Bottlenecks and their Performance Implications in E-Commerce Systems ⋆

Qi Zhang[1], Alma Riska[2], Erik Riedel[2] and Evgenia Smirni[1]

[1] College of William and Mary, Williamsburg, VA 23187
{qizhang, esmirni}@cs.wm.edu
[2] Seagate Research, Pittsburgh, PA 15222
{Alma.Riska, Erik.Riedel}@seagate.com

Abstract. We present a detailed workload characterization of a multi-tiered system that hosts an e-commerce site. Using the TPC-W workload and via experimental measurements, we illustrate how workload characteristics affect system behavior and operation, focusing on the statistical properties of dynamic page generation. This analysis allows to identify bottlenecks and the system conditions under which there is degradation in performance. Consistent with the literature, we find that the distribution of the dynamic page generation is heavy-tailed, which is caused by the interaction of the database server with the storage system. Furthermore, by examining the queuing behavior at the database server, we present experimental evidence of the existence of statistical correlation in the distribution of dynamic page generation times, especially under high load conditions. We couple this observation with the existence (and switching) of bottlenecks in the system.
Keywords: TPC-W, bottleneck identification, workload characterization, query time distribution, autocorrelation.

1 Introduction

The ubiquity of web browsers as the standard interface for IT applications such as online news, shopping, and banking, makes web usage a daily practice. Contemporary web servers provide content that is mostly dynamically generated. Understanding the resource requirements of dynamic requests is not straight-forward. It is possible that a request will cause a substantial portion of the database to be accessed even when just a few kilobytes of text are eventually sent back to the client. Identifying the bottlenecks and the exact conditions under which they occur can lead to better system provisioning, resource allocation, and quality of service.

In this paper, we focus on the analysis of the necessary time to generate dynamic objects in an e-commerce environment, focusing on the performance behavior of a Business-to-Consumer (B2C) site. We build a three-tier architecture and use the popular TPC-W workload (the Transaction Processing Council's e-commerce benchmark) to drive the web server system. We conduct measurements in all tiers of the system aiming at providing detailed statistical information of the workload behavior, and at identifying bottleneck switching across the tiers. Our experiments show that the lower tiers

⋆ This work was partially supported by the National Science Foundation under grants CCR-0098278, and ACI-0090221.

of the system, i.e, the database server and the storage system, become the bottleneck while the resources at the front-end remain underutilized. The system capacity, directly related with the existence of bottlenecks in the system, depends not only on the load, but also on the characteristics of the system workload. Some workloads utilize system resources much faster than others, for the same number of users in the system. These workloads are more I/O bound than others. Furthermore, we observe the existence of long-range dependence in dynamic page generation times. Long-range dependence in dynamic page generation times has been previously reported in the literature [16] and has been attributed to the bursty nature of the arrival process. Here, we decouple the existence of long-range dependence and correlation from the arrival process and we strictly focus on the service process. More specifically, we find that the observed burstiness in the query time distribution is mainly due to the existence (and switching) of bottlenecks.

In this paper, we dwell on the performance implications of the statistical characteristics in the dynamic page generation of e-commerce workloads aiming to aid system modeling and design. First, knowing the distribution of the response time only as expressed by the mean of the empirical distribution and its higher moments is not enough since queuing behavior does change drastically if burstiness is present [4]. Second, from the systems perspective, observing the presence of correlation in the service process may help in identifying bottlenecks and changes in the work done in the system, which can in turn lead to the design of more effective resource allocation policies at the various tiers. Our experimental results consistently point at the existence of statistical correlation in dynamic page generation.

This paper is organized as follows. Section 2 summarizes related work. Section 3 gives an overview of the experimental environment and the tracing mechanisms we used across the various tiers. In Section 4, we present performance results that can be used for capacity planning, focusing on response time analysis for bottleneck identification. Finally, Section 5 summarizes our contributions.

2 Related Work

Understanding system workload is crucial for robust designs of web servers. Contemporary systems, including today's e-commerce servers, support objects that are mainly dynamically generated, and in contrast to static requests, they need an array of resources: CPU, memory, and I/O, resulting often in *significantly* longer service times than static requests due to the dominating cost of database activity. Multi-tiered architectures that separate the database server from the web and application servers are traditionally employed in such systems to improve performance, but introduces more points where bottlenecks may occur [2]. Furthermore, e-commerce systems have stricter requirements for quick response time, high security transactions, and persistent and reliable storage [17, 5, 8, 11] making the need of detailed workload knowledge even more pressing. Several recently published studies indicate that different e-commerce servers do share similar characteristics: arrivals are best characterized as bursty with high degrees of self-similarity, the most significant portion of requests are for dynamically generated objects, and the popularity distribution is Zipf-like [17, 10, 16, 3].

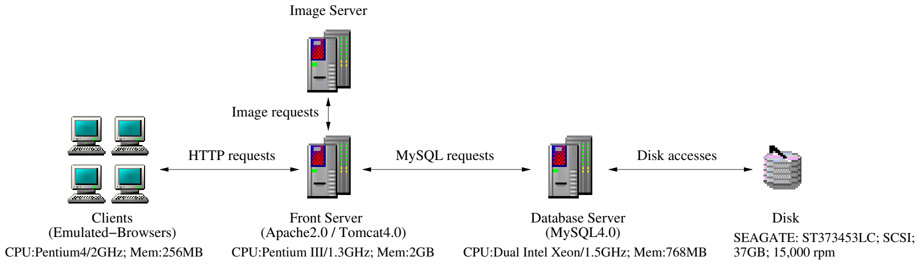


Fig. 1. The e-commerce site experimental set-up driven by the TPC-W workload.

Web server data on actual e-commerce sites are impossible to be obtained as studies on such systems are subject to non-disclosure agreements. Consequently, one can only resort to synthetic workload generators to study such systems, with the most prominent being the TPC-W benchmark [15]. Studies based on the TPC-W and its variants focus on bottleneck identification [6, 1]. In these studies, there is a consensus that mostly always the CPU of the database server is the bottleneck while for the TPC-C workload the the time spent waiting for database locks dominates the transaction execution time [9]. Vallamsetty et al. [16], using traces from actual sites that depict autocorrelation in the arrival process, show that the response time distribution of dynamic object generation is heavy-tailed and that the service process also is long-range dependent.

In contrast to all of the above works, we dissect the stochastic characteristics of the response time distribution at the back-end server. In an e-commerce system driven by TPC-W workloads, the arrival process is *not* self-similar or correlated. As such, we isolate the performance implications of the service process at the database server. We further illustrate that autocorrelation in the response time distribution can be an outcome of the nature of the service process at the database server, which we attribute to the database memory and storage system interaction.

3 Experimental Environment

We conduct measurements in an e-commerce site, implemented according to the TPC-W benchmark [15], which simulates the operation of an on-line bookstore. A high-level overview of the experimental set-up and specifics of the software/hardware used are illustrated in Figure 1.

The load in the system is measured as the number of active clients, i.e., emulated browsers (EBs). As depicted in Figure 1, we generate the load from four clients machines. In the TPC-W implementation that we use [13], all requests received by the web server are treated as dynamic requests and forwarded as queries to the database server. The database contains all the data necessary to run the e-commerce site in ten MySQL tables. Although both the number of customers and the number of items for sale determine the database size, we found that the number of items is critical for performance. Therefore, we present results on four databases that are distinguished by the number of available items for sale (see Table 1).

Table 1. Sizes of the different databases

Number of Items	10K	100K	500K	1M
ITEM Table Size	5.1 MB	51 MB	256 MB	510 MB
Database Size	1.5 GB	1.5 GB	1.9 GB	2.1 GB

TPC-W defines 14 different Web interactions which are coarsely classified as either browsing or ordering. According to the weight of each type of activity in a given traffic mix, TPC-W defines 3 types of traffic mixes: *browsing mix*, *shopping mix* and *ordering mix* in the order of increasing portion of ordering interactions [15]. Because we focus on understanding the system behavior under a variety of workloads, we introduce two additional traffic mixes that stress the system further by increasing the I/O traffic:

- *modified browsing mix*, where the percentage of new product searches accounts for 90% of all browsing requests while in the browsing mix new product searches are only 11% of requests, and

- *modified ordering*, where we raise the portion of administration interactions of the ordering mix from originally 0.11% to 10%.

Each experiment runs for 60 minutes, of which we discard the first 20 minutes in order to mask out warm-up effects.

4 Workload Analysis

In a multi-tiered system, the load of each tier has an impact on the user-perceived performance. The personalized nature of the requests sent to e-commerce servers makes the second tier, i.e., the database server, often the bottleneck [9]. Our experiments indicate that this bottleneck is triggered either by excessive *load*, as reflected by the number of simultaneous requests in the system, or by excessive *work*, as reflected by the amount of required system resources to service the requests.

4.1 Capacity Planning

Initially, we illustrate how load (see Figure 2) and work (see Figure 3) affect user perceived performance and consequently capacity planning. Figure 2 illustrates system throughput (measured in interactions per second) at the first tier, the front-end CPU utilization, and the database CPU and memory utilizations (i.e., the second tier) as a function of the number of emulated browsers in the system (i.e., system load). We do not report on the front-end memory utilization as it is always low in our experiments. Results are presented for one workload type, the browsing mix, and four database sizes (see Section 3).

Figure 3 illustrates the effect of different workloads. In this case, we experiment with only one database, i.e., the one with 500K items, and report on the system throughput, front-end CPU utilization, and database CPU and memory utilization for the ordering, shopping, and browsing workload mixes. Figure 3 indicates that workload affects system performance and resources availability. Observe that different workloads determine different levels of the sustainable system load, Finally, Figures 2 and 3 show that it

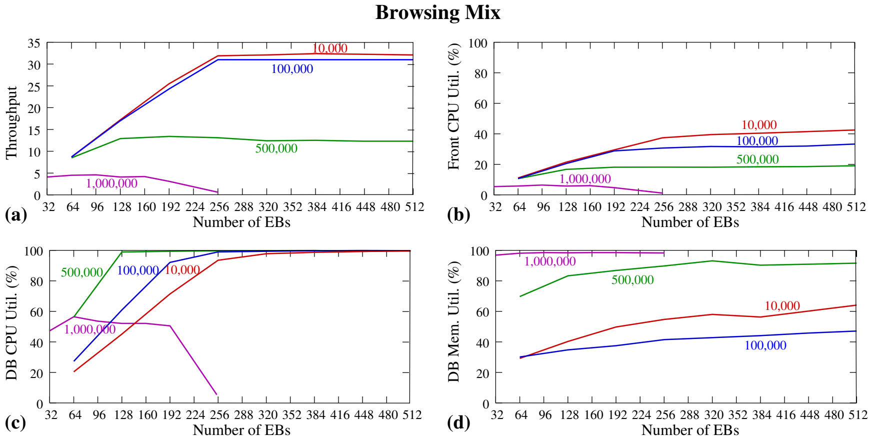


Fig. 2. (a) Throughput, (b) front-end CPU utilization, (c) database CPU utilization, and (d) database memory utilization for the browsing mix and database sizes of 10,000, 100,000, 500,000, and 1,000,000 items.

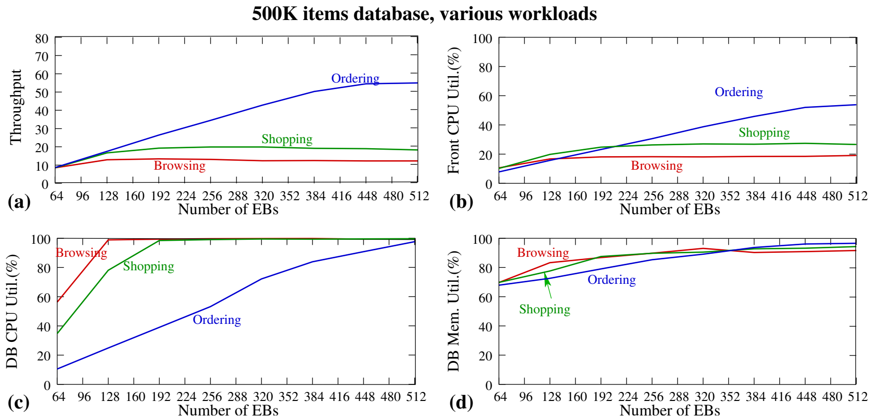


Fig. 3. (a) Throughput, (b) front-end CPU utilization, (c) database CPU utilization, and (d) database memory utilization as a function of system load for the database with 500,000 items and various workload mixes.

is the database server that becomes the bottleneck independently of the load or the work in the system. In the following subsection, we focus on the database server performance and the characteristics of dynamic page generation.

4.2 Database Performance

Figure 4 illustrates query time distributions, as a metric of the database server performance. Since the database server is the bottleneck, this metric directly relates to the user perceived performance. We focus on the query time distribution for various

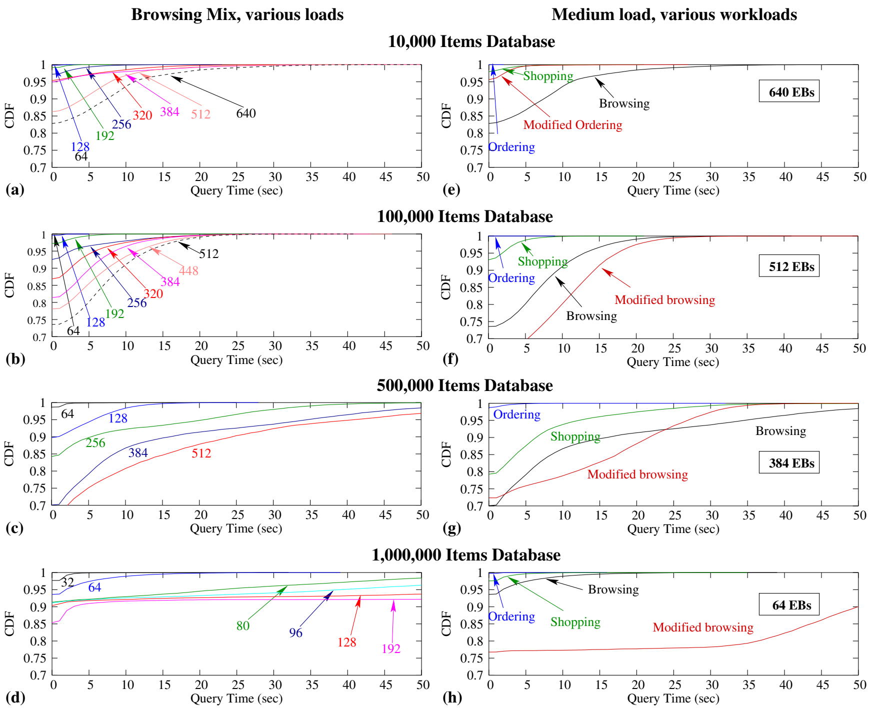


Fig. 4. Query time distribution (CDF) for the browsing mix under various loads (left column) and under medium load for various workloads (right column).

database sizes, loads, and workloads. For each database size, which essentially determines the level of system resources availability, higher load degrades database server performance. For the two large databases, where the system resources are limited, the impact of higher load is more apparent, resulting in distributions with considerably longer tails (see Figure 4(c)(d)). For various workloads (see the right column of Figure 4), the query time distribution for work-intensive workloads, such as the modified browsing mix, is quite different from that of less work-intensive workloads, such as the shopping or the ordering mix. We identify a workload as work-intensive when it requires more CPU, memory, and I/O to generate the response to a dynamic request.

As a piece of evidence that can assist us toward a more comprehensive understanding of the system behavior, we also plot the autocorrelation function (ACF) of the query times. The ACF is the autocorrelation between observations in a sample as a function of the lag, i.e., the distance between the observations. The lag-k autocorrelation value for ($k > 0$), is computed as

$$\mathrm{corr}[X_0, X_k] = \frac{E[(X_0 - E[X])(X_k - E[X])]}{\mathrm{Var}[X]}$$

where X_0 and X_k denote two observations in the sample that are k events, i.e., lags, apart and $E[X]$ is the sample mean. Figure 5 presents the autocorrelation function of

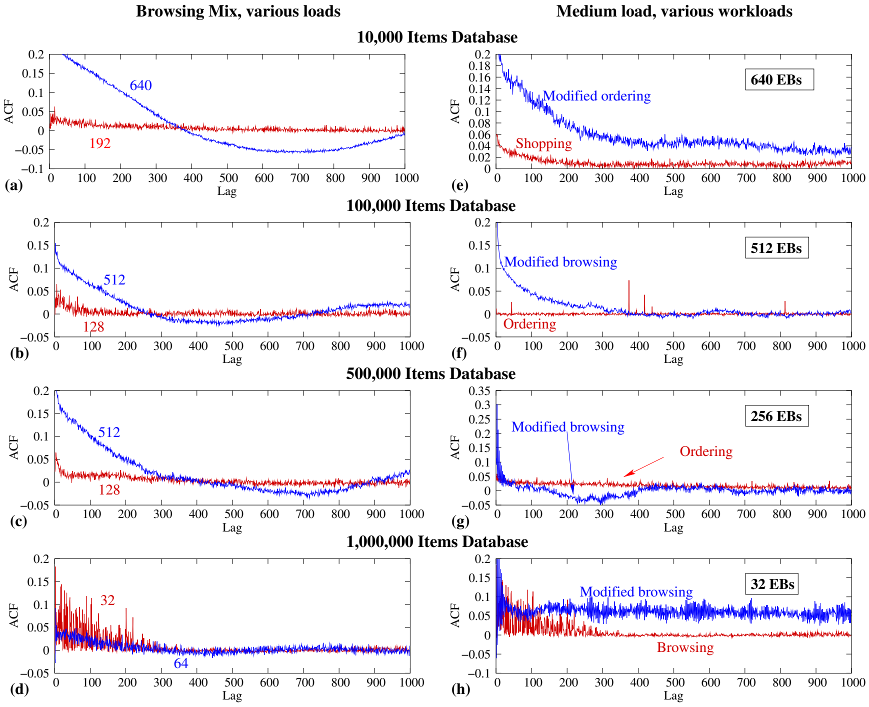


Fig. 5. ACF of query times at the database server.

query response times for various databases, loads, and workloads.

Note that the arrival process in the database is independent with measured ACF as low as 0 and as high as 0.00005 for up to lag 1000, indicating practically no autocorrelation in the arrival process.[3] Therefore, any autocorrelation that is measured in the query response time process is not due to the arrival process but depends on the scheduling discipline and the stochastic characteristics of the service process. The left column of graphs in Figure 5 shows the ACF of query times of two different load levels for each of the four database sizes. As load increases, the dependence nature of the response times increases. The graphs of the right column in Figure 5 represent the ACF of query response times for medium load but various workloads. Notice the dependence of ACF characteristics to the workload characteristics for the same system settings i.e., hardware, database size, and number of EBs. In the next subsection, we present an explanation of this behavior based on simple queuing systems and on resource consumption during the duration of the experiment.

[3] In the TPC-W workload specification, think and inter-request times at the EBs are exponentially distributed, resulting in an independent stream of arrivals to the database.

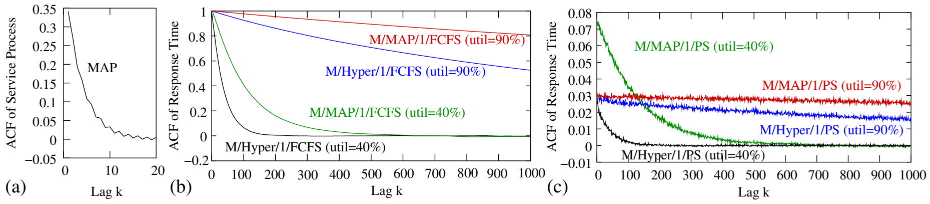


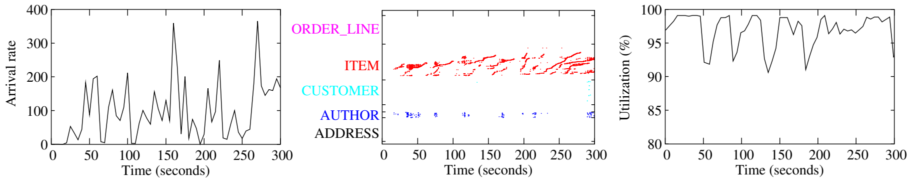
Fig. 6. (a) ACF of the service time distribution, (b) ACF of the response time distribution under FCFS, and (c) ACF of the response time distribution under PS.

4.3 Discussion and System Implications

Key to the behavior of the query response time is the interaction of the database CPU and data retrieval either from the database server memory or the disk. The ACF of the query response time sheds some light into this interaction. Recall that since there is no correlation in the arrival process at the database server, the observed ACF in the query response time is attributed only to the service process characteristics. The service process, of course, is complex as it involves two servers: the database server which operates under a time-sharing discipline and the disk which schedules requests depending on their physical location on the disk [14]. The closest theoretical scheduling schemes to those used by the database and the disk are processor sharing (PS) for the database server and first-come-first-serve (FCFS) for the disk. In reality, the operation of the system straddles between the two types of service, but looking at the two extremes it may help us understand system behavior.

Figure 6 gives examples of the expected ACF in the response time as a function of the system load (as expressed by the system utilization) and the scheduling policy for a simple queue. All results are obtained via simulation of a 10 million sample space. Figure 6(b) illustrates the ACF for a system that accepts Markovian arrivals (i.e., there is no autocorrelation in the arrival process) and uses the FCFS service discipline. Two different experiments are presented. One where the service process is a hyperexponential distribution (captures variability but not correlation [7]) and one where the service process is a Markovian Arrival Process - MAP (captures variability and correlation [12] and depicted in Figure 6(a)). To isolate the effect of autocorrelation in the service process, we select the first and second moments of the hyperexponential and MAP processes to be the same. Figure 6(b) shows that the higher the load, the higher the ACF of the response time. The presence of autocorrelation in the service process also further increases the response time ACF. Figure 6(c) illustrates the same results but for the PS discipline. Again, the presence of autocorrelation in the service process is reflected in the response time process but it is not as high as the FCFS case. In this experiment, high loads tend to sustain correlation at higher lags, indicating the presence of long-range dependence.

The ACF of the workloads that we presented in the previous subsection lie somewhere in-between the ACF curves reported in Figure 6. To better understand the behavior of the service process, we take a close look at the database behavior using the modified browsing mix (see Figure 7). The number of EBs is set to 32, which results in the significant arrival rate at the database disk (see Figure 7(a)). Note that in the same experiment but with the original browsing mix the arrival rate to the disk is at most 50. Figure 7(b) reports on the disk access pattern as a function of time. The entire disk is mapped on the y-axis which also marks the physical layout of each database table on the disk. The figure clearly shows that I/O accesses are bursty. Figure 7(c) reports on the memory utilization of the database server and shows that memory is periodically freed because of memory pressure. Intuitively, if there is memory pressure at the database, a memory miss suggests that another memory miss will soon occur, or if the disk is accessed then more accesses to the disk are to follow, which suggests that the service process is correlated. Such bulks of disk I/O operations result in heavier tail of the query-time distribution (see Figures 4(h)) and higher autocorrelation (see Figure 5), which is attributed to the slower service rates at the disk and the FCFS-like scheduling discipline. On the other hand, the burstiness of the disk accesses indicates that the system does not consistently suffer from memory misses. During the less I/O intensive period, CPU/memory performance, which employs PS scheduling discipline, dominates the response time.



(a) Arrival rate in database disk **(b) Disk access positions** **(c) Database server memory utilization**

Fig. 7. Disk access pattern for the 1,000,000 items database. The system is under 90% new searches mix.

5 Conclusions

We used measurements in a 3-tier e-commerce system, built according to the TPC-W specifications, to evaluated how workload impacts system performance. We conducted our measurements in systems with different levels of resources, which we determine by the database size. By measuring resource utilization across all tiers, we identified that the time spent at the database server (including the disk) is the one that dominates user-perceived performance. We, further, analyzed the statistical characteristics of the query time distributions, and observe the presence of autocorrelation despite the fact that in our experiments there is no autocorrelation in the arrival stream of queries at the database. We showed that the workloads which utilize mostly the database server memory and less the disk, can sustain more load, while the workloads that are more I/O oriented sustain less load as a result of a slower service process and FCFS-like service discipline at the disk. In the future, we plan to use the results that we presented in this paper to construct approximate analytic models of multi-tier systems, in order to understand their behavior for a wider range of workloads and system resources.

References

1. C. Amza, E. Cecchet, A. Chanda, A. Cox, S. Elnikety, R. Gil, J. Marguerite, K. Rajamani, and W. Zwaenepoel. Specification and implementation of dynamic content benchmarks. In *5th IEEE Workshop on Workload Characterization(WWC-5)*, Nov. 2002.
2. M. Andreolini, M. Colajanni, and R. Morselli. Performance study of dispatching algorithms in multi-tier web architectures. *ACM SIGMETRICS Performance Evaluation Review*, 30(2):10–20, Sept. 2002.
3. M. Arlitt, D. Drishnamurthy, and J. Rolia. Characterizing the scalability of a large web-based shopping system. *ACM Transactions on Internet Technology*, 1(1):44–69, 2001.
4. A. Erramilli, O. Narayan, and W. Willinger. Experimental queueing analysis with long-range dependent packet traffic. *IEEE/ACM Trans. Netw.*, 4(2):209–223, 1996.
5. G. M. C. Gama, W. M. Jr., and M. L. B. Carvalho. Resource placement in distributed e-commerce servers. In *The Evolving Global Communications Network (GLOBECOM 2001)*, San Antonio, Texas, Nov. 2001.
6. D. Garcia and J. Garcia. Tpc-w e-commerce benchmark evaluation. *IEEE Computer*, pages 42–48, Feb. 2003.
7. L. Kleinrock. *Queueing Systems, Volume I: Theory*. Wiley, 1975.
8. Z. Liu, M. Squillante, and J. Wolf. On maximizing service-level-agreement profits. In *Third ACM Conference on Electronic Commerce*, pages 213–223, Tampa, Florida, Oct. 2001.
9. D. McWherter, B. Schroeder, N. Ailamaki, and M. Harchol-Balter. Priority mechanisms for oltp and transactional web applications. In *20th International Conference on Data Engineering (ICDE)*, Boston, MA, Apr. 2004.
10. D. Menasce, V. Almeida, R. Riedi, F. Pelegrinelli, R. Fonseca, and W. Meira. In search of invariants for e-business workloads. In *Second ACM Conference on Electronic Commerce*, Minneapolis, MN, Oct. 2000.
11. D. Menasce, D. barbara, and R. Dodge. Preserving qos of e-commerce sites through self-tuning: A performance model approach. In *EC'01*, pages 224–234, Tampa, Florida, Oct. 2001.
12. M. F. Neuts. *Structured Stochastic Matrices of M/G/1-type and their Applications*. Marcel Dekker, New York, NY, 1989.
13. PHARM Project. *Java TPC-W Implementation Distribution*. Department of Electrical and Computer Engineering and Computer Sciences Department, University of Wisconsin-Madison.
14. A. Riska and E. Riedel. It's not fair - evaluating efficient disk scheduling. In *Proceedings of the 11th IEEE/ACM International Symposium on Modeling, Analysis, and Similation of Computer Telecommunication Systems, (MASCOTS)*, pages 288–295, Orlando, FL, Oct. 2003.
15. Transaction processing performance council. `http://www.tpc.org/`.
16. U. Vallamsetty, K. Kant, and P. Mohapatra. Characterization of e-commerce traffic. In *WECWIS2002*, Newport Beach, California, 2002.
17. Q. Wang, D. Makaroff, H. K. Edwards, and R. Thompson. Workload characterization for an e-commerce web site. In *CASCON 2003*, pages 313–327, Toronto, Ontario, Canada, Oct. 2003.

This article was processed using the LaTeX macro package with LLNCS style