

## Design and Implementation of a Tool for Measuring the Performance of Complex E-commerce Sites

Goedson T. Paixão  
Dept. of Computer Science  
Univ. Federal de Minas Gerais  
Belo Horizonte, MG 30161  
Brazil  
gopaixao@dcc.ufmg.br

Virgilio A. F. Almeida  
Dept. of Computer Science  
Univ. Federal de Minas Gerais  
Belo Horizonte, MG 30161  
Brazil  
virgilio@dcc.ufmg.br

Wagner Meira Jr.  
Dept. of Computer Science  
Univ. Federal de Minas Gerais  
Belo Horizonte, MG 30161  
Brazil  
meira@dcc.ufmg.br

Daniel A. Menascé  
Dept. of Computer Science  
George Mason University  
Fairfax, VA 22030  
USA  
menasce@cs.gmu.edu

Adriano M. Pereira  
Dept. of Computer Science  
Univ. Federal de Minas Gerais  
Belo Horizonte, MG 30161  
Brazil  
adrianoc@dcc.ufmg.br

### Abstract

E-commerce applications are growing at unprecedented rates, resulting in overloaded sites with poor quality of service. The workload intensity of an e-commerce site is not totally predictable given that external events can generate load spikes that exceed by far the average load. Therefore, e-commerce site managers need to be able to understand the performance of the site and be able to tune it to cope with varying traffic patterns. In this paper we present PROFIT, a new tool for profiling the performance of e-commerce sites. PROFIT measures both throughput and response time and breaks down the response time in terms of components (e.g., Web server, application server, and database server) and services (e.g., search, browse, select, add to cart, and pay). To illustrate the use of the tool, the paper shows an analysis of performance and security in e-commerce applications, measuring the impact of the Secure Sockets Layers (SSL) protocol on the request response time.

## 1 Introduction

E-commerce is one of the most important applications on the Internet, that attracts a very large number of users. Consequences of surge in volumes of e-commerce transactions are overloaded sites, with users facing long response times and system outages. E-commerce sites are significantly different from traditional web servers, both in the nature of the workload and architecture. The workload of e-commerce sites has to be characterized in terms of sessions, which are sequences of interrelated requests [6]. In order to improve performance and availability, the architecture of e-commerce sites is usually organized in multiple layers of software components and servers. This

implies that an e-commerce transaction is executed by many components, such as web servers, application servers, and database servers [2]. The processing costs at each of these components vary widely according to the functions executed. Also, the need for security in e-commerce transactions adds an extra load to e-commerce sites. Security protocols can consume significant amount of processing resources [7]. Transactions may experience congestion at each of the various servers as they queue for both hardware resources (e.g., processors and disks) as well as software resources (e.g., slot in a TCP listen queue, http thread).

The workload intensity of an e-commerce site is not totally predictable since external events can generate load spikes that exceed by far the average load. Therefore, e-commerce site managers need to be able to monitor the performance of the site, in order to understand the bottlenecks and to tune the system to cope with varying traffic patterns. For this purpose, adequate monitoring tools are needed. This paper presents *Profit*, a new tool for profiling the performance of e-commerce sites. Profit has the following characteristics. It measures both throughput and response time and breaks down the response time in terms of components (e.g., web server, application server, and database server) and services (e.g., search, browse, select, add to cart, and pay).

The rest of the paper describes in details the tool and shows examples of its use. Section 2 discusses the typical architecture of complex e-commerce sites. Section 3 presents the approach used to measuring the performance of e-commerce servers. This section describes the architecture of the tool, the measurement approach, the metrics obtained, and shows a screenshot of the GUI for Profit. The next section discusses two examples of the use of the tool. One example shows the overhead costs introduced by the use of secure connections through the Secure Sockets Layers (SSL) protocol [10]. The other example compares the processing cost of the e-commerce application when dealing with US customers versus international customers. Finally, section 5 presents the concluding remarks and discusses directions for future work.

## 2 Architecture of E-commerce Sites

From the functional viewpoint, e-commerce sites are usually organized into layers that perform classes of services. Typical services can be grouped into the following categories:

**Presentation:** It is the front-end of the electronic store. Standard functionalities available at this level include: satisfying requests for static documents and images, redirecting store requests to the proper component, and ensuring access security. This layer is implemented by Web servers that consist of both secure and non-secure HTTP servers.

**Business Logic:** It implements all application-related services. In the case of a bookstore, for instance, this layer implements services such as browse, search, select, add to shopping cart, and payment. This layer also keeps some temporary information such as customer sessions. The component that implements the business logic is called an application server.

**Database services:** It provides persistent and reliable storage for the site data. Typically, database servers are used to keep information about items in the catalog, customer identification, customer profile, inventory, and orders in progress.

### 2.1 Implementing e-Commerce Services

The processing flow of a request in an e-commerce site is illustrated in Fig. 1. A customer request arrives at the site as an HTTP request, which is then handled by the Web server. The request may be a request for a static document (e.g., an html page or an image) or it may be a request to execute an e-commerce function. In the latter case, the Web server invokes the function at the e-commerce server, through a CGI-like script or a servlet [2]. The e-commerce server performs the requested service and most of the time needs access to data stored in the database server.

While the three components mentioned above may all reside in a same computer, this is not the case in high-volume e-commerce sites. In large sites, separate computers are dedicated to

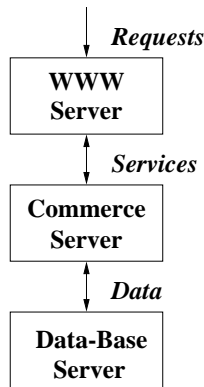


Figure 1: Logical Flow of Requests in an E-commerce Architecture

specific components of the e-commerce architecture. In many cases, there may be many instances of the same component (e.g., various Web servers), each running on a separate machine. In many cases, separate LAN segments are used to interconnect all machines that support the same type of component. Fig. 2 shows a typical architecture used in large e-commerce sites.

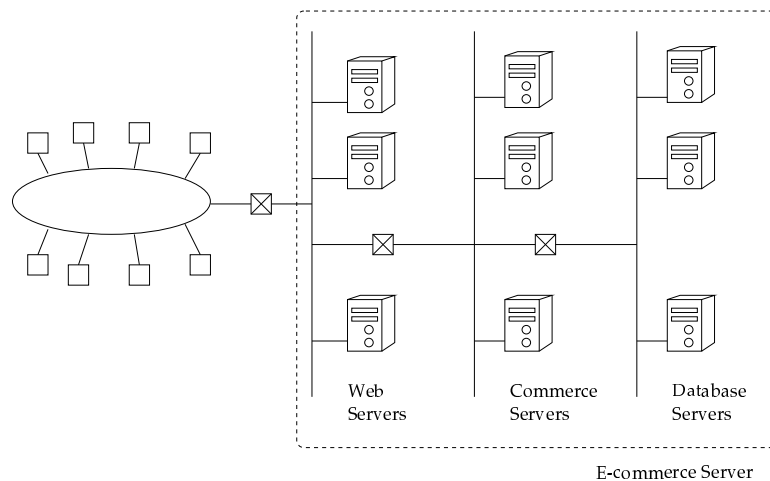


Figure 2: E-commerce Site Architecture

## 2.2 Performance and Congestion in E-commerce Sites

Two basic traditional performance metrics are used to quantify the performance of e-commerce sites: throughput and response time. Throughput measures the rate at which operations are performed. One very broad measure of throughput used by Web sites is HTTP requests/sec. HTTP requests to e-commerce sites are associated with requests for execution of e-commerce services. These services have a large variance in terms of the resource demands placed on the site's infrastructure. For example, a request to search for items in the store's catalog is bound to consume more resources than a request to retrieve the site's home page. Thus, throughput could be measured in terms of specific e-commerce operations executed per time unit. Examples could be searches/sec, checkouts/sec, or browses/sec.

Response time measures the time needed to answer a customer request. The customer response time is a sum of different components, namely: the site response time, the network latency and customer's browser formatting time. In this paper we focus on measuring the site response time, which is a function of the site architecture, the server capacity, and the software structure. We divide the factors that affect the latency perceived by customers into the following groups:

**Computation Time:** This is the total time spent executing the requested task at the various processors involved in it.

**Waiting Time:** Waiting times are due to contention for physical resources and contention for software resources. The former results from the fact that a service is delayed because it needs to use a hardware resource that is being held by another service. Software contention delays stem from tasks that have to queue up for software resources, such as database locks, semaphores, or threads.

**Communication Time:** This group comprises costs associated with communication among the various components of a site architecture.

### 2.3 Characteristics of E-commerce Workloads

A customer interacts with an e-commerce site through a sequence of interrelated requests to the site. These requests constitute a session. During a session, a customer requests e-commerce services such as browse, search, select, add to the shopping cart, and pay. The navigational pattern of a customer or group of customers can be characterized by a Customer Behavior Model Graph (CBMG) as described in [4, 6]. A CBMG is a graph in which nodes correspond to e-commerce services and arcs correspond to transitions from one service to the next in the session. Arcs are labeled by a pair of the type  $(p, z)$  where  $p$  is the probability that the transition occurs and  $z$  is the average think time. In order to characterize and model an e-commerce workload, one has to:

- identify the different types of sessions that compose the workload and generate one CBMG per session,
- calculate the workload intensity parameters, such as session arrival rates, and
- determine the resource usage parameters for each type of service request (e.g., search, browse, select, pay, etc).

A monitoring tool for an e-commerce site should aim at obtaining the data needed to calculate the parameters that characterize the site's workload.

## 3 Measuring the Performance of E-commerce Server

The design of the tool focuses on monitoring the behavior of e-commerce servers, to provide data that support capacity planning and performance tuning activities [8, 9]. The interaction between customers and an electronic store is characterized by several service requests, each of them representing a step towards a business transaction. The customer satisfaction, in terms of performance, is proportional to how fast he/she receives the response to a request. Thus, the performance metric is site response time. The tool, whose architecture is described in the next sections, provides information to explain the sources of large response times. It also helps system administrators determine the sources of performance degradation, such as contention, communication bottlenecks, and execution overhead.

### 3.1 Measurement Approach

Basically, there are four factors that influence the measurement approach used in the design of a tool to monitor the performance of an e-commerce site.

**Heterogeneity:** e-commerce sites are composed of different components, such as HTTP servers, application servers, and database servers. The performance behavior of each component depends on its capacity and the characteristics of the workload viewed by the component. For example, the Apache server keeps a pool of `httpd` processes running for answering connections. The number of processes in the pool may increase or decrease according to the request arrival rate.

**Multiprogramming:** a single host may satisfy several requests simultaneously. Multiprogramming is implemented through the allocation of one process per request (e.g., event-driven processes or thread-based processes). Multiprogramming is one of the main reasons for software and hardware contention. Devices such as the network interface are usually shared by many processes.

**Workload diversity:** a customer session is composed of several requests that differ significantly in terms of resource usage. For instance, the cost of a search is a function of the query complexity and its resource usage may vary widely from an operation such as add-to-cart.

**Monitoring cost:** a measurement tool should not be intrusive, in the sense that it does not disrupt the performance of the component being monitored [5].

Each component was instrumented at the source code level to collect measurements concerning process execution. The service information is obtained from the request and is also recorded during the execution of the component. Basically, Profit collects two measurements for each code segment: processing costs and elapsed time. The processing costs quantify the processor time spent in executing the code segment, while the elapsed time provides the wall clock time for the same execution. The performance measurements of an e-commerce site are organized in the following categories:

- **Component.** It identifies in which component the measurement was collected. It helps to locate performance problems. For instance, if the system administrator notices that all DBMS operations are quite slow, he/she may upgrade the DBMS machine.
- **Service.** It identifies the nature of the request being measured, such as search, browse or login. Knowing which services are more expensive may guide implementation enhancements. Notice that services are a function of the nature of the business application implemented by the server.
- **Phases.** The execution of a service usually comprises several phases performed in each component. For example, the processing time of an HTTP request by a Web server could be broken into three phases. Parsing phase is the interval that begins just after the establishment of the connection and ends when the header of the request has been parsed and is ready to be processed. Processing phase is the time actually spent processing the requests. Logging phase corresponds to the time spent performing standard HTTP logging. After logging, a server is ready to process a new request [1]

## 3.2 Architecture of the Tool

In this section we describe the architecture of Profit. This tool allows site administrators to measure, analyze, and visualize the performance of e-commerce sites. Performance analysis with Profit usually comprises three steps: (1) monitoring, (2) analysis, and (3) visualization. Next we describe the implementation of the first two steps. The visualization is described in Section 3.3.

In order to demonstrate the use of Profit, we implemented a prototype that analyzes the performance of electronic stores composed by three software components:

**Apache:** Apache is a very popular Web server that supports several protocols and secure access.

**Minivend:** Minivend is a Perl-based commerce server. It supports several stores and provides several facilities such as tag-based templates for specifying store contents.

**MySQL:** MySQL is an open source database management system that acts as the database server.

The stores implemented by using Minivend usually provide seven distinct services:

- Home: The entry page to the store.
- Browse: The customer verifies pre-classified items.
- Search: The customer verifies items selected according to a given query.
- Add: The customer adds a product to his/her basket for later purchase.
- Login: The customer registers in the store for purchasing an item.
- Checkout: The customer is ready to order, and provides payment and delivery information.
- Place order: The customer has chosen the items he/she wants and is ready to complete the purchase.

As mentioned before, all monitors collect three types of measurement associated with each triple component, service, and operation: user time, system time, and elapsed time.

### 3.2.1 Web Monitor

The Web Monitor collects and analyzes data from the Apache server. We distinguish three operations that are performed by an Apache server while working on a request [1]:

**Parse request:** Receives and parses the client request.

**Access check:** Verifies whether the client or origin server can issue requests to the store.

**Handle request:** Perform the operations specified by the request, which may be reading a static object from a local file, or dispatching a request to the transaction server.

### 3.2.2 Commerce Server Monitor

We distinguish four operations that are usually performed by the Minivend component for answering a customer request:

**Get request:** Receive a request from the Web server and parse it, determining the service to be performed and its parameters.

**Process request:** Process the request, managing the user session, and performing the computations necessary to accomplish the requested service.

**Access database:** Request data from the database server.

**Issue response:** Build the response page and send it to the Web server.

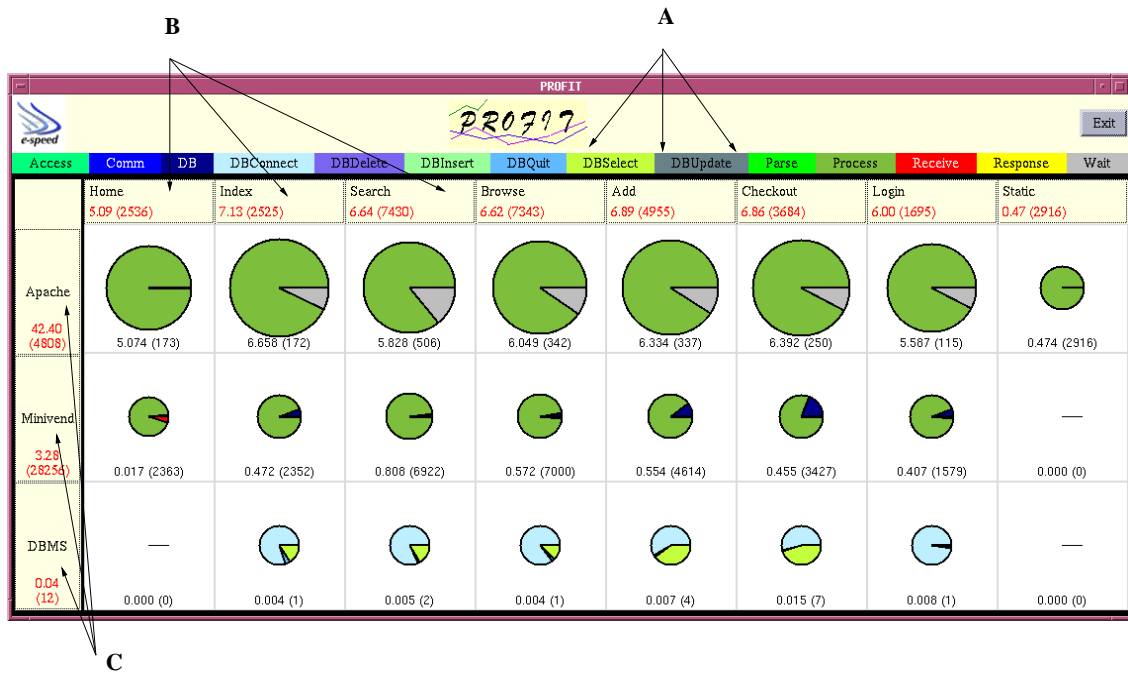


Figure 3: Profit

### 3.2.3 DBMS Monitor

The DBMS monitor provides information about the various operations usually performed by database systems:

**Insert:** Insert data into a table.

**Update:** Update data from a table.

**Connect:** Connect to a database in order to perform queries to it.

**Quit:** Close the connection to the database.

**Select:** Select data from a table.

### 3.3 The Interface of PROFIT

In the first version of Profit, users visualize the various profiles through a Tk-based graphical interface. The main window of Profit provides a fast focusing mechanism using “pies” and colors, as follows (see Figure 3). The pies are organized as a table, where each column is associated with a service and each row presents data for an architectural component. We can also observe that each row and column header contains the name of the service or component and the respective cumulative time (Figure 3 B and C). The radius of the pies vary and is proportional to the time of the associated service and component. Each slice in the pie represents the amount of time associated with a category, which may be easily identified through the reference bar above the pies (Figure 3 A). By clicking on the pie we obtain detailed information about the operations performed by each component and service.

By analyzing this display, users may easily identify the sources of client-perceived latency in terms of component, service, and process activity. For example, in Figure 3 we can observe that the service checkout presents the highest Process cost, both at the Minivend and DBMS levels.

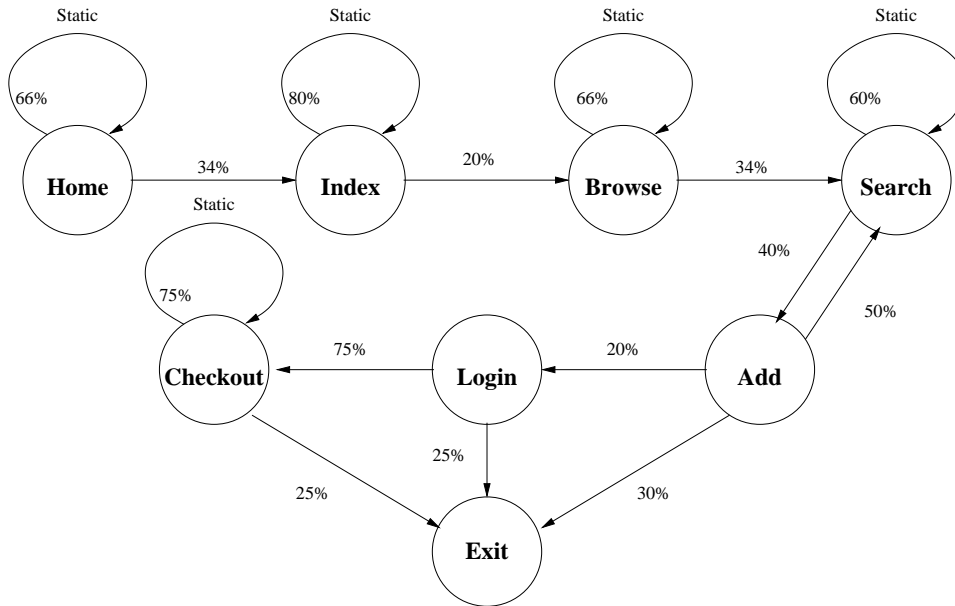


Figure 4: Customer Behavior Model Graph

## 4 Case Study

In this section we show the use of Profit for evaluating and analyzing the performance of e-commerce sites.

### 4.1 Experimental Setup

The experimental e-commerce site is organized in three levels: Web server, commerce server, and database server. The Web server is an Apache 1.3.9 server running on a 200MHz dual PentiumPro. The commerce server executes a Minivend catalog 3.14 and is a 200 MHz quad PentiumPro. The database server runs MySQL 3.22.22, an open source database, on a 200MHz Pentium server. The client is a Pentium 200Mhz PC with 64Mb RAM. All machines are on the same LAN and execute LINUX 2.2.5.

### 4.2 Workload Generation

We generated the workload for the experiments using a modified version of Surge [3] that handles sessions, i.e., sequences of requests issued by a customer. For our experiments, each session comprises at most 34 requests, including 14 images. The types of requests and the CBMG for the session is depicted in Figure 4.

During each experiment we started five processes, and each process emulates two clients (i.e., each process starts two threads). The experiment lasts for 30 minutes. The results presented in the sections that follow are the averages from three experiments.

### 4.3 Example 1: Overhead of Secure Connections

In this section we use Profit to quantify the overhead of SSL transactions by comparing the profiles of requests executing on secure and non-secure servers. In order to quantify this overhead we performed a set of experiments employing non-secure connections. The profiles are shown in Tables 2 and 3 where we can see that the use of SSL connections increased the client response time from 5.81 to 6.10 seconds (i.e., approximately 5%). Furthermore, the throughput of the non-secure version was 10% greater (for checkout) than the secure server.

Service	Average Response Time (sec)	Relative Standard Deviation	Throughput (ops/sec)	Size of Response (Bytes)
Home	36.595307	1.437508	26	270
Index	46.735409	1.487291	26	17303
Browse	22.263258	1.727070	51	8629
Search	10.481092	2.307287	66	16154
Add	12.058564	2.373890	45	16332
Checkout	47.684966	1.692389	23	7356
Login	7.669289	0.195370	10	10815
Static	0.990687	1.800952	343	5155

Table 1: Non-secure services for international customers

By verifying the profiles of each component, we can see that the use of secure connections affected both Apache and Minivend. Apache has to decrypt and encrypt the messages exchanged with the client, while Minivend has to parse the tags inherent to secure connections and to generate secure response pages. Tables 7 and 8 show the average cost and throughput for several phases for the services Home, Browse, and Checkout. We notice that the Apache server is quite overloaded, since a significant part of the client latency is associated with server processing (i.e., CGI-BIN and Minivend process creation) for both secure and non-secure servers, although the costs for secure requests are usually greater. The costs for Minivend operations also show the same trend, as indicated in Tables 9 and 10. Finally, Tables 12 and 11 show the costs of database operations for the same three services. By comparing the tables, we can observe that the cost of database operations does not vary significantly as a consequence of secure requests.

#### 4.4 Example 2: US vs. International Customers

Profit is also useful for determining sources of performance degradation in e-commerce services. We observed that the checkout time for international customers is usually much higher than the time for US customers (as we can observe in Table 1). On average, it increases from 5.81 to 23.06 seconds (i.e., almost 400%).

By verifying the performance profiles from each component (Tables 4, 5, and 6), we observe that the highest increase in response time is associated with the service Checkout, explained by a very large number of `select` operations per service execution (607). As a result, the Minivend process waits three orders of magnitude longer for database operations. Checking the logs and the Minivend source code, we learned that checkout services performed by international customers involve several `select` operations to retrieve information about the various countries. Furthermore, the high cost associated with checkout operations also affect other operations performed by the Minivend server, which also has to wait for performing database operations, as we can see in Table 5.

## 5 Conclusions and Future Work

This paper describes the design and implementation of PROFIT, a new tool for profiling the performance of e-commerce sites. PROFIT measures both throughput and response time. It also breaks down the response time in terms of components (e.g., Web server, application server, and database server) and services (e.g., search, browse, select, add to cart, and pay). We have shown two examples of the use of PROFIT, where the tool helped us to identify performance problems. We are currently extending the tool to obtain a breakdown of kernel time. This will allow us to quantify the communication and disk costs, as discussed in [1]. We are also planning to implement

Service	Average Response Time (sec)	Relative Standard Deviation	Throughput (ops/sec)	Size of Response (Bytes)
Home	5.546047	0.134592	210	270
Index	7.190153	0.116303	210	6263
Browse	6.540331	0.123612	415	3531
Search	6.356434	0.145684	616	8969
Add	6.865902	0.109594	411	5861
Checkout	6.930684	0.127299	304	3748
Login	6.002217	0.096223	141	4405
Static	0.977825	1.776961	2912	4292

Table 2: Non-secure services for US customers

Service	Average Response Time (sec)	Relative Standard Deviation	Throughput (ops/sec)	Size of Response (Bytes)
Home	5.492079	0.130767	200	283
Index	7.679366	0.089000	200	7366
Browse	7.376390	0.092726	394	9542
Search	6.477950	0.146425	582	9495
Add	7.259395	0.102764	387	6224
Checkout	7.516117	0.090454	273	11010
Login	6.099522	0.101143	140	4667
Static	1.027058	1.711809	2759	4366

Table 3: Secure Services for US customers

Service	Category	Time(secs)	Count
Home	Access	0.000720	26
Home	Parse	0.000299	26
Home	Process	5.617883	26
Home	Comm	0.003604	26
Home	Wait	39.517120	26
Browse	Access	0.000737	51
Browse	Parse	0.000330	51
Browse	Process	6.198209	51
Browse	Comm	0.000898	51
Browse	Wait	13.486590	51
Checkout	Access	0.000742	25
Checkout	Parse	0.000367	25
Checkout	Process	4.411304	25
Checkout	Comm	0.002276	25
Checkout	Wait	40.094317	25

Table 4: Non-secure international: Apache

Operation	Category	Time(secs)	Count
Home	DB	0.016895	364
Home	Process	0.459416	364
Home	Receive	0.029779	364
Home	Response	0.002684	364
Browse	DB	0.159234	1064
Browse	Process	1.537703	1064
Browse	Receive	0.088497	1064
Browse	Response	0.004200	1064
Checkout	DB	19.222906	420
Checkout	Process	13.765786	420
Checkout	Receive	0.136524	420
Checkout	Response	0.215580	420

Table 5: Non-secure International: Minivend

Operation	Category	Time(secs)	Count
Browse	DBConnect	0.031223	51
Browse	DBQuit	0.002543	51
Browse	DBSelect	0.000524	204
Checkout	DBConnect	0.015829	25
Checkout	DBQuit	0.000096	25
Checkout	DBSelect	0.028266	15175

Table 6: Non-secure International: Database

Operation	Category	Time(secs)	Count
Home	Access	0.000047	200
Home	Comm	0.002161	200
Home	Parse	0.000202	200
Home	Process	4.591339	200
Home	Wait	0.299027	200
Browse	Access	0.000053	397
Browse	Comm	0.002223	397
Browse	Parse	0.000222	397
Browse	Process	5.512299	397
Browse	Wait	1.307565	397
Checkout	Access	0.000058	273
Checkout	Comm	0.002215	273
Checkout	Parse	0.000238	273
Checkout	Process	5.135797	273
Checkout	Wait	1.831560	273

Table 7: Secure US: Apache

Operation	Category	Time(secs)	Count
Home	Access	0.000698	211
Home	Comm	0.001402	211
Home	Parse	0.000289	211
Home	Process	5.055103	211
Home	Wait	0.016885	211
Browse	Access	0.000710	418
Browse	Comm	0.000922	418
Browse	Parse	0.000319	418
Browse	Process	0.463305	418
Browse	Wait	0.583990	418
Checkout	Access	0.000715	306
Checkout	Comm	0.001268	306
Checkout	Parse	0.000346	306
Checkout	Process	5.901837	306
Checkout	Wait	0.488162	306

Table 8: Non-secure US: Apache

Operation	Category	Time(secs)	Count
Home	DB	0.011137	2800
Home	Process	0.252991	2800
Home	Receive	0.020603	2800
Home	Response	0.001645	2800
Browse	DB	0.026215	8274
Browse	Process	1.244023	8274
Browse	Receive	0.012125	8274
Browse	Response	0.001975	8274
Checkout	DB	0.013735	3822
Checkout	Process	1.767341	3822
Checkout	Receive	0.017386	3822
Checkout	Response	0.003252	3822

Table 9: Secure US: Minivend

Operation	Category	Time(secs)	Count
Home	DB	0.000331	2954
Home	Process	0.015485	2954
Home	Receive	0.000740	2954
Home	Response	0.000047	2954
Browse	DB	0.014363	8750
Browse	Process	0.549141	8750
Browse	Receive	0.006876	8750
Browse	Response	0.001260	8750
Checkout	DB	0.086908	4284
Checkout	Process	0.362929	4284
Checkout	Receive	0.004086	4284
Checkout	Response	0.000651	4284

Table 10: Non-secure US: Minivend

Operation	Category	Time(secs)	Count
Browse	DBConnect	0.004762	8274
Browse	DBQuit	0.000085	8274
Browse	DBSelect	0.000483	33096
Checkout	DBConnect	0.006976	3822
Checkout	DBQuit	0.000229	3822
Checkout	DBSelect	0.000785	7644

Table 11: Secure US: Database

Operation	Category	Time(secs)	Count
Browse	DBConnect	0.003509	8750
Browse	DBQuit	0.000095	8750
Browse	DBSelect	0.000508	35000
Checkout	DBConnect	0.007982	4284
Checkout	DBQuit	0.000085	4284
Checkout	DBSelect	0.000648	8568

Table 12: Non-secure US: database

a more detailed instrumentation of Apache so that we can quantify, for instance, the overhead of the executing CGI-bin scripts.

## References

- [1] J. Almeida, V. Almeida, and Yates D., Measuring the Behavior of a World-Wide Web Server, High Performance Networking VII, ed. Ahmed Tantawy, IFIP and Chapman & Hall, April 1997.
- [2] P. Bernstein and E. Newcomer *Principles of Transaction Processing*, Morgan Kaufmann Publishers, Inc., San Francisco, 1996.
- [3] P. Barford and M. Crovella, Generating Representative Web Workloads for Network and Server Performance Evaluation, in *Proc. 1998 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*, Madison, July 1998.
- [4] Menascé, D. A., V. Almeida, R. Fonseca, and M. A. Mendes, Resource Management Policies for E-commerce Servers, Proc. Second Workshop on Internet Server Performance (WISP'99) held in conjunction with ACM Sigmetrics'99, Atlanta, GA, May 1st, 1999.
- [5] D. Ferrari, G. Serazzi, and A. Zeigner, *Measurement and Tuning of Computer Systems*, Upper Saddle River, Prentice Real, 1983.
- [6] Menascé, D. A., V. Almeida, R. Fonseca, and M. A. Mendes, A Methodology for Workload Characterization of E-commerce Sites, *Proc. 1999 ACM Conference on Electronic Commerce*, Denver, CO, November 3-5, 1999.
- [7] R. Kinicki et al., Electronic Commerce Performance Study, Proceedings of Euromedia '98, Leicester, United Kingdom, January 1998.
- [8] Menascé, D. A. and V. Almeida, *Capacity Planning for Web Performance: metrics, models, and methods*, Prentice Hall, 1998.
- [9] Menascé, D. A., V. Almeida, and L. Dowdy, *Capacity Planning and Performance Modeling: from mainframes to client-server systems*, Prentice Hall, 1994.
- [10] S. Grafinkel and G. Spafford, *Practical Unix & Internet Security*, O'Reilly & Associates, Inc. 1996.