known values. Therefore, for customers in these cases, the comparative equation does not hold. Instead their allocations are given simply by making $\lambda_i = \Lambda_i$.

This restriction has an effect in the derivation of the first-allocation formula. Since its derivation requires the result from the comparative equation, the first-allocation equation is derived by applying the comparative equation for customers i from the set Φ, defined as the set of customers for which $\lambda_i < \Lambda_i$, and using $\lambda_j = \Lambda_j$ for customers j from set complementary to Φ. It is useful to define the cardinality l of Φ: $l = |\Phi|$.

The number of servers per customer i, η_i, $1 \leq i \leq m$ is finally computed using a rounding procedure applied to the ratio Λ_i/λ_i.

We remark that in practice a provider avoids underprovisioning its set of servers. This will permit $\Lambda_i > \lambda_i$ for every customer i, $1 \leq i \leq m$. In this case the result from the first-allocation equation can be applied to the comparative equation for all customers, since the comparative equation is indeed valid for all customers. In such circumstances the allocations are found using only one formula per customer given by plugging the expression from the first-allocation equation for an arbitrary λ_i into the comparative equation for any other λ_j, $1 \leq j \leq m$, $j \neq i$. *This is of practical importance when demands for customer e-commerce sites fluctuate. In that case, a server provider has a simple mechanism (a set of m equations—a single equation per customer) to periodically reallocate servers taking into account current demand measurements.*

In Section 5.3 we elaborate further on practical aspects of finding the set Ψ, the rounding procedure, and effectively computing the allocations.

5.2 Methods Given by Approximations

We define three methods that use this framework and three bounding functions for the *ccdf* of the response times. We generally assume for the analysis the steady-state response time characteristics. Little effect in response times of a queue in steady-state is observed when taking into account timescales beyond the Critical Time Scale [Grossglauser and Bolot 1999; Ryu and Elwalid 1996]. Therefore, even considering only our timescales of interest, we can use the steady-state solutions.

5.2.1 *Average-Based Method.* The first method, called *Average-based method*, requires a provider to know the average service demand times for every one of its customers. These measures can be known in practice via experimentation with a customer's application tier procedures. For a PS queue with utilization ρ, $0 < \rho < 1$, a job requiring w units of time is expected to complete,

on average, after $w/(1-\rho)$ units of time, due to the simultaneous processing of other jobs.[2]

We approximate the number of service misses as a function of averages of response time, and 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 by $\Lambda_i P(S_i = k)\frac{w_{i,k}/d_{i,k}}{1-\rho_i}$. 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 3, subject to the constraints given by (2), (3), and (4).

We follow the procedure described in the previous section to derive the comparative equation for this method:

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

where $\gamma_{i,j}$ are constants that group known parameters in the comparative equation and simplify the written expression:

$$\gamma_{i,j} = \sqrt{\frac{c_i E[B_j]}{c_j E[B_i]} \frac{\sum_{k=1}^{s_i} w_{i,k} P(S_i = k)/d_{i,k}}{\sum_{v=1}^{s_j} w_{j,v} P(S_j = v)/d_{j,v}}}.$$

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 $0 < \lambda_i E[B_i] < 1$, $1 \leq i \leq m$ in (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

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

5.2.2 *Variance-Based Method.* 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 the response time distribution in this method applies Chebyshev's inequality using the variance of response times. Consider a PS queue with average demand time $E[B]$, second

[2]The slow down factor, defined as the ratio $D(w)/w$, is known to converge to $1/(1-\rho)$ for large w under any work-conserving discipline [Harchol-Balter et al. 2002].

moment of general demand time $E[B^2]$, and utilization ρ. 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 [Yashkov 1983], we can show that $\text{var}[D(w)] \leq w\frac{\rho}{1-\rho}\frac{E[B^2]}{E[B]}\frac{1}{(1-\rho)^2}$.[3] Zwart and Boxma [2000] find this same expression on the right-hand side to be an an asymptotic limit for the variance 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 via Chebyshev's inequality. First, from Chebyshev's inequality it follows that $P(D(w) - E[D] \geq d) \leq \frac{\text{var}[D(w)]}{d^2}$. Using the necessary condition for stability, $\rho < 1$, we then find:

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

Using (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 λ_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 $\beta_u = \frac{c_u E[B_u^2]}{(E[B_u])^2}\sum_{k=1}^{s_u} w_{u,k}P(S_u = k)/d_{u,k}^2$. 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 (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 (4).

(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 (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, λ_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 [Epperson 2001] can be applied in this case.

[3]An exact expression for the variance of response times involves an integration term [Yashkov 1983], making an exact derivation difficult.

5.2.3 *The EBB–Based Method.* Our third method utilizes bounds derived for a class of processes called Exponential Bounded Burstiness (EBB) processes as defined in Yaron and Sidi [1993], and later generalized in Starobinski and Sidi [2000]. A process $R(t)$ is defined to be in the class of EBB processes if there exist parameters (ν, ϕ, θ) satisfying the condition $P(\int_s^\zeta d\,R(u) > \nu(\zeta - s) + \sigma) \leq \phi e^{-\theta\sigma}$. Zhang et al. [1995] find statistical guarantees for a generalized processor sharing discipline when the arrival process belongs to the class of EBB processes. With earlier results from Yaron and Sidi [1993], it is relatively simple to prove (shown in this section) that the Poisson process belongs to the class of EBB processes. We can therefore use the results of Zhang et al. and establish a bound on the *ccdf* of response times in our $M/G/1/PS$ model.

We now show that the Poisson process is in the class of EBB processes. We let $R(t)$ be a Poisson process with rate λ. We then deduce from Proposition 3 (regarding a Bernoulli random process) in Yaron and Sidi [1993] that an $\alpha > 0$ can be found that satisfies $P(\int_s^\zeta d\,R(u) > (\lambda + \epsilon)(\zeta - s) + \sigma) \leq e^{-\alpha\sigma}$, where $\epsilon > 0$. Thus, a Poisson process with rate λ is an EBB process with parameters $(\lambda + \epsilon, 1, \alpha)$.

We proceed to find appropriate values that can be used for parameters α and ϵ. Since a Poisson process is time-invariant, we substitute $N(t) = \int_s^\zeta d\,R(u)$ in the previous inequality, where $t = \zeta - s$, and $N(t)$ remains a Poisson process with rate λ. 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 α given an $\epsilon > 0$:

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

Mapping parameters of (11) to the ones of $P(\int_s^\zeta dR(u) > (\lambda + \epsilon)(\zeta - s) + \sigma) \leq e^{-\alpha\sigma}$ yields $P(N(t) > (\lambda + \epsilon)t + \sigma) \leq e^{-log\,(z)\sigma}$, with the necessary condition that $\lambda(1 - z) + (\lambda + \epsilon)log\,(z) \geq 0$. Furthermore, given that $\epsilon > 0$, the value of z should be greater than one 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 ϵ, we find larger values of z_m satisfying this condition. However, increasing ϵ also relaxes the tightness of the bound, unless the desired range of response times is much larger than the mean value λt. Thus, a tradeoff exists for the selection of ϵ.

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, $log\,(z_m)$, we derive a bound on the probability of response time D, (Eq. 36 of Zhang et al. [1995]): $P(D \geq d) \leq \phi^* e^{-\alpha g d}$, 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, $P(D \geq d) \leq \phi^* e^{-\alpha g d}$ expands to

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

We write ϕ^* as a function of δ. A suitable value for δ 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 (12), we find a third bound for the *ccdf* of response times to be applied in our framework:

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

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

We then derive 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, but as mentioned previously, such a condition is often used in practice. A natural value of w_i is $E[B_i]$, as discussed in Section 3. 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 (13):

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

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 ρ_i such that $1/(1-\gamma) < 1+\epsilon^{'}$, $\epsilon^{'} > 0$. Thus, here we add an extra constraint to the original problem.

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 $1+\epsilon^{'}$ is used across all customers as a multiplicative constant, we need to solve the optimization problem as formulated in our general model (Section 3) with the function $\psi_i(\lambda_i, w_i, d_i) = c_i \Lambda_i e^{1-log(z_m)\frac{1-\rho_i}{\rho_i}d_i}$.

We find the comparative equation for pairs of variables λ_i and λ_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 (3) for λ_i:

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

5.3 Solving Allocations via Comparative and First-Allocation Equations

The typical procedure to find allocations requires that one first constructs the first-allocation equation and determines the value of one of the rates, say λ_i.

This equation makes 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 λ_j, $1 \leq j \leq m$ are known. Here, we present an algorithm that iterates over candidate values of λ_j, $1 \leq j \leq m$, and l until an appropriate solution is found.

Let us assume that 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 select values for λ_i must select a value for $\hat{l}$.

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

The selection of the index j for a λ_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, λ_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 λ_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 method for finding such an 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 Λ_i/λ_i. This ratio does not necessarily result in an integer value, hence a rounding procedure is 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 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, η_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.

6. 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, 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 and 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 [Nahum 2002]. For this reason, we select a lognormal distribution to describe the distributions of B_i, $1 \leq i \leq m$, 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: $(e_1, \ldots, e_m)$, where $e_i \in$ {TL, TH, SL, SH}, $1 \leq i \leq m$. We assume only one response time level per customer: $\forall i, s_i = 1$. In this case, the demand w_i equals $E[B_i]$. 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 [Epperson 2001].

6.1 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 4 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"

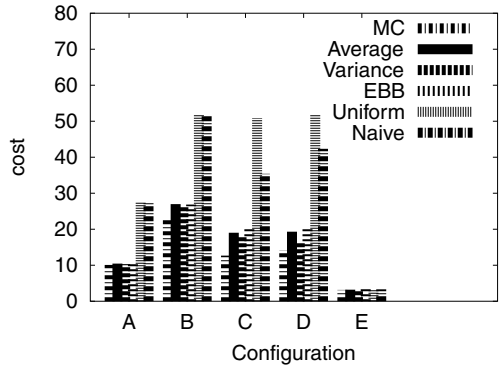
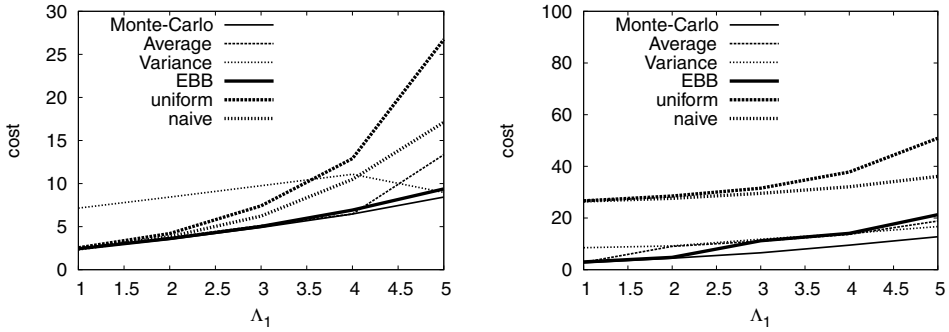


Fig. 4. Different configurations under different provisioning schemes.



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

Fig. 5. Costs incurred when varying only one customer (1) arrival rate (Λ_1).

(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).

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. In configurations C and D, for instance, the cost obtained via any of the three methods is approximately 40% of the cost obtained when using a uniform allocation. Configuration E, however, is a singular case, 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 5. The overall arrival rate for the first customer, Λ_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

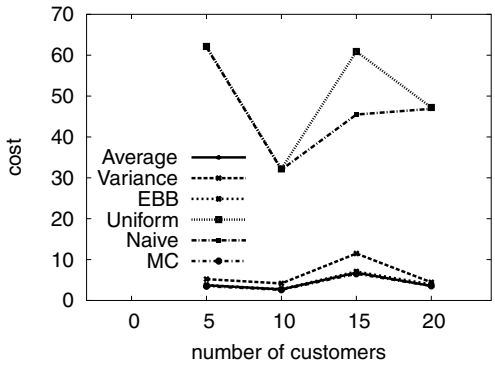


Fig. 6. Costs obtained when varying the number of customers.

simulation. The curves labeled "Average," "Variance," "EBB," depict results for the Average-based, Variance-based, and EBB-based methods, respectively, and the curves labeled "uniform" and "naive" depict the cost for the heuristics' methods.

In Figure 5(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 almost identical to those achieved from the configuration discovered by the Monte-Carlo simulation, and the variance method's resulting cost is 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.

In Figure 5(b), the base-configuration is (Sx, TL, TH, SL). Figure 5(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-based, Variance-based, and EBB-based methods are close to the estimated minimum cost.

In Figure 6, 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.

Note that a Monte-Carlo simulation is an option to find the optimal server allocations. It is indeed an alternative, but in practice there are several drawbacks in using a Monte-Carlo simulation in a realistic scenario as opposed to limited experiments. It is generally time-consuming, since one must consider a very large number of events in order to get accurate results. It becomes even worse as the number of servers and the number of customers increase, since the number of possible states to be considered in a Monte-Carlo simulation increase on a faster rate. Even if applying methods to reduce the number of events, it is

hard to make the number of events to scale as well as the linear scaling for the described methods.

7. DETERMINING AN OPTIMAL NUMBER OF SERVERS TO DEPLOY

Previously, we assumed n was a given parameter of the problem. Here, we show how to use the comparative equation and the first allocation equations, in both Average-based and EBB methods, to find the optimal number of servers where there is a cost p for each server used by the provider. We apply rates $(\lambda_1^*, \ldots, \lambda_m^*)$, obtained by the comparative and first-allocation equations to the cost function, to define the function $\Omega(n) = \sum_{i=1}^{m} \sum_{k=1}^{s_i} c_i \Lambda_i \psi_{i,k}(\lambda_i^*, w_{i,k}, d_{i,k})$.

We assume that the SLAs are contracted such that all rates $(\lambda_1^*, \ldots, \lambda_m^*)$ are given by the comparative equation, thus $l = m$. In practical circumstances, both a provider and a customer should be expected to agree on an SLA for which this condition holds true. The provider's cost $\Omega(n)$ is still expressed as a sum of partial costs, more exactly in the form $\Omega(n) = \sum_{i=1}^{m} a_i \frac{n+g_i}{n+g_i-f_i}$, where a_i, g_i and f_i are constants that depend on the chosen method. Assuming that instantiation of servers cost a provider p units per server, a provider can estimate the necessary number of servers to minimize the cost by matching the derivative of the cost function with the price p such that

$$p - \sum_{i=1}^{m} a_i \frac{f_i}{(n + g_i - f_i)^2} = 0. \tag{15}$$

For instance, for the Average-based method, the constants a_i, g_i and f_i are:

$$\begin{aligned} a_i &= c_i \Lambda_i \sum_{k=1}^{s_i} \frac{w_{i,k}}{d_{i,k}} P(S_i = k), \\ g_i &= \gamma_{j,i} \sum_{v=1}^{m} \bar{\rho}_v (\gamma_{v,j} - 1), \\ f_i &= \gamma_{j,i} \sum_{v=1}^{m} \bar{\rho}_v \gamma_{v,j}, \end{aligned}$$

where j is taken arbitrarily, $1 \leq j \leq m$. The roots for (15) can be found via numerical methods. The procedure of finding the adequate number of servers is especially useful for a provider that needs to make adjustments "on demand."

8. CONCLUSION

We studied 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 used as our setting, the standard business model in which the provider forms separate service level agreements (SLA) with each customer, where each SLA specifies 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 EBB 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 set-up an optimization problem in the context of a set of $M/G/1/PS$ queueing systems.

We showed that the costs of the derived methods are near-optimal, and are significantly lower than more naive heuristic methods. Such methods can be integrated into already-deployed tools that are used by a hosting provider to better allocate its application servers to its customers. A demonstration of such integration has been shown in the experimental setup of Urgaonkar et al. [2005] which enables the evaluation of provisioning strategies by constructing a testbed consisting of a Linux-based server cluster that runs an auctioning site in order to model actual sites. We finally 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.

ACKNOWLEDGMENTS

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

REFERENCES

ALMEIDA, V., FONSECA, R., MENDES, M. A., AND MENASCE, D. 2000. Resource management policies for e-commerce servers. *Perf. Eval. Review 27,* 4 (Mar.).

CHALLENGER, J., DANTZIG, P., IYENGAR, A., SQUILLANTE, M., AND ZHANG, L. 2004. Efficiently serving dynamic data at highly accessed web sites. *IEEE/ACM Trans. Netw. 12,* 2.

DE FARIAS, D., KING, A., AND SQUILLANTE, M. 2002. Dynamic control of web server farms. In *INFORMS Revenue Management Section Conference.*

EPPERSON, J. F. 2001. *An Introduction to Numerical Methods and Analysis*. J. Wiley, New York, NY.

FEDERGRUEN, A. AND GROENVELT, H. 1986. The greedy procedure for resource allocation problems: necessary and sufficient conditions for optimality. *Oper. Res. 34*, 908–918.

GROSSGLAUSER, M. AND BOLOT, J.-C. 1999. On the long range dependence in network traffic. *IEEE/ACM Trans. Netw. 7,* 5 (Oct.), 629–640.

HARCHOL-BALTER, M., SIGMAN, K., AND WIERMAN, A. 2002. Understanding the slowdown of large jobs. *Perf. Eval. Review 30,* 3, 9–11.

LIBMAN, L. AND ORDA, A. 1999. The designer's perspective to atomic noncooperative networks. *IEEE/ACM Trans. Netw*.

LIU, Z., SQUILLANTE, M., AND WOLF, J. 2001a. On maximizing service-level-agreement profits. In *Proceedings of the ACM Conference on Electronic Commerce*, 213–223.

LIU, Z., SQUILLANTE, M., AND WOLF, J. 2001b. Optimal control of resource allocation in e-business environments with strict quality-of-service performance guarantees. Tech. rep., IBM Research Division.

MCWHERTER, D., SCHROEDER, B., AILAMAKI, N., AND HARCHOL-BALTER, M. 2004. Priority mechanisms for OLTP and transactional Web applications. In *Proceedings of the International Conference on Data Engineering (ICDE 2004)*. Boston, MA.

MENASCE, D., ALMEIDA, V., RIEDI, R., FONSECA, R., AND JR., W. M. 2000. In search of invariants for e-business workloads. In *Proceedings of the ACM Conference on Electronic Commerce*. Minneapolis, MN, 56–65.

NAHUM, E. 2002. Deconstructing specweb99. In *Proceedings of the WCW'99*. Boulder, CO.

RYU, B. AND ELWALID, A. 1996. The importance of long-range dependence of VBR video traffic in ATM traffic engineering: myths and realities. In *Proceedings of the ACM SIGCOMM'96*. Palo Alto, CA, 3–14.

SAIRAMESH, J., FERGUSON, D., AND YEMINI, Y. 1995. An approach to pricing, optimal allocation and quality of service. In *Proceedings of the INFOCOM'95*, 1111–1119.

SHI, W., COLLINS, E., AND KARAMCHETI, V. 2003. Modeling object characteristics of dynamic web content. *J. Para. Distrib. Comput*.

SQUILLANTE, M., WOO, B., AND ZHANG, L. 2001. Analysis of queues under correlated arrivals with applications to Web server performance. *Perf. Eval. Rev. 28,* 4 (Mar.), 41–43.

SRIRAM, K. AND WHITT, W. 1986. Characterizing superposition arrival processes in packet multiplexers for voice and data. *IEEE J. Sel. Areas Comm. 4,* 6 (Sept.), 833–846.

STAROBINSKI, D. AND SIDI, M. 2000. Stochastically bounded burstiness for communication networks. *IEEE Trans. Info. Theory 46,* 1 (Jan.), 206–212.

TANTAWI, A. AND TOWSLEY, D. 1985. Optimal static load balancing in distributed computer systems. *J. ACM 32,* 2, 445–465.

TANTAWI, A., WOLF, J., AND TOWSLEY, D. 1988. Optimal allocation of multiple class resources in computers systems. In *Proceedings of the Sigmetrics*, 253–260.

URGAONKAR, B., PACIFICI, G., SHENOY, P., SPREITZER, M., AND TANTAWI, A. 2005. An analytical model for multi-tier internet services and its applications. In *Proceedings of Sigmetrics 2005*, 291–302.

VILLELA, D., PRADHAN, P., AND RUBENSTEIN, D. 2004. Provisioning servers at the applicatior tier for e-commerce systems. In *Proceedings of the International Workshop on Quality of Service (2004)*. Montreal, Canada.

WOLF, J. AND YU, P. 2001. On balancing the load in a clustered web farm. *ACM Trans. Internet Tech. 1,* 2 (Nov.), 231–261.

YARON, O. AND SIDI, M. 1993. Performance and stability of communication networks via robust exponential bounds. *IEEE/ACM Trans. Netw. 1,* 3, 372–385.

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

YASHKOV, S. F. 1987. Processor-sharing queues: Some progress in analysis. *Queueing Syst. 2*, 1–17.

ZHANG, Z., KUROSE, J., AND TOWSLEY, D. 1995. Statistical analysis of generalized processor sharing scheduling discipline. *IEEE J. Sel. Areas in Comm. 13,* 6 (Aug.), 1071–1080.

ZWART, B. AND BOXMA, O. 2000. Sojourn time asymptotics in the $M/G/1$ processor sharing queue. *Queueing Syst. 35*, 141–166.

Received January 2005; revised March 2006; accepted August 2006