Logging Clickstream Data into a Database on a Consolidated System

Mark Alan Wong
B.S., Civil Engineering, Tufts University, Medford, Massachusetts, 1999

A thesis presented to the faculty of the
OGI School of Science and Engineering
at Oregon Health and Science University
in partial fulfillment of the
requirements for the degree
Master of Science
in
Computer Science and Engineering

January 2002

The thesis "Logging Clickstream Data into a Database on a Consolidated System" by Mark Alan Wong has been examined and approved by the following Examination Committee:

Dr. David Maier
Professor, Oregon Graduate Institute

Dr. Leonard Shapiro
Professor, Portland State University

Mary Edie Meredith
Staff Software Engineer, IBM Corporation

Acknowledgements

I want to thank Len Shapiro, David Maier, and Mary Meredith for their support and advice on my thesis research.

I am also grateful to Curran Filer, Marisa Rogoway, Lisa Kleinman, and especially Ang Tan for their comments and suggestions for my paper.

I owe additional gratitude to my managers, Bhagyam Moses and Tony Petrossian, and my coworkers at Sequent Computer Systems, Inc., which has become part of IBM Corporation, as they were very accommodating with my responsibilities as I worked towards finishing my degree.

Table of Contents

List of Tables

List of Figures

Abstract

In the age of growing Internet commerce, companies are eager to understand their customers' behavior in order to build strong relationships with their customers. Companies are analyzing the log files from their Web servers and are integrating that data with their data warehouses.

This thesis presents a method for storing clickstream data into a database as it is generated on a Web server on a single, highly scalable system. This study analyzes the performance effect of this method and gives performance data that show consolidation has little effect on the performance of the Web server and the database server used.

Additional benefits of this method include accelerating the process of analyzing customer behavior, reducing costs associated with maintaining a large number of systems, the ability to reallocate resources between tasks, and the ability to upgrade the system to meet the demands of a growing business.

1 Introduction

Today's most popular sites on the World Wide Web receive millions of visitors a day. The Web sites run by Microsoft (microsoft.com, expedia.com, msnbc.com, hotmail.com, etc.) receive roughly two billion hits a day from twenty-five million individuals [1]. The users accessing Microsoft's sites generate more than 200 gigabytes (GB) of data describing the Web pages requested every day, totaling over 70 terabytes (TB) of data every year, and these numbers continue to grow.

In this thesis, I propose a method of immediately storing this data from a Web server into a database on a single, highly scalable system to manage the immense amount of data created each day. This method combines processes that are typically performed on two separate systems and I show the performance implications of consolidating a Web server and a database server onto a single system.

Web servers, also known as *hypertext transfer protocol* (HTTP) servers, are the components in a Web site that send us the information we see in our Web browsers when we visit sites such as Yahoo! or Amazon.com. HTTP [2] is the protocol that allows a browser to communicate with a Web server and request Web objects. Web objects include *Hypertext Markup Language* (HTML) Web page, images, Java applets, streaming audio and video, and many other types of files.

Every Web page, image, video, and audio file requested from a Web site for viewing, listening, or reading is recorded into the Web server's log file. For example, viewing an HTML page with eleven images registers a total of twelve requests into the Web server log file, one request for the HTML page and eleven requests for each of the images displayed. Figure 1-1 displays an example of a Web page with eleven images.

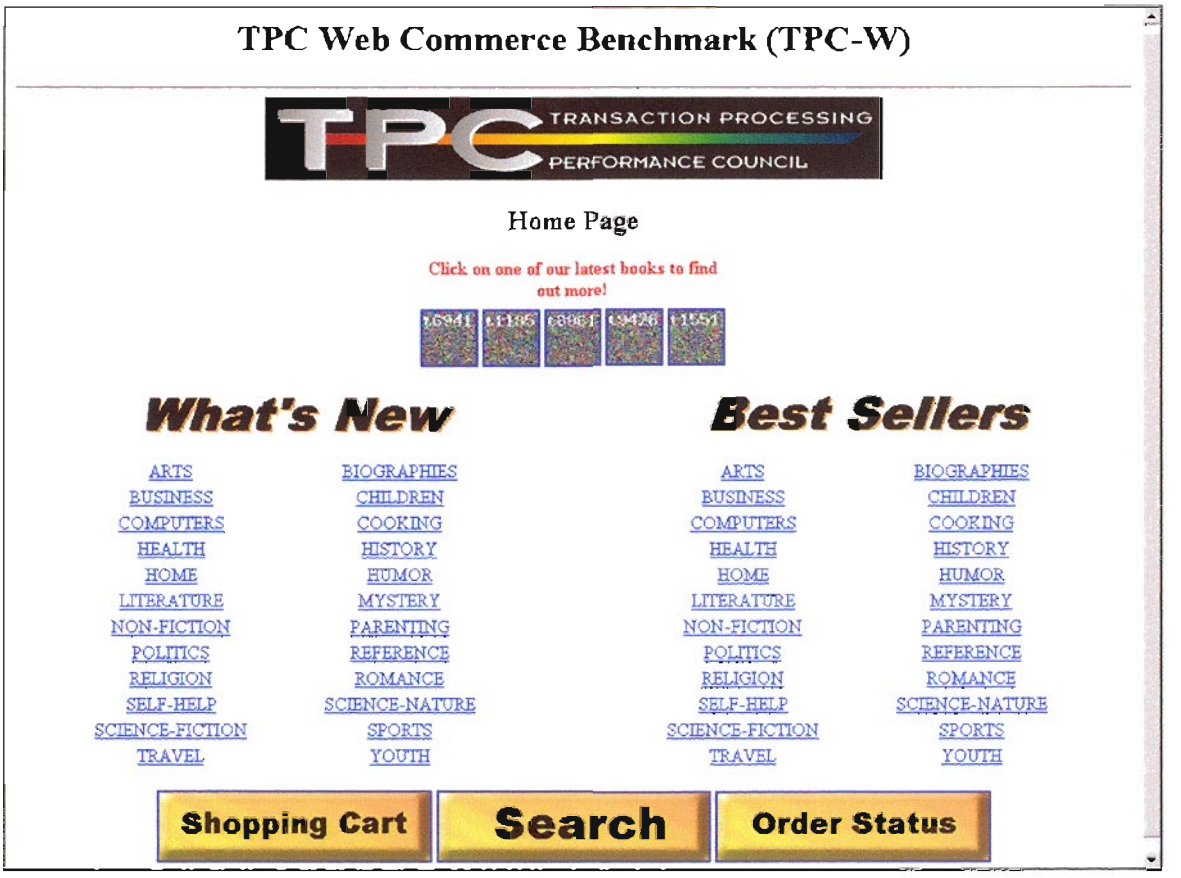

Figure 1-1: Sample Web Page

The *common log format* (CLF) [3] defines a set of information a Web server must record when a user views a Web page. Table 1-1 displays the corresponding entries made into the Web server log file for the sample Web page displayed in Figure 1-1. The entries populating the Web server log files are popularly called *clickstream data*. CLF is not as detailed as clickstream data can be, but it is expressive enough to illustrate the kind of data that is recorded by the Web server. The following list describes each column recorded in CLF:

- The *Host* column records the address of the computer from where the HTTP request is made.
- The *Ident* column contains the remote logname of the requester only when the identd protocol is used, which it is not in this sample.
- The *Authuser* column contains the authenticated username of the requester only when an HTTP request is made over a secure connection to the Web server to identify the requester, which also is not used in this sample.

- The *Time* column contains the date and time the Web server receives an HTTP request.
- The *Request* column contains the first line of the HTTP request, which describes the type of request and what object is requested.
- The *Status* column contains a code that signifies whether or not the request is successfully serviced or what error occurred while servicing the request.
- The *Bytes* column contains the number of bytes sent to the requester for the object requested.

Host	Ident	Authuser	Time	Request	Status	Bytes
192.168.0.133	-	-	06/Jun/2001:20:12:32	"GET /scripts/w/dll HTTP/1.1"	200	6284
192.168.0.133	-	-	06/Jun/2001:20:12:33	"GET /images/7000/t6941.jpg HTTP/1.1"	200	5483
192.168.0.133	-	-	06/Jun/2001:20:12:33	"GET /images/2000/t1185.jpg HTTP/1.1"	200	5509
192.168.0.133	-	-	06/Jun/2001:20:12:34	"GET /images/9000/t8961.jpg HTTP/1.1"	200	5491
192.168.0.133	-	-	06/Jun/2001:20:12:34	"GET /images/10000/t9428.jpg HTTP/1.1"	200	5456
192.168.0.133	-	-	06/Jun/2001:20:12:34	"GET /images/2000/t1551.jpg HTTP/1.1"	200	5473
192.168.0.133	-	-	06/Jun/2001:20:12:36	"GET /images/best.gif HTTP/1.1"	200	1301
192.168.0.133	-	-	06/Jun/2001:20:12:36	"GET /images/tpclogo.gif HTTP/1.1"	200	8948
192.168.0.133	-	-	06/Jun/2001:20:12:36	"GET /images/new.gif HTTP/1.1"	200	1301
192.168.0.133	-	-	06/Jun/2001:20:12:36	"GET /images/search.gif HTTP/1.1"	200	4169
192.168.0.133	-	-	06/Jun/2001:20:12:38	"GET /images/cart.gif HTTP/1.1"	200	4244
192.168.0.133	-	-	06/Jun/2001:20:12:38	"GET /images/status.gif HTTP/1.1"	200	4192

Table 1-1: Sample Common Log Format Web Log

Additional information that enriches clickstream data, such as the Web page that led to the next request, and cookies, can also be recorded into the Web server log depending on the functionality provided by a Web server. Some of this data, especially cookies, must be generated by an application. If a Web server is not capable of recording information in addition to the data defined by CLF, an application can be developed to supplement the data in the Web server log.

Analyzing clickstream data helps companies understand the behaviors of their online customers. Web server log files detail every request a Web browser makes while a user navigates a Web site. The Web servers can record the site from which a visitor requests a Web page, when the page is requested, and which hyperlink led the visitor to a page. The behaviors that are exposed by clickstream analysis are described as searching,

seeking specific support questions, or other information gathering [4]. Ultimately, these behaviors tell us if a Web user is successful in finding information. This data can also be used to determine how a Web site should be laid out to help users find what they seek with a minimum number of clicks.

Behaviors such as ordering products and services can also be determined from analyzing clickstream data. A purchase obviously signifies that an order is successful, and analyzing clickstream data tells us if an order is incomplete or has been cancelled.

In each of these examples, it may be possible to determine the satisfaction of a Web user's visit. Whether a user is a happy, frustrated, or inclined to visit again can help a company build a stronger relationship with a customer.

1.1 Clickstream Analysis Architecture

Clickstream analysis is sometimes referred to as *Web usage mining*. An architecture for clickstream analysis has been defined by Cooley, Mobasher, and Srivastava [5]. Their architecture involves several steps and Figure 1-2 shows the processes involved. The actual analysis of user behaviors is done in the latter portion of the architecture. The steps leading up to the data mining phase of the diagram outline how clickstream data is prepared before it can be analyzed for user behaviors.

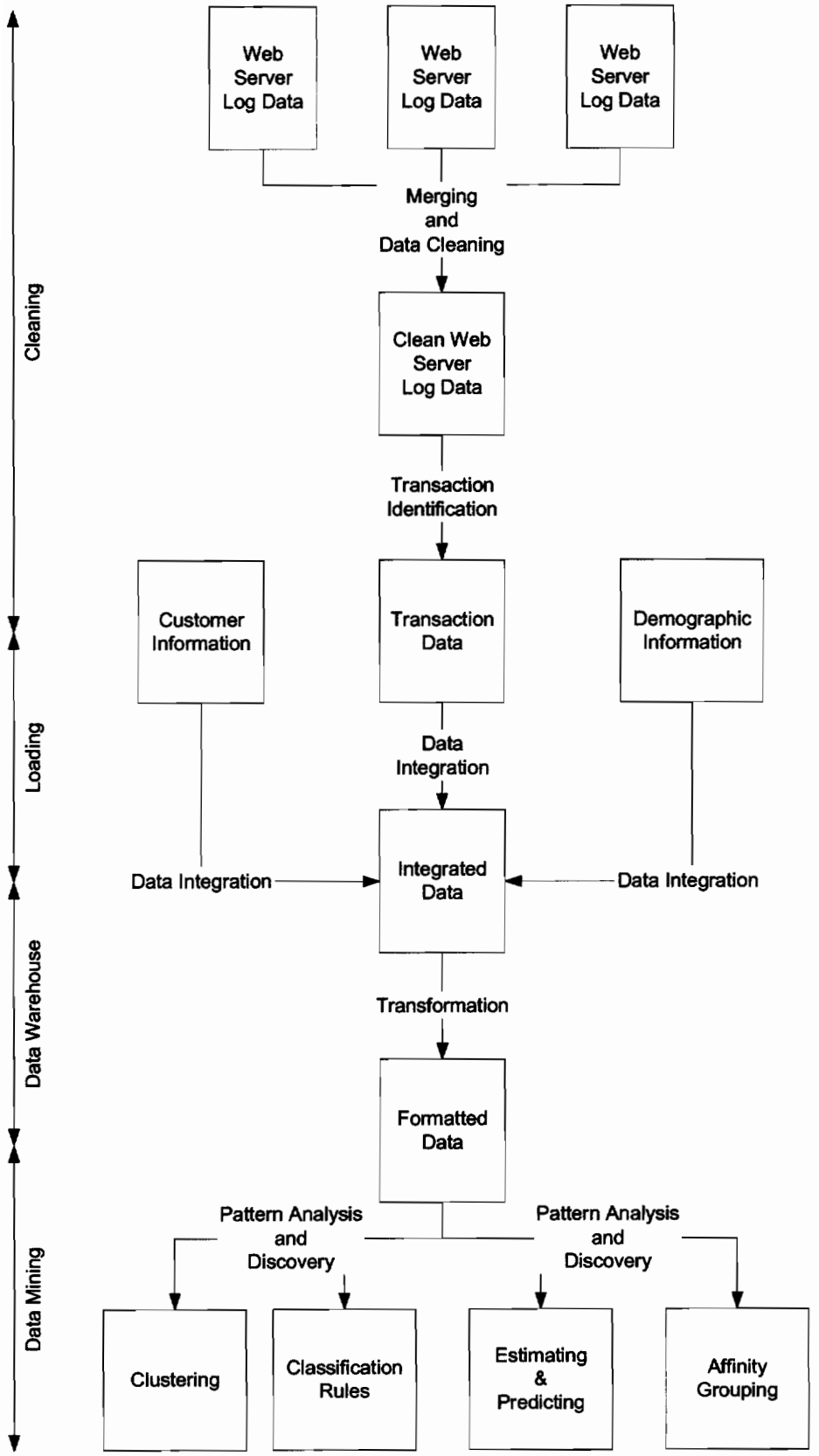

Figure 1-2: General Clickstream Analysis Architecture

Beginning in the cleaning phase, log files from every Web server and Web cache in a Web site are merged together to gather all of the clickstream data generated into a single location. Useless entries are filtered out at this stage since a large amount of the clickstream data generated does not help understand Web user behaviors. HTTP requests for HTML pages help determine user behaviors since this data is what tells us how a Web user is navigating through a Web site. Images are typically used for beautifying an HTML page or for displaying advertisements and do not indicate anything about how a Web user is navigating through a Web site. From the example shown in Figure 1-1 there can be ten times as many requests for images than Web pages and filtering out these requests dramatically reduces the amount of data to be dealt with.

In the clickstream loading phase, the cleaned clickstream data are grouped into transactions and loaded into a database or the clickstream data is loaded into a database first before being grouped into transactions. The goal is to cluster related HTTP requests together into meaningful classifications and prepare the clickstream data to be integrated into a data warehouse. Two typical classifications of a series of HTTP requests are a purchase transaction and a browsing transaction. A purchase transaction consists of a sequence of HTTP requests that leads a customer to ordering a product, and a browsing transaction is a sequence of HTTP requests that leads a customer to information regarding a product.

Once the clickstream data is loaded, it is ready to be imported into a data warehouse and integrated with other data sources. The method for integrating clickstream data varies and may be domain specific, meaning that parts of the integration process may vary based on how data are used by a particular company. For example, identifying users differs depending on how a customer is tracked. Users can be identified by inserting information as part of the hyperlinks, also known as *Uniform Resource Locators* (URL), generated on a Web page, or with the use of cookies [6]. An example of an independent data source would be demographic information about a Web user gathered from the user's *Internet protocol* (IP) address.

At this point, the clickstream data is loaded into a data warehouse and is ready for analysis. Various data mining tools can query the clickstream data once it is in the data

warehouse. These queries can answer questions relating to path analysis, association rules, sequential patterns, clusters and classification rules.

1.2 Clickstream Analysis

Clickstream analysis is performed with data mining. Data mining is categorized into four main activities, as shown in Figure 1-2: clustering, classifying, estimating and predicting, and affinity grouping.

Clustering involves grouping data into natural categories. It is a type of analysis that examines data and determines if the data clusters around certain points. For example, if a company's customer addresses are analyzed and a dot is printed on a map for each address, we can see if there are any natural categories relating to the location of the customers. So if a company is in the agriculture business, this map may display a high concentration of customers in the agricultural regions on the map.

Classifying involves assigning data into predefined categories. For example, an online bookstore may classify their customers by the type of books purchased. Categories such as non-fiction, fiction, science fiction, fantasy, and romance can be defined to group customers into.

Estimating and predicting are similar activities that involve calculating numerical results. For example, the same online bookstore may track the purchases of a customer who is a regular shopper. Analyzing the data collected can aid the bookstore in estimating how much money a customer spends a month and in predicting what items will be purchased each month.

Affinity grouping involves clustering data together that occur simultaneously. Using a similar example of an online bookstore tracking the purchases of a customer, the data collected can be analyzed to show that a customer who purchases the best selling science fiction book also purchases the best selling fantasy book.

1.3 Modeling a Real World Internet Commerce Site

An Internet commerce Web site usually consists of several components and the *Transaction Processing Performance Council* (TPC) has released a specification called *TPC Benchmark™ W* (TPC-W) [7] that represents the activities of a business oriented transactional Web server. TPC-W primarily models an online book reseller and Figure

1-3 outlines a general set of components used to support a transactional Web server. There are two primary locations where clickstream data is generated as shown in Figure 1-3. The Web servers record every request for an object and the reverse proxy server also generates clickstream data if it is configured as a Web cache.

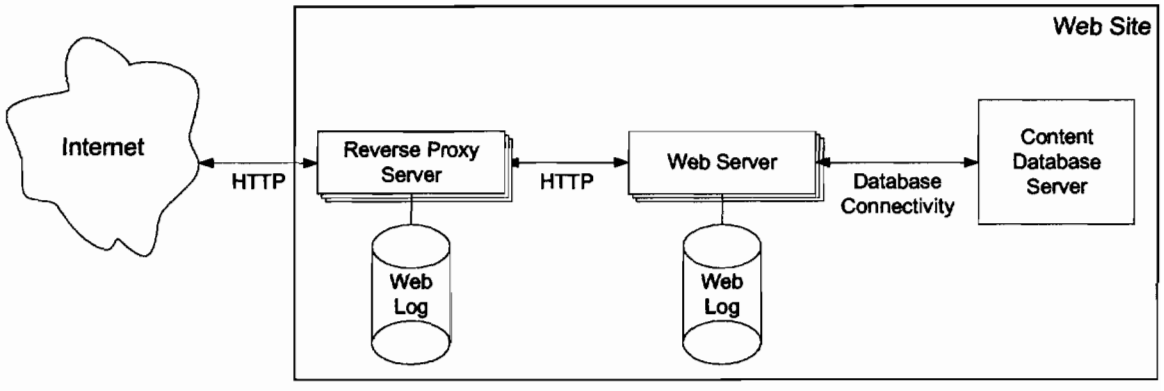

Figure 1-3: Basic Web Site Configuration

Reverse proxy servers intercept HTTP requests from Web browsers to the Web servers to accomplish a couple of tasks. By intercepting all HTTP requests to the Web servers, the reverse proxy servers can cache the Web objects requested. Reverse proxy servers are more commonly referred to as Web caches or Web server accelerators when they cache objects because they store information locally to relieve some of the workload of the Web servers. Using reverse proxy servers also provide security for a Web site, which is important for protecting online businesses.

The database server in the TPC-W benchmark, which will be referred to as a *content database*, provides information used to generate most of the Web pages. For example, inventory information stored in the content database is used to determine what the new products are and which products have been selling well. The content database also stores information about customers, such as billing and shipping information as well as information about their previous orders. Queries are made based on this information to suggest other books the customer may wish to consider. Basically, the content database is used to supply a Web page with information in addition to storing customer and inventory data.

While the TPC-W benchmark models a Web site supported by a content database, this thesis uses the benchmark to model traffic to a Web server. The content database discussed previously is a different database than the one used in this study. The database

in this study will be referred to as the *clickstream database* for clarity. Also, the results of this study are in no way comparable to any other TPC-W results nor can they used to project any kind of TPC-W results.

1.4 Challenges Collecting Clickstream Data

There are many factors that a company cannot control despite all its efforts to study its customers. First, the very nature of the Web does not make it easy to track customers. Customers must also be willing to be tracked by companies and even if they are, the infrastructure of the Web continues to make tracking difficult.

Web users are not always easily identifiable. The HTTP protocol is a stateless protocol meaning that information about the connection an HTTP is made over is not kept between requests. This also means that the identity of a Web user cannot be maintained between HTTP requests. HTTP/1.1 has improved the protocol's ability to maintain information between requests, but many challenges still remain.

The IP address of a Web user's computer is not a reliable means of identifying a user, because there is not always a one-to-one relationship between IP addresses and users. A Web user may use a computer at home, at work, in an Internet cafe, or on the road while traveling. In each scenario, each computer has a unique IP address, so a Web user may be associated with more than one IP address. Therefore, identifying individual users by an IP address could lead a company to believe it has an exaggerated number of users. Conversely, several members of a household may use the same computer so different Web users sharing the same computer may be identified by the same IP address leading a company to believe that fewer users are accessing its site.

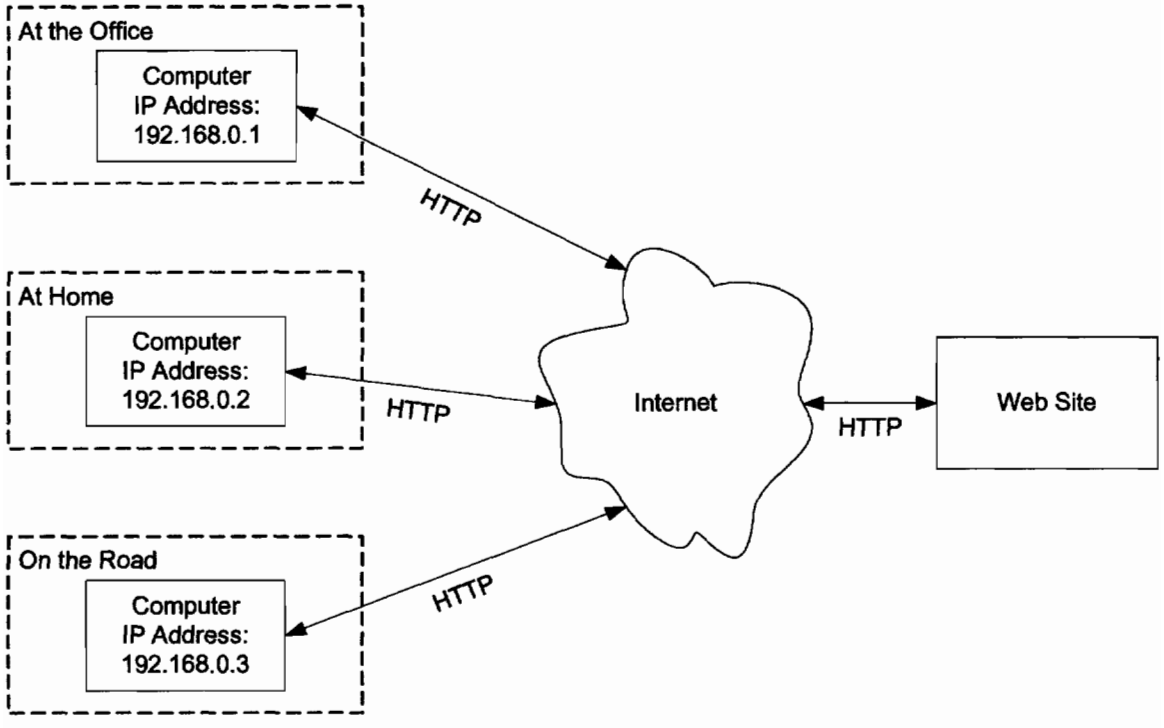

Figure 1-4: User Making Requests from Multiple Locations

Computers used at work or at home may be protected by a firewall used by the company or the Internet service provider (ISP). All of these computers access the Internet through a finite set of forward proxy servers. A forward proxy server, sometimes referred to as a proxy server, as shown in Figure 1-5, is similar to the reverse proxy server mentioned earlier. The only difference is that a forward proxy server handles requests originating from within a site to the Internet while a reverse proxy server handles requests coming in from the Internet into the site. Each HTTP request going through a forward proxy server contains the IP address of the proxy server. This adds further complications by associating multiple computers with the same IP address. Obviously, this situation can be even more confusing if multiple users share those computers.

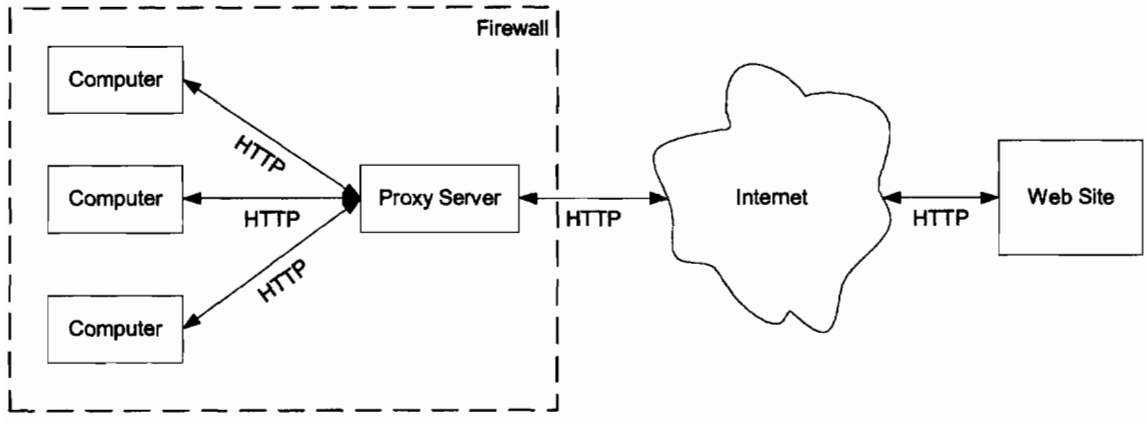

Figure 1-5: Firewall Masking Computer IP Addresses

It may still be difficult to collect data even if information about different users can be separated. Web caches, which include forward and reverse proxy caches, can also prevent a Web site from collecting clickstream data [8]. Furthermore, other intermediate Web caches between the Web user and the Web site may intercept and service a user's request. Web caches that are part of the Web site do not add to the problem since their Web log data are accessible.

Another challenge in collecting clickstream data derives from the importance of synchronizing the clock between all systems supporting a Web site. A Web site typically consists of multiple Web servers for functional or performance reasons, and various load balancing techniques can prevent a Web user from making requests to a single Web server. The first HTTP request may go to the first Web server, the second request may go to the second Web server, and so on. As a result, it may be impossible to determine the proper sequence of HTTP requests if the clock varies significantly between Web servers.

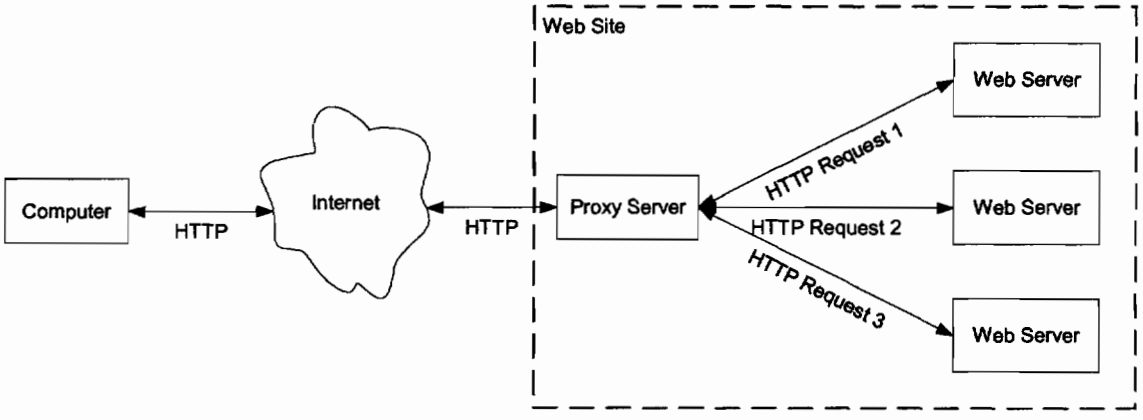

Figure 1-6: Multiple Web Servers Handling Requests from a Single User

URL links can be generated dynamically to contain information about a Web user as he navigates throughout the Web site, which is called *URL tagging.* Using identifying data in the URL creates possible performance problems because it typically renders Web caches useless, since caches index their data by the URL. For example, the Web page *cached_page.html* at the Web site www.somesite.com is requested and indexed without the use of URL tagging by the URL *http://www.somesite.com/cached_page.html.* Now let us use a key called *user_id* to tag the URL so that we can identify Web users, so one user is identified by a user_id of 1 and another with a user_id of 2. The URL to request the page cached_page.html is *http://www.somesite.com/cached_page.html?user_id=1* and *http://www.somesite.com/cached_page.html?user_id=2* for each user respectively. This results in duplicated information being cached for every user who requests this particular HTML page. Thus the workload increases on the Web servers since the Web cache is no longer helping service requests.

The use of cookies has been the most successful method of maintaining state information with a Web browser. However, the effectiveness of cookies depends on cooperation with Web users. Some Web users consider cookies an invasion of privacy, and Web browsers can be configured to ignore cookies sent by a Web server. Therefore, a Web site depending on the use of cookies to track its clients fails when the Web user does not accept cookies.

1.5 Current Practices

Clickstream analysis is a growing area for research and businesses with the explosion of electronic commerce on the Web. Section 2 describes previous work influencing this study that summarizes today's research and commercial products.

The previous research most similar to this study is done with a personal Web site and uses a database for simple Web site statistics. Although the article [12] summarizing this research is not explicitly clear, the work is likely done on a single system. The traffic on a personal Web site is very light compared to the workload emulated in this study, and the data stored in this database is not comprehensive enough to allow for analysis of Web user behaviors.

The current commercial products presume that Web server log files are archived before they enter the process for analysis of Web user behaviors. This cycle can occur on a weekly or even monthly basis and uses a multi-tiered environment. The tiers are typically the Web server systems, a system where Web server log files are archived, and the data warehouse system where the Web server log data are integrated.

All published benchmark configurations for the TPC-W use a multi-tier configuration to implement a Web site [9].[1] Each publication reports a benchmark configuration using separate systems running the Web servers, the Web caches, and a database sever for content. Some configurations have split the functionality of the Web servers further by using some Web servers to handle HTTP requests only for images.

1.6 Motivation

I have implemented a method of inserting clickstream data into a clickstream database from a Web server as it serves HTTP requests on a single, highly scalable system. The intent of this study is to determine if consolidating a Web server and a *clickstream database server*, which is a database server containing a clickstream database, onto a single system affects the performance of either component significantly.

---

[1] This information is current for TPC-W publications as of September 17, 2001.

This method may also allow clickstream analysis to begin as soon as the Web server generates clickstream data, to enable real-time analyses. This method addresses issues of:

- Current methods not allowing immediate clickstream analysis.
- A single system reducing the administrative costs in large-scale environments.
- A single system with the ability to allocate hardware resources dynamically between tasks.

Clickstream data in its raw form contains a wealth of information and must not be lost. Understanding Web users' behaviors leads to improved online interactions with customers, which in turn leads to increased loyalty, revenues, and profits. This customer data is precious and should be stored in a medium that is durable, easy to query, and easy to gather information from. A database is an excellent place to store clickstream data especially if the database also contains a data warehouse. Clickstream data adds many new data sources about customer behavior to the data warehouse. A data warehouse containing clickstream data is sometimes called a data webhouse or webhouse [10].

Enterprises expanding from their brick-and-mortar roots into Internet commerce are candidates for consolidating Web services and data warehouses when they provide their own Web services in a central information technology (IT) organization [11]. These companies tend to shop in the high-end segment of the server market where they purchase systems with enough resources to sustain peak loads.

There are several advantages for combining Web servers and database servers on a single system. Having a single system reduces administrative tasks and related costs. Some Web sites use dozens of computers to support their operations. Each of those systems requires networking and other administrative tasks to be operational. Total software costs, licensing, and maintenance fees can also increase with the number of systems used.

Resources can be dynamically allocated between applications on a single system since they share the same hardware components. For example, if a Web server needs more processing power, work can be directed to another central processing unit (CPU) in a multiprocessor system.

A highly scalable system also allows upgrades to the system to meet the demands of the future. More CPU's, memory, and storage can be added to the system to sustain a growing business. While future demands can also be met by purchasing additional systems, this thesis focuses on working towards consolidation.

1.7 Hypotheses

The hypotheses of this thesis are:

- Consolidating a Web server and a clickstream database server onto a single system will demonstrate that the clickstream database server running on a separate system will not require more resources when running on a system with Web servers and vice versa.
- There will not be any significant difficulties when running Web services and a clickstream database server on the same system.

The Web server and the clickstream database server are likely to use different resources. I expect the Web server to primarily use CPU resources since its main task is to generate Web pages and I expect the clickstream database server to primarily use hard disk storage resources since its main task is to store clickstream data.

Since these two components are likely to depend on different resources, they should be configurable such that they do not compete for the same resources or execute any significantly conflicting tasks.

1.8 Thesis Overview

The next section, Section 2, summarizes previous work related to moving clickstream data into a database and other phases in the data mining architecture. Next, in Section 3, I describe how I implement my experiments followed by test cases in Section 4. Then the results of the test cases are presented in Section 5 with an analysis of the data. The thesis ends with a conclusion in Section 6 and suggestions for future work in Section 7.

2 Previous Work

Clickstream analysis has been intensively researched, but there has not been much work related to consolidating a Web server and a database server on a single system for clickstream analysis. This idea has not been researched as a single concept, but in parts, as researchers have worked with loading clickstream data into a database, both from a Web server and from a repository. Other researchers have worked on planning for consolidating data center resources for cost savings. While data centers are not directly related to this thesis, the reasons for the consolidating systems are.

Stein [12] has experimented with redirecting Web server log entries into a relational database for simple analyses. Web log data can be analyzed in its raw form to gather simple statistics about a Web site, and Stein uses Perl to query for these statistics from the relational database. These simple statistics usually reveal how many hits a site receives in a day, how much data is transmitted over the network, when the busiest time of day is, and what the most popular pages are. This experimentation does not make use of a data warehouse and does not perform any analyses to determine Web-user behavior. Stein's primary interest is using Perl as a tool to develop reports detailing these simple statistics.

There is at least one commercial product that takes data, including Web log data, from a repository and loads it into a data warehouse. Torrent, a privately owned company, has a product called *Orchestrate* that performs this task, but it does not load clickstream data as a Web server services HTTP requests. A white paper describing the architecture of Orchestrate [13] shows the product loading data into a data warehouse from a repository external to the data warehouse system.

Warkow [14] discusses the cost considerations when planning for data center consolidation. Reducing the total number of systems, software packages, and support personnel generally reduces the costs required to maintain those systems. Although Warkow focuses on data centers, the same principles can be applied to Web sites.

3 Implementation

This section describes the applications developed, database schema, and the hardware layout of the systems used in this study. The applications consist of the Web application that generates the Web pages on the site and the application that emulates the workload of Web users.

The application for the online storefront and the application emulating the Web user activities are based on the TPC-W specification. The storefront consists of fourteen different Web pages. Figure 3-1 below displays each Web page and how a user navigates between them. The lines with the arrows indicate the direction a Web user travels to get from one Web page to the next. A line with arrows at each end indicates that a Web user can travel in either direction. The fourteen Web pages and their purpose are:

- Admin Request - Request an update for an item.
- Admin Results - Confirm an update for an item.
- Best Sellers - List the best-selling items.
- Buy Confirm - Confirm an order.
- Buy Request - Request an order.
- Customer Registration - User sign in.
- Home - Home page.
- New Products - List the newest items.
- Order Display - Display the most recent order.
- Order Inquiry - Request the most recent order.
- Product Detail - Item information.
- Search Request - Item search form.
- Search Result - Results for an item search.
- Shopping Cart - Current items in the shopping cart.

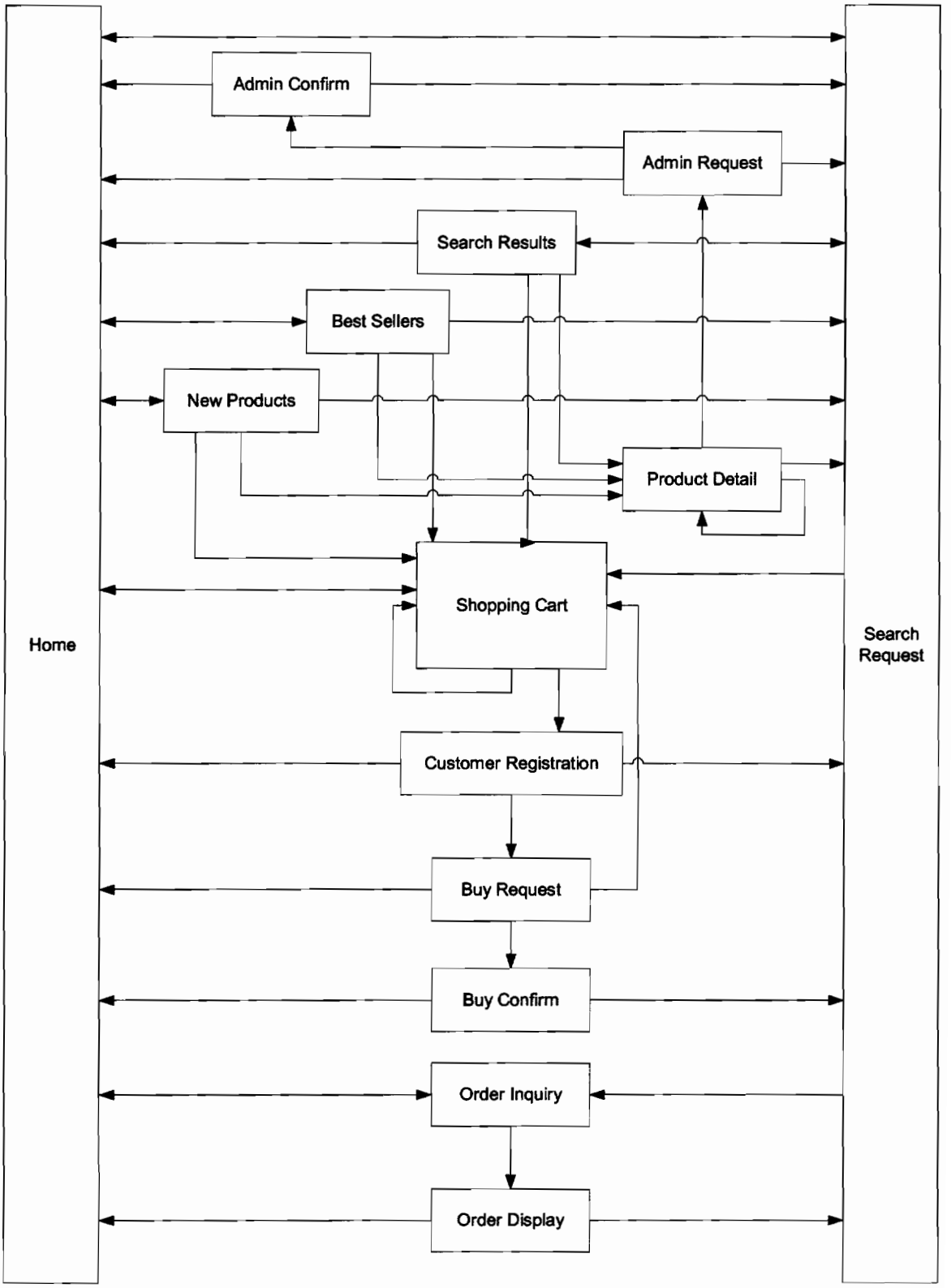

Figure 3-1: TPC-W Web Site Flow Chart

Error! Reference source not found. displays the hardware and software components used to demonstrate the consolidation of a Web server and a clickstream database server. The dotted lines indicate the physical systems and the solid lines indicate the software components on each system. The lines between each software component indicate communication used between those two components.

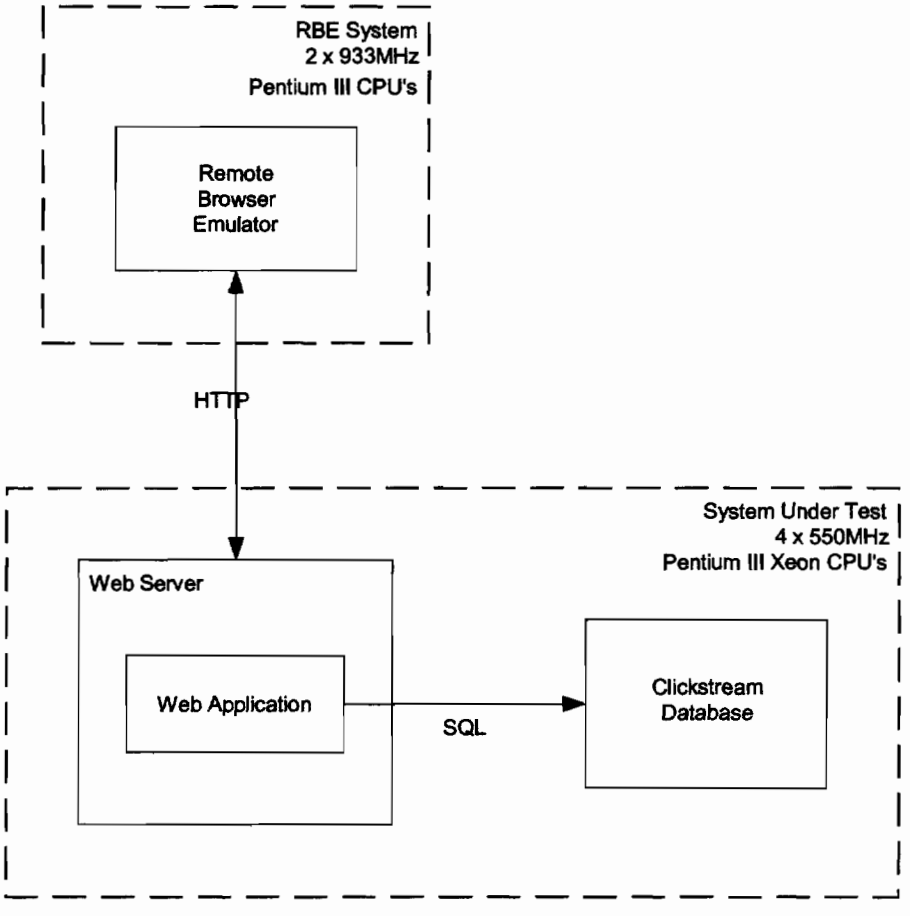

Figure 3-2: Hardware and Software Component Diagram

3.1 Remote Browser Emulator

The *remote browser emulator* (RBE) system has two 933MHz Pentium III processors with 256KB of level 2 (L2) cache and 512MB of random access memory (RAM). The RBE software is a homegrown application that simulates the activities of Web users based on the guidelines given in the TPC-W specification. The RBE system is connected to the system under test through a gigabit Ethernet adapter card.

The RBE requests Web pages following the guidelines from the TPC-W specification for a mix of requests. The RBE simulates *tracked visitors* [15], users who

are reliably identifiable across multiple visits, to generate the richest clickstream data possible.

Only four connections are opened to the Web server to simulate approximately 7,000 Web users that generate a steady workload on the four processors in the test system. To ensure a consistent load on the test system's processors, HTTP requests are executed immediately after one another.

3.2 System Under Test

The system under test contains two main components, the Web server and the clickstream database server. The system has four 550MHz Pentium III Xeon processors with 2MB L2 cache and 4GB of RAM, and is connected to an array of 9.1GB 10,000 RPM hard disk drives with 512KB cache. The number of disks allocated for each component is detailed in the following sub-subsections.

3.2.1 Web Server

The Web application is a multi-threaded program that uses a pool of threads, called the *Web pool*, to generate HTML pages. An additional pool of threads, called the *database client pool*, is used to insert the clickstream data into the clickstream database. Threads are used to keep the various tasks in the Web application asynchronous. Figure 3-3 displays a flow chart detailing the pools of threads and queues used in the application. The Web pool threads generate the HTML pages and queue up HTTP log entries to be inserted into the clickstream database. The database client pool threads take the HTTP log entries off the queue and insert sets of rows into the clickstream database.

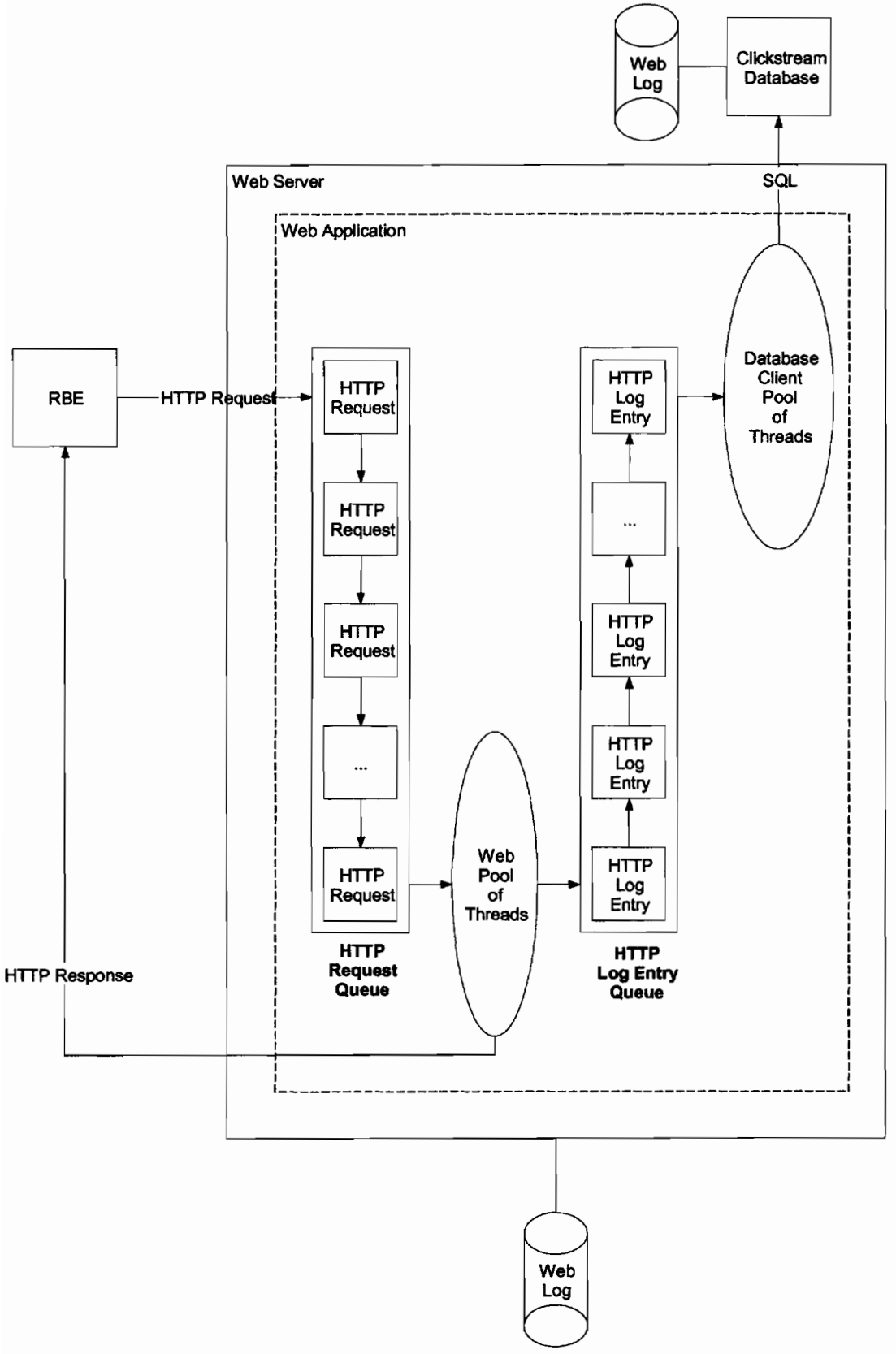

Figure 3-3: Web Application Flow Chart

The Web server uses ten hard disk drives in a redundant array of independent disks (RAID) Level 5 configuration to store the Web server log files. The Web server log is essentially written to two separate places, which are the Web server log and the clickstream database, to provide durability in case of a system failure and to prevent valuable information from being lost.

3.2.2 Clickstream Database

A commercial database management system, IBM DB2 Universal Database, is used to store the clickstream data. The clickstream database is built on a total of twenty-one hard disk drives in a RAID Level 5 configuration to make the data durable in case of a disk failure. Ten hard disk drives are used for the user tables, three are used for the system tables, three are used for temporary space, and five are used for the database log files. Table 3-1 describes the database table used to store the clickstream data in the clickstream database. This table captures the same information as the CLF in addition to all the cookies used by the Web application to track a Web user.

Column	Datatype	Description
HOST_ADDR	Variable text, size 16	Internet protocol (IP) address.
HOST	Variable text, size 64	Domain name of IP address.
IDENT	Variable text, size 32	HTTP user identifier.
AUTHUSER	Variable text, size 32	HTTP user authorization.
REQTIME	Date and time	Time request is received by the Web server.
REQUEST	Variable text, size 512	The first line of the HTTP request.
STATUS	Numeric, 5 digits	The status code of the HTTP request.
BYTES	Numeric, 19 digits	The amount of data, in bytes, returned to the requester.
C_ID	Numeric, 19 digits	A unique identifier for the user.
S_ID	Numeric, 19 digits	A unique session id for the user.
SC_ID	Numeric, 19 digits	A unique shopping cart id for the user.

Table 3-1: HTTP_LOG Table Description

3.3 Simplifications

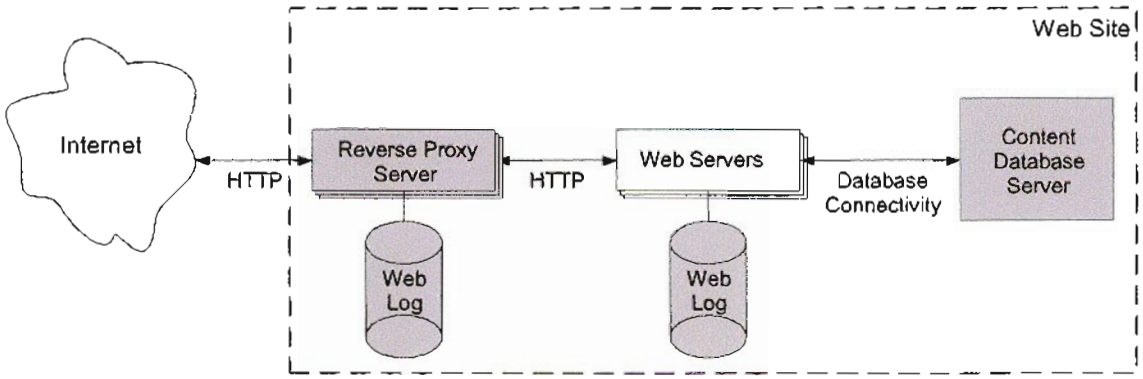

Figure 3-4: Simplified Web Site Configuration

TPC-W is a complex benchmark and I have made several assumptions to simplify the benchmark to make it easier to run so I can focus on generating clickstream data and loading it into a database. Figure 3-4 displays the same general Web site configuration shown previously in Figure 1-3 with the components removed in gray. Each of these simplifications omits components of the TPC-W benchmark so that I can focus on testing the component in the system that generates clickstream data. Some of these components are required to comply with the TPC-W specification but none of these changes affect the amount of clickstream data generated. These changes only affect the Web server directly and, in specific situations, would slightly increase the amount of processing required by the Web server. Sub-subsection 3.3.2 illustrates an example where Web server processing would be increased.

3.3.1 Use of Web Caches

The uses and effects of Web caches are ignored in this study to reduce the complexity of loading clickstream data into a database from multiple sources. This includes Web caches internal and external to the site to allow the Web servers to receive and record all HTTP requests for content on the site. Eliminating Web caches external to the site allows the clickstream data in the Web server logs to remain rich, and eliminating Web caches internal to the site only saves the additional work of merging the Web log data from the Web cache with the Web log data from the Web server.

A technique called *cache busting* can be used to render Web caches useless even if they are used in this study. The HTTP/1.1 protocol allows information in a response to contain caching directives. These directives can specify how long a Web object may be cached, so a Web cache can be directed to never cache specific objects. Unfortunately, a Web cache must be fully compliant with the HTTP specification in order for this method to be effective.

3.3.2 Use of Images

There are no requests for images from the Web server in this study. The clickstream data generated from HTTP requests for images do no contribute to understanding customer behaviors and are generally ignored for clickstream analysis.

The majority of the site configurations used in TPC-W publications [9] split the functionality of the Web server into two components. One component is still named the Web server, which I will call the *application server* in this sub-subsection to avoid confusion with the rest of this thesis, while the other component is named the *image server*. The application server continues to handle requests for HTML pages and the smaller images representing buttons and banners, while the image server handles requests for all of the larger images representing pictures of book covers.

If another Web server was configured into the system to handle HTTP requests for images, the application server would have required less processing time although the overall performance of the system would increase. The reason that less processing time is required by the application server is because the number of HTTP requests handled per second decreases when images are also requested. For example, the following steps briefly describe the actions taken to request an HTML page complete with images:

1. Issue an HTTP request for an HTML page.
2. Receive the HTML page.
3. Parse the HTML page for images to retrieve.
4. Issue an HTTP requests for each image and receive each image.

Time is saved when images are not requested since there are fewer steps to perform, namely the steps to retrieve the images on an HTML page. The resulting effect is that more HTTP requests can be serviced over time, thus the application server has more work to do.

There would have also been additional data sent across the network, but the additional traffic would not have caused a bottleneck on the gigabit Ethernet network used in this study. Since the clickstream data generated from the image requests are typically filtered out and not inserted into the database, the absence of image requests has no direct impact on the quality of the data in the database.

3.3.3 Use of Static Content

TPC-W specifies that some of the Web pages generated in the benchmark be created with data from a content database. For example, the Web page listing new books queries the content database to determine the latest fifty products added to the company's inventory. The TPC-W specification outlines instructions for creating a content database that supplies this information. Out of the fourteen Web pages defined in the benchmark, thirteen query a database for content.

Implementing this content database requires additional management and tuning, and additional hardware to support the database. The custom Web application that I have implemented generates static content so that I do not have to worry about the complexities with implementing a content database. It is likely that this content database can be implemented without significantly affecting other components on the system, but since this is a complex component, I will leave the verification work for future study.

While removing the content database is a significant change to the TPC-W benchmark, the Web server experiences an increased load for similar reasons given in the previous sub-subsection for image requests. Since the Web server does not have to spend additional time retrieving information from the content database, more HTTP requests can be serviced over a given period of time.

4 Performance Tests

This section describes the four tests and the metrics used in this study to measure the change in performance of the Web server and the clickstream database server between each test. The ultimate goal of the tests outlined in this section is to determine if performance suffers when a Web server and a clickstream database server are consolidated onto a single system.

Each test is described in greater detail in the following subsections, but the following briefly tells the purpose of each test. The first test examines the performance capabilities of the Web server without using a database to store clickstream data. The second test examines at the performance of the Web server and the database used to store clickstream data when they are implemented on two separate systems. This test allows us to identify performance changes from using a clickstream database on a separate system compared to consolidating the database server with the Web server. The third test experiments with implementing the Web server and the same clickstream database on a single system with the Web server logging redundant clickstream data. Finally, the fourth test is a variation on the third test where the Web server does not redundantly log clickstream information.

4.1 Test 1: Standalone Web Server

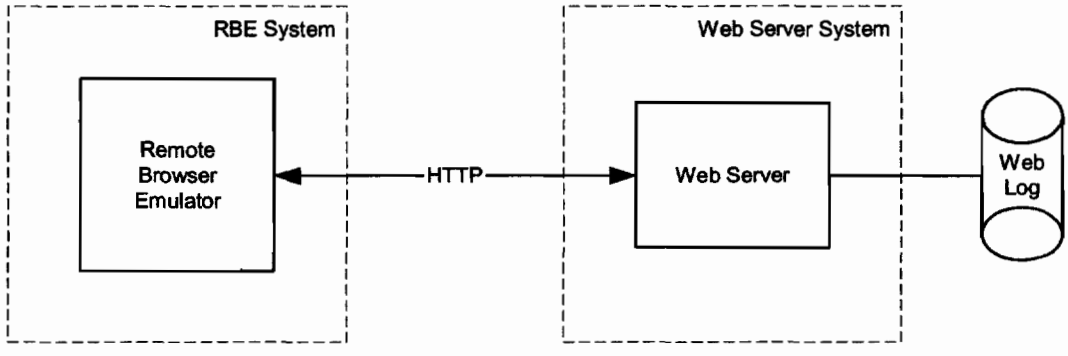

Figure 4-1: Test 1 System Configuration

Test 1 demonstrates the performance of the Web server when it runs without the use of a database server to store clickstream data. The results of this test are used as a baseline for the results of the following tests. The configuration of the systems is shown in Figure 4-1. Thus, this test should produce the highest number of HTTP requests serviced per second.

4.2 Test 2: Web Server and Database Server in a Multi-Tiered Environment

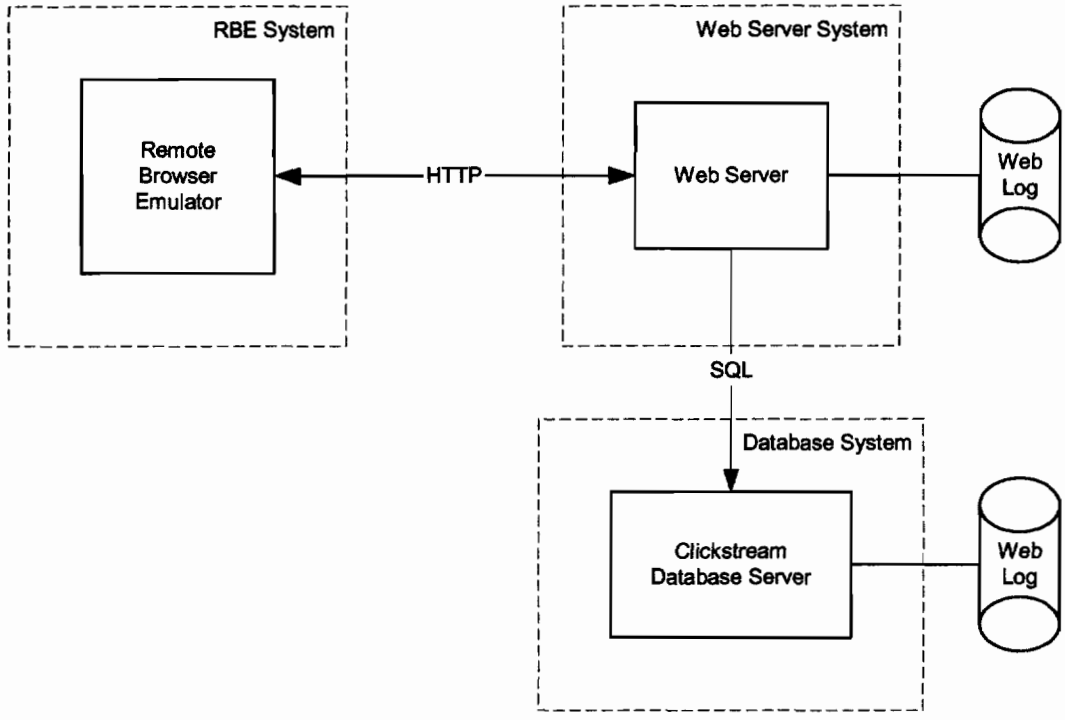

Figure 4-2: Test 2 System Configuration

Test 2 demonstrates the performance impact of adding a clickstream database server into the Web site configuration. The configuration of the system is shown in Figure 4-2 and models an idea similar to the multi-tier environment shown in the Orchestrate white paper [13]. The Web server and the clickstream database server are not consolidated onto a single system for this test so that the impact of using the clickstream database server can be measured independently from any effects from consolidating these two components. The Web server continues to log HTTP requests into its own log file to provide durability in case of a system failure that may result in the loss of clickstream data currently being inserted into the clickstream database. The

redundant logging done by the Web server is expected to affect the number of HTTP requests per second and the hard disk drive activity from the Web server should be affected accordingly.

4.3 Test 3: Consolidated Web Server and Database Server

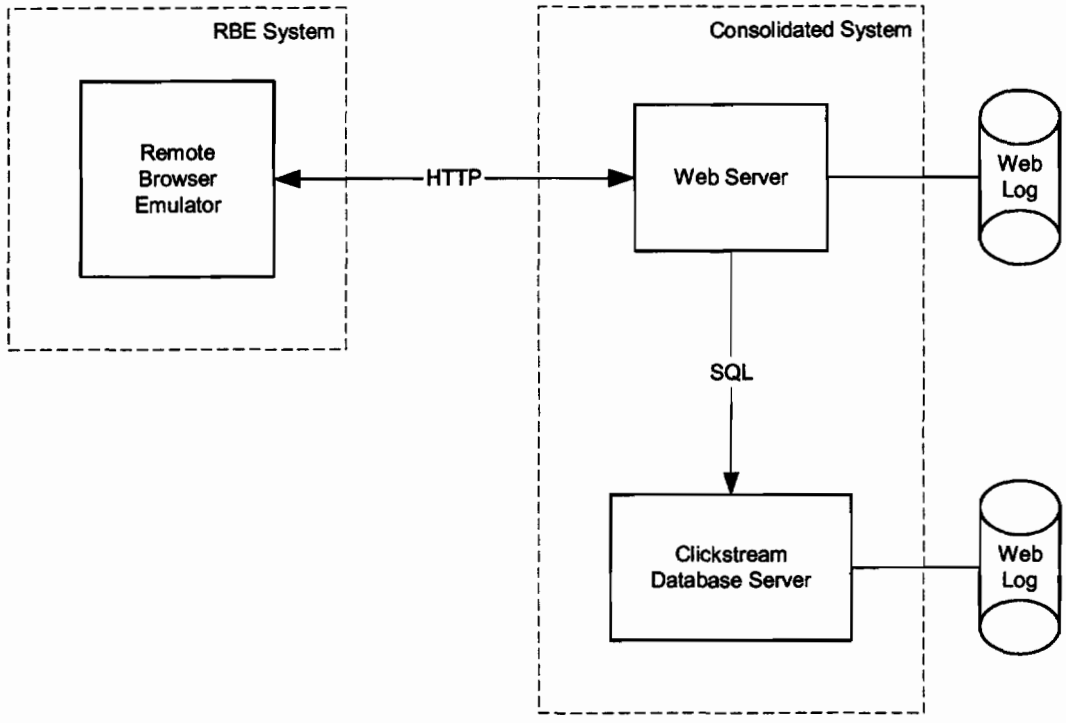

Figure 4-3: Test 3 System Configuration

Test 3 demonstrates the effect of consolidating the Web server and the clickstream database server onto a single system. Any performance differences between this test and the previous test show the impact that consolidating has on a Web server and a clickstream database server compared to using each component on a separate system. The configuration of the system is shown in Figure 4-3. The Web server still continues to log HTTP requests into its own log file to provide durability in case of a system failure. The results of this test are not expected to be significantly different from those in Test 2. The previous tests should show that the Web server and clickstream database server are not competing for the same resources so each component should be able to run concurrently without any serious performance impact.

4.4 Test 4: Consolidated System without Web Server Logging

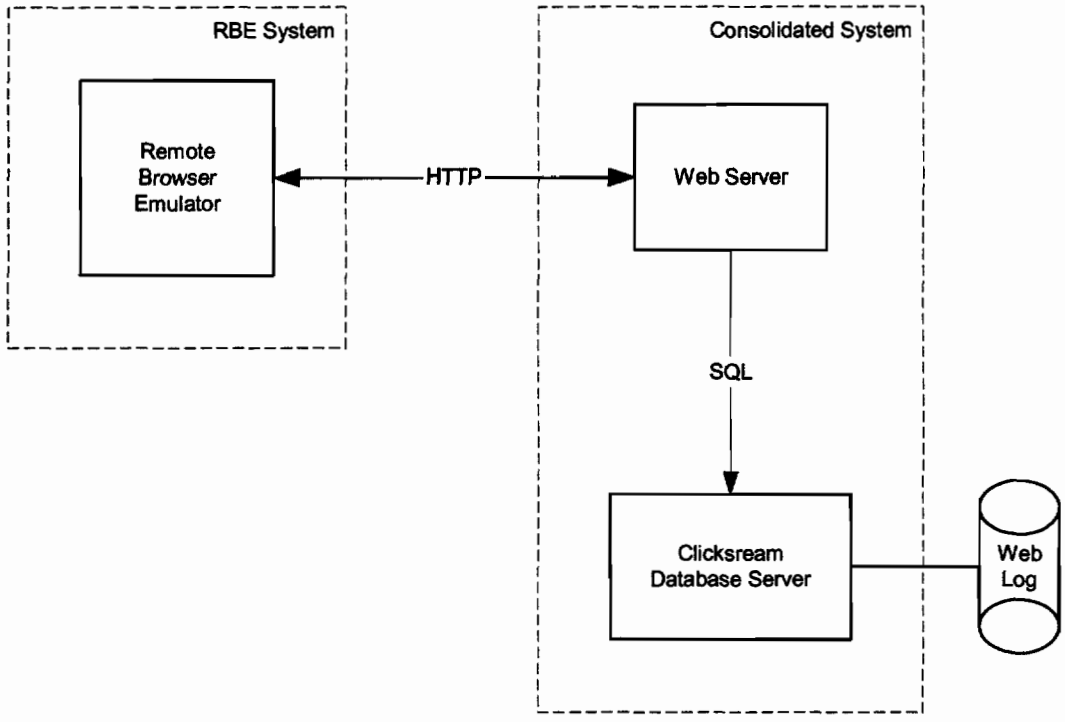

Figure 4-4: Test 4 System Configuration

Test 4 is similar to Test 3 except the Web server does not redundantly log HTTP requests, as shown in Figure 4-4. The purpose of this test is to determine if any load can be removed from the Web server while letting the clickstream database handle all of the logging. The results of this test are expected to be very similar to the results of Test 3 since the changes in the system configuration are not significant. The performance of the clickstream database server should not change from Test 3, and the resource utilization of the Web server should decrease since it is doing less work than in the previous test.

4.5 Metrics

Three main metrics are used to analyze the performance in each of these tests:

- The number of HTTP requests serviced per second.
- The individual processor utilization by the Web server and the clickstream database server.
- The amount of hard disk activity from writing to the Web server log, the clickstream database, and the database log.

The number of HTTP requests serviced per second measures the throughput of the entire system. While I expect the processor utilization and hard disk drive activities to be the resources primarily used in this study, the impact of any changes in these two metrics are measured by the corresponding change in the number of HTTP requests serviced per second.

The processor utilization of the Web server and the clickstream database server measures how much processing is required to achieve the measured throughput. It is also used to gauge each component's requirements for processing resources.

The Web server and the clickstream database server are the two components writing the most data to the hard disk drives. The Web server needs to write each HTTP request to its log file, and the database server is writing clickstream data into a table while making the corresponding entries into the database log. The activity for reading data from the hard disk drives is ignored since the Web server and the clickstream database server do not perform any functions that require reading data in this study.

Five additional metrics are examined to verify the findings of the previous three metrics:

- The average response time to service an HTTP request.
- The amount of data sent over the network.
- The Web log growth rate.
- The clickstream database growth rate.
- The available system memory on the Web server system.

These metrics also aid in determining if any other resources are bottlenecks in this study. The average response time is examined to determine the effect each test has on the responsiveness of the Web server. The amount of data transmitted over the network is watched to be sure that the throughput of the entire system is not affected by a network bottleneck. The Web log growth and the clickstream database growth are also calculated to be sure that there is plenty of hard disk drive space for each component. The available system memory indicates that the Web server is able to service HTTP requests and insert data into the clickstream database at the measured throughput.

5 Test Results and Analysis

The graphs presented in this section show the change in performance for each metric. Each test ran for approximately one hour with data sampled every thirty seconds. Metrics pertaining primarily to the Web server are displayed as changes from Test 1 to Test 2, from Test 1 to Test 3, and from Test 1 to Test 4. The changes in performance from Test 1 to Test 2 show the effect of using a database for clickstream analysis. Performance changes from Test 1 to Test 3 show if consolidating the Web server and the clickstream database server onto the same system causes any additional performance degradation with redundantly logging clickstream data by the Web server. Similarly, performance changes from Test 1 to Test 4 show if consolidating the Web server and the clickstream database server onto the same system causes any performance degradation without the Web server redundantly logging clickstream data. Charts of the raw data used to calculate the performance differences shown in this section are displayed in Appendix A with some of the secondary metrics described in the previous section. Similarly, metrics pertaining primarily to the clickstream database server are displayed as changes from Test 2 to Test 3, and from Test 2 to Test 4 since the clickstream database is not used in Test 1.

5.1 HTTP Get Requests per Second

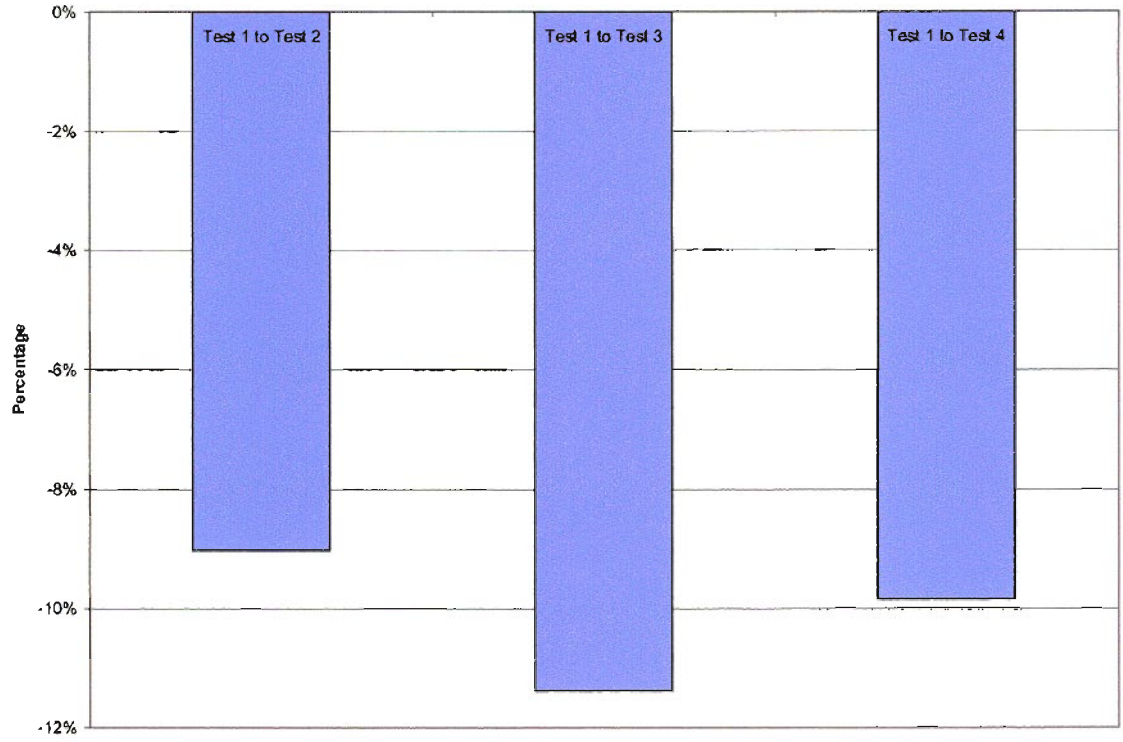

Figure 5-1: Change in HTTP Requests per Second

Figure 5-1 displays the percentage difference in the number of HTTP requests serviced per second between Test 1 and Test 2, between Test 1 and Test 3, and between Test 1 and Test 4. Using a clickstream database server reduces the throughput by approximately 9%. Consolidating the Web server while redundantly logging clickstream data and the clickstream database server reduces the throughput from Test 1 by approximately 11%. The throughput decreased only 2% between Test 2 and Test 3.

The 9% reduction in overall throughput from implementing a clickstream database to store clickstream data initially appears to contradict the hypothesis that storing clickstream data in a database will not significantly impact performance. However, the response time data for the time taken to service HTTP requests, shown in Table 5-1, suggest otherwise. Table 5-1 displays the average time taken to receive a Web page after sending an HTTP request measured by the RBE. The average response time calculated in Test 2 is 12% higher than the average response time calculated in Test 1, which is close to the change in the HTTP requests serviced per second between Test 1

and Test 2. Likewise, the average response time calculated for Test 3 is 16% higher than the average response time calculated in Test 1, which is close to the change in the HTTP requests serviced per second between Test 1 and Test 3. Even though there is a 9% difference in the number of Web pages the Web server is generating per second in Test 1, the average response time calculated in each test does not vary more than one millisecond, which is hardly noticeable to a user. The difference in the average response time is likely due to the extra task of inserting clickstream data into the clickstream database. Also, the 4% difference between Test 2 and Test 3 supports the hypothesis that the Web server and the clickstream database server do not compete for the same resources when both components are consolidated onto a single system.

	Average (Seconds)
Test 1	0.00380
Test 2	0.00426
Test 3	0.00439
Test 4	0.00430

Table 5-1: Average Web Page Response Times

5.2 Processor Utilization

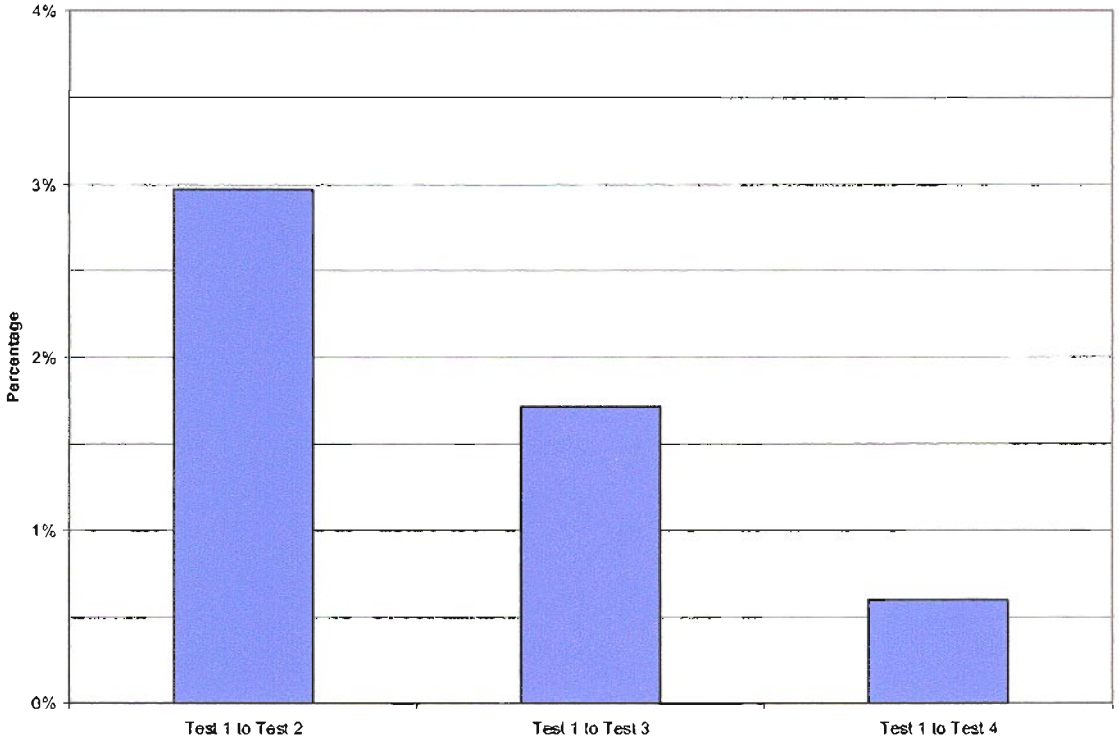

Figure 5-2: Change in Web Server Processor Utilization

Figure 5-2 displays the percentage difference in the processor utilization by the Web server between Test 1 and Test 2, between Test 1 and Test 3, and between Test 1 and Test 4. The use of the clickstream database server increases the processor utilization of the Web server by 3%. Consolidating the Web server while redundantly logging clickstream data and the clickstream database server increases the processor utilization by almost 2%, and increases the processor utilization about 0.5% without the Web server redundantly logging clickstream data. The processor utilization decreased about 2% between Test 2 and Test 3, and decreased about 1% between Test 3 and Test 4.

The processor utilization of the Web server is not significantly affected by implementing a clickstream database server or by consolidating the Web server with a clickstream database server on a single system. Obviously, inserting records into a clickstream database requires additional processing by the Web server. However, it does not appear to require a significant amount of additional work since the processor utilization decreased in Test 3. Thus this data supports the hypothesis that the Web

server and the clickstream database server do not compete for the same resources when both components are consolidated onto a single system.

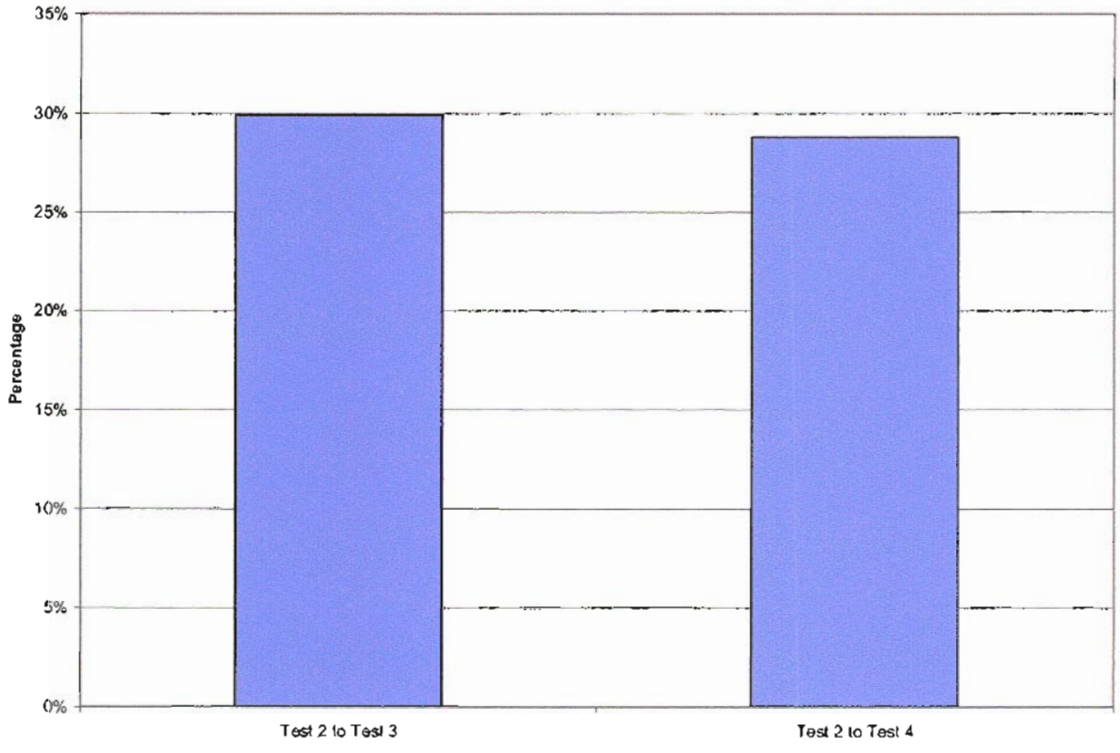

Figure 5-3: Change in Clickstream Database Server Processor Utilization

Figure 5-3 displays the percentage difference in the processor utilization by the clickstream database server between Test 2 and Test 3, and between Test 2 and Test 4. Consolidating the Web server while redundantly logging clickstream data and the clickstream database server increases the processor utilization by 30%, and increases the processor utilization 29% without the Web server redundantly logging clickstream data. While these percentages are a factor of ten higher than the changes in processor utilization, shown previously for the Web server, the absolute change in processor utilization for the clickstream database server is only 0.5%, which is an insignificant change supporting the hypothesis that the Web server and the clickstream database server do not compete for the same resources when both components are consolidated onto a single system.

5.3 Hard Disk Drive Activity

The hard disk drive metrics are presented in the following sub-subsections for each software component. The metrics are bytes per write, bytes written per second, and writes per second. The bytes written per second could be measured without the other two metrics since it is just the multiple of the number of bytes per write and the number of writes per second, but if there is a performance issue with the hard disk drives, measuring all three metrics allows us to determine the best course of action to resolve the issue. The data is only collected for the clickstream database server in Test 2, Test 3, and Test 4 since it is not used in Test 1.

5.3.1 Web Server Log

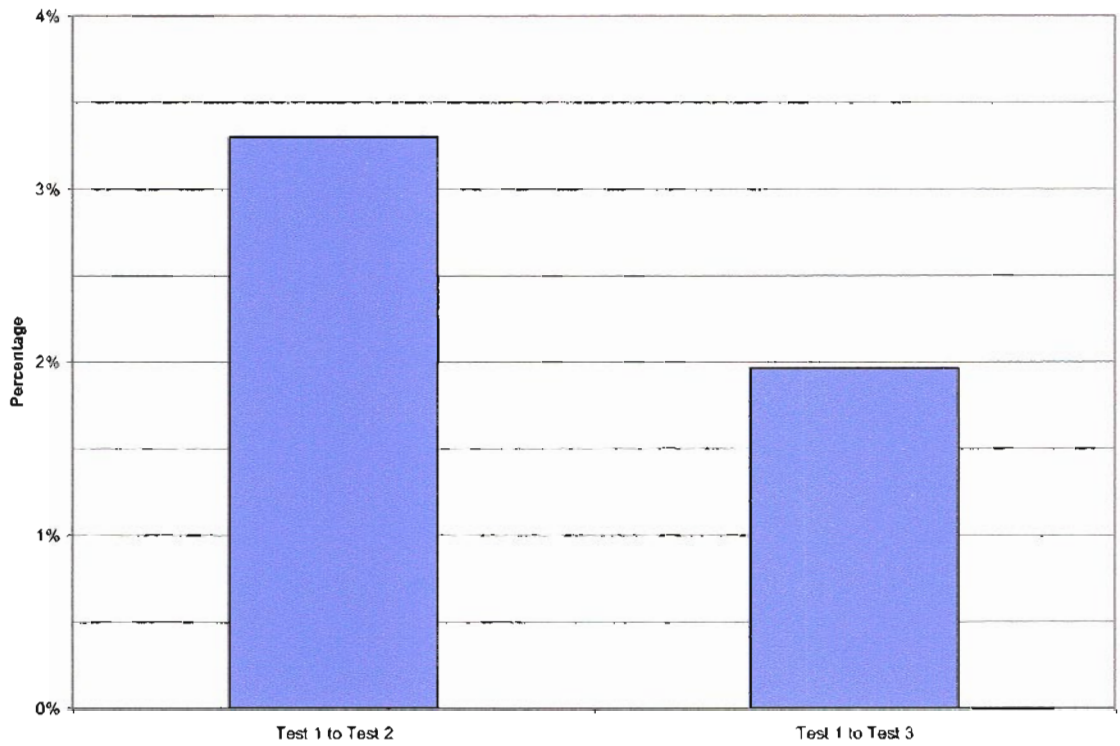

Figure 5-4: Change in Web Server Log Bytes per Write

Figure 5-4 displays the percentage difference in the number of bytes per write to the Web server log between Test 1 and Test 2, and between Test 1 and Test 3. The use of the clickstream database server increases the number of bytes per write by almost 3.5%. Consolidating the Web server while redundantly logging clickstream data and the

clickstream database server increases the number of bytes per write by almost 2.0%. The number of bytes per write decreases approximately 1.5% between Test 2 and Test 3.

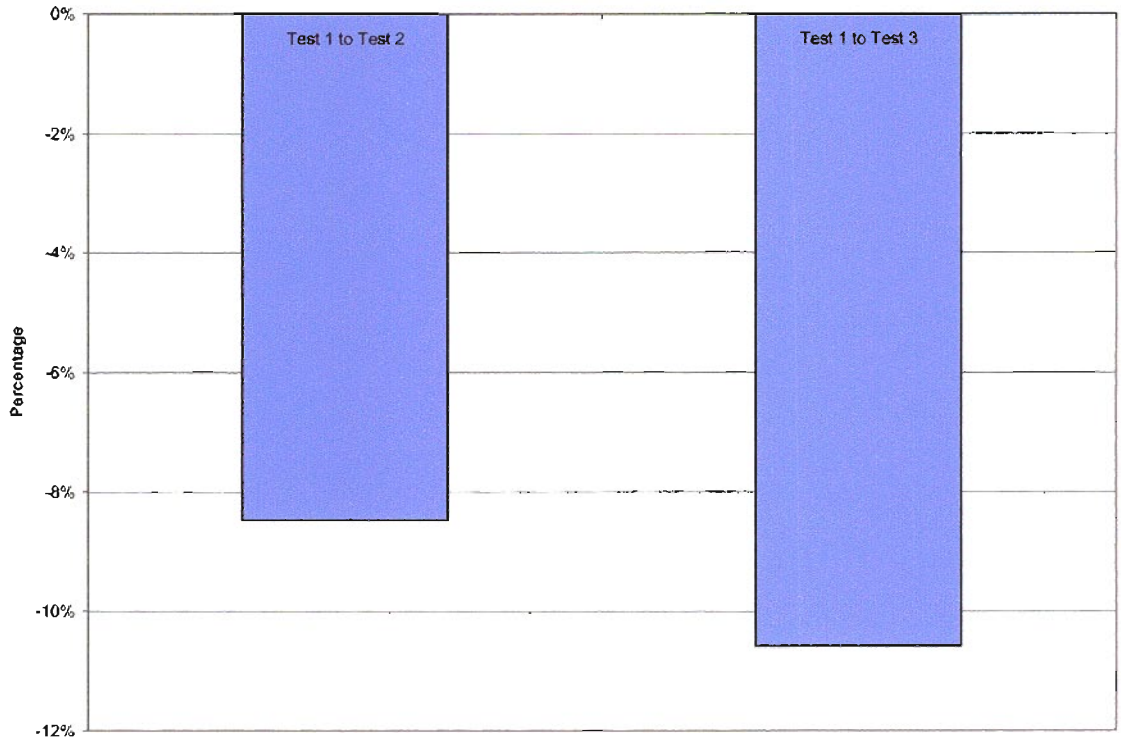

Figure 5-5: Change in Web Server Log Bytes Written per Second

Figure 5-5 displays the percentage difference in the number of bytes written to the Web server log between Test 1 and Test 2, and between Test 1 and Test 3. The addition of the clickstream database server reduces the number of bytes written per second to the Web server log by about 8%. Consolidating the Web server while redundantly logging clickstream data and the clickstream database server reduces the number of bytes written per second to the log file by about 10%. The number of bytes written per second to the Web server log decreases only 2% between Test 2 and Test 3.

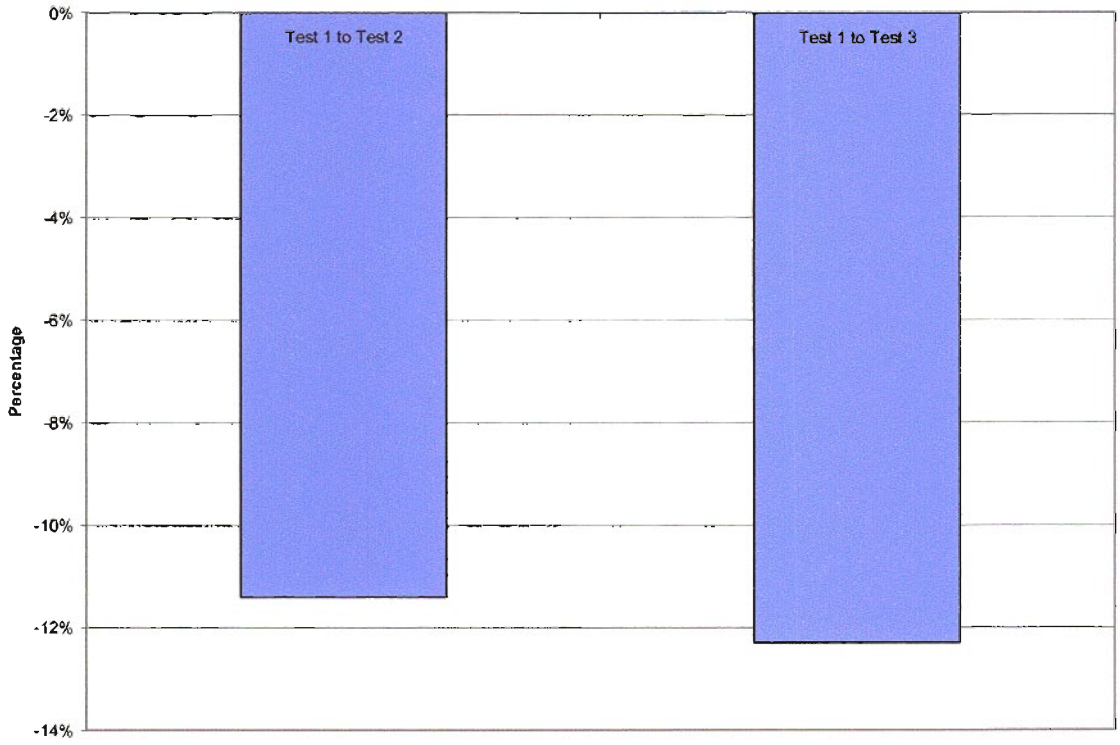

Figure 5-6: Change in Web Server Log Writes per Second

Figure 5-6 displays the percentage difference in the number of times the Web server log is written to between Test 1 and Test 2, and between Test 1 and Test 3. The addition of the clickstream database server reduces the number of writes per second to the log file by almost 11.5%. Consolidating the Web server while redundantly logging clickstream data and the clickstream database server reduces the number of writes per second to the log file by about 12.0%. The number of writes decreases about 0.5% between Test 2 and Test 3.

The results of the hard disk drive activity for the Web server log do not show any significant performance degradation in the system, which supports the hypothesis that the Web server and the clickstream database server do not compete for the same resources when both components are consolidated onto a single system. The decrease in the number of bytes written per second in Test 2 and Test 3 is supported by the number of bytes per write and in the number of writes per second to the Web server log. The latter two metrics support the result shown by the number of bytes written per second to the Web server log. The overall decrease in the hard disk drive activity also coincides with

the decrease in the number of HTTP requests serviced per second in Test 2 and Test 3. Since there is a decrease in the number of HTTP requests serviced per second, there must be a decrease in the amount of data written to the Web server log file.

5.3.2 Clickstream Database User Tablespace

The user tablespace is a fixed amount of space reserved on the hard disk drives by the database to use for the data of any number of tables within the database. In this study, the only table in the user tablespace is the HTTP_LOG table.

Since the clickstream database server is configured to always write 4,096 bytes per write to the user tablespace, there is no change in the number of bytes per write to the HTTP_LOG table between any of the tests.

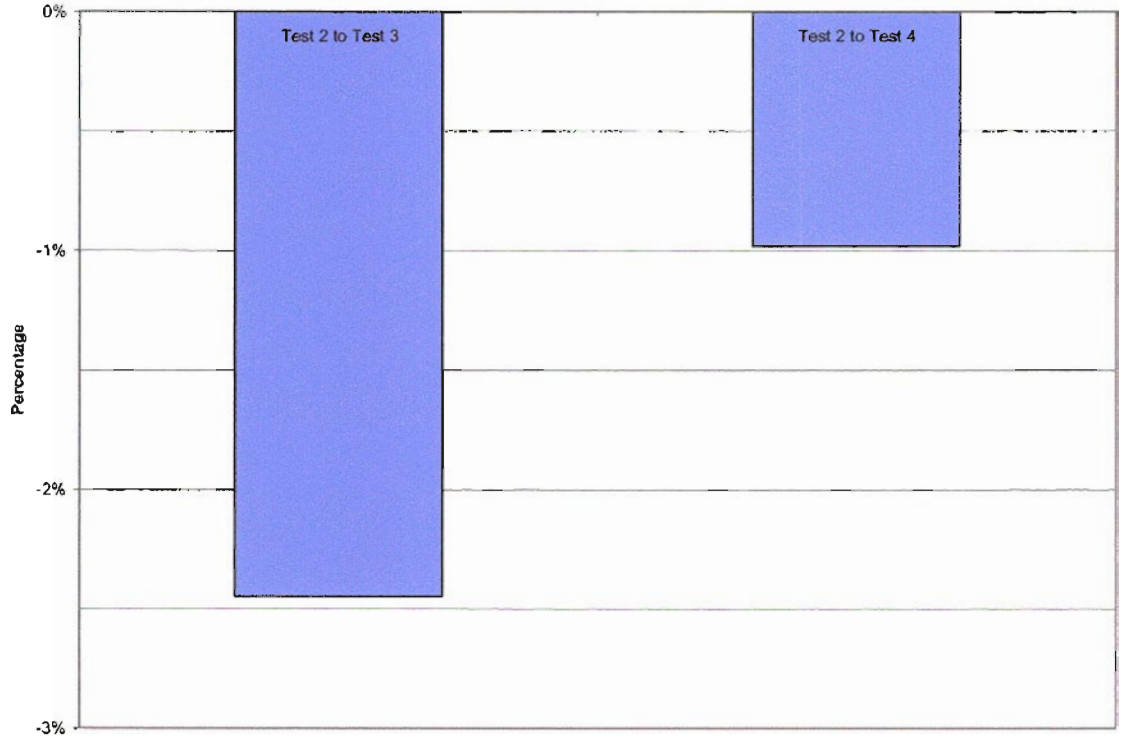

Figure 5-7: Change in Clickstream Database User Tablespace Bytes Written per Second

Figure 5-7 displays the percentage difference in the number of bytes written per second to the user tablespace between Test 2 and Test 3, and between Test 2 and Test 4. Consolidating the Web server while redundantly logging clickstream data and the clickstream database server reduces the number of bytes written per second to the user tablespace by almost 2.5%, and reduces the number of bytes written per second by almost

1.0% without the Web server redundantly logging clickstream data. The number of bytes written per second increases almost 1.5% between Test 3 and Test 4.

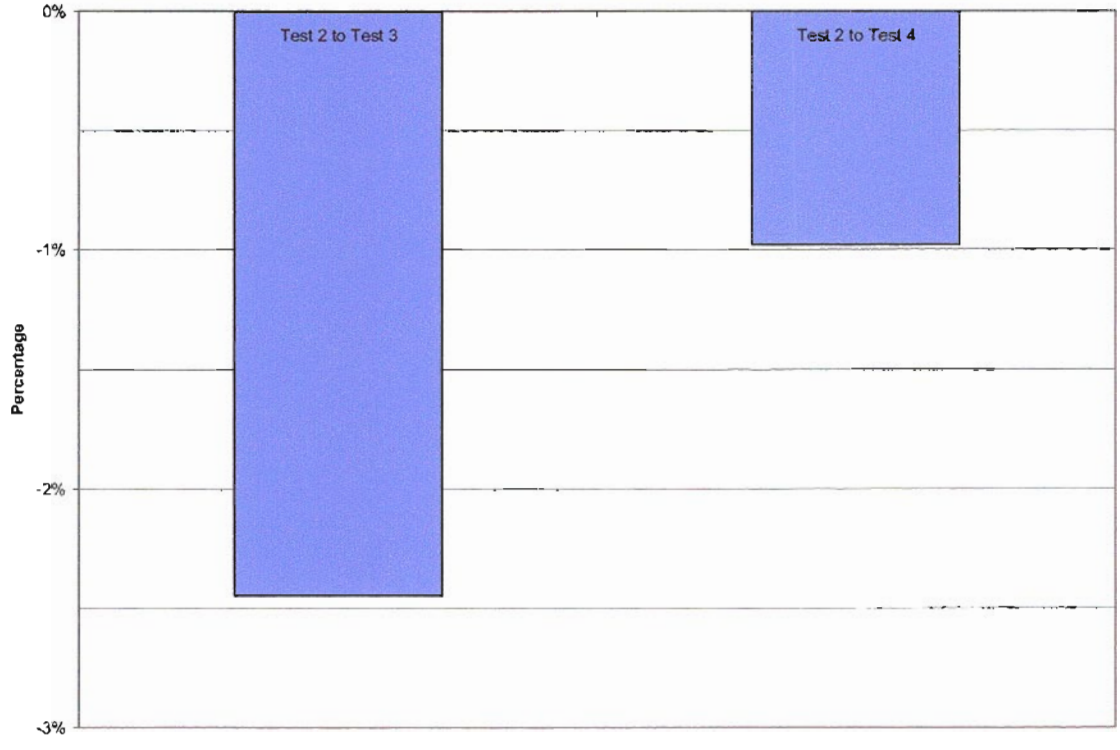

Figure 5-8: Change in Clickstream Database User Tablespace Writes per Second

Figure 5-8 displays the percentage difference in the number of times the user tablespace is written to per second between Test 2 and Test 3, and between Test 2 and Test 4. Consolidating the Web server while redundantly logging clickstream data and the clickstream database server reduces the writes per second to the user tablespace by almost 2.5%, and reduces the number of writes per second by almost 1.0% without the Web server redundantly logging clickstream data. The number of writes per second increases almost 1.5% between Test 3 and Test 4.

The results of the hard disk drive activity for the user tablespace do not show any significant performance degradation, which supports the hypothesis that the Web server and the clickstream database server do not compete for the same resources when both components are consolidated onto a single system. The decrease in hard disk drive activity also coincides with the decrease in the number of HTTP requests serviced per second in Test 2, Test 3, and Test 4 when compared to Test 1. Since there is a decrease

in the number of HTTP requests serviced per second, there must be a decrease in the amount of data written to the HTTP_LOG table in the user tablespace.

5.3.3 Clickstream Database Log

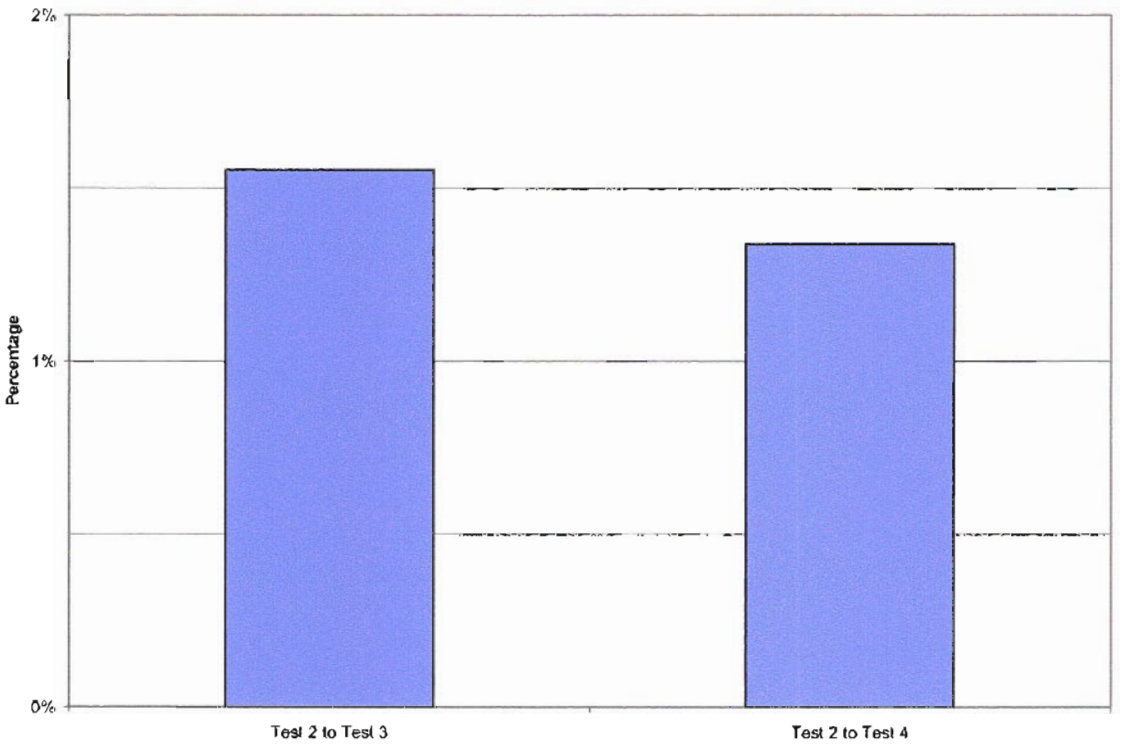

Figure 5-9: Change in Clickstream Database Log Bytes per Write

Figure 5-9 displays the percentage difference in the number of bytes per write to the database log between Test 2 and Test 3, and between Test 2 and Test 4. Consolidating the Web server while redundant logging clickstream data and the clickstream database server increases the number of bytes per write to the database log by about 1.5%, and increases the number of bytes per write by almost 1.5% without the Web server redundantly logging clickstream data. The number of bytes per write increases about 0.5% between Test 3 and Test 4.

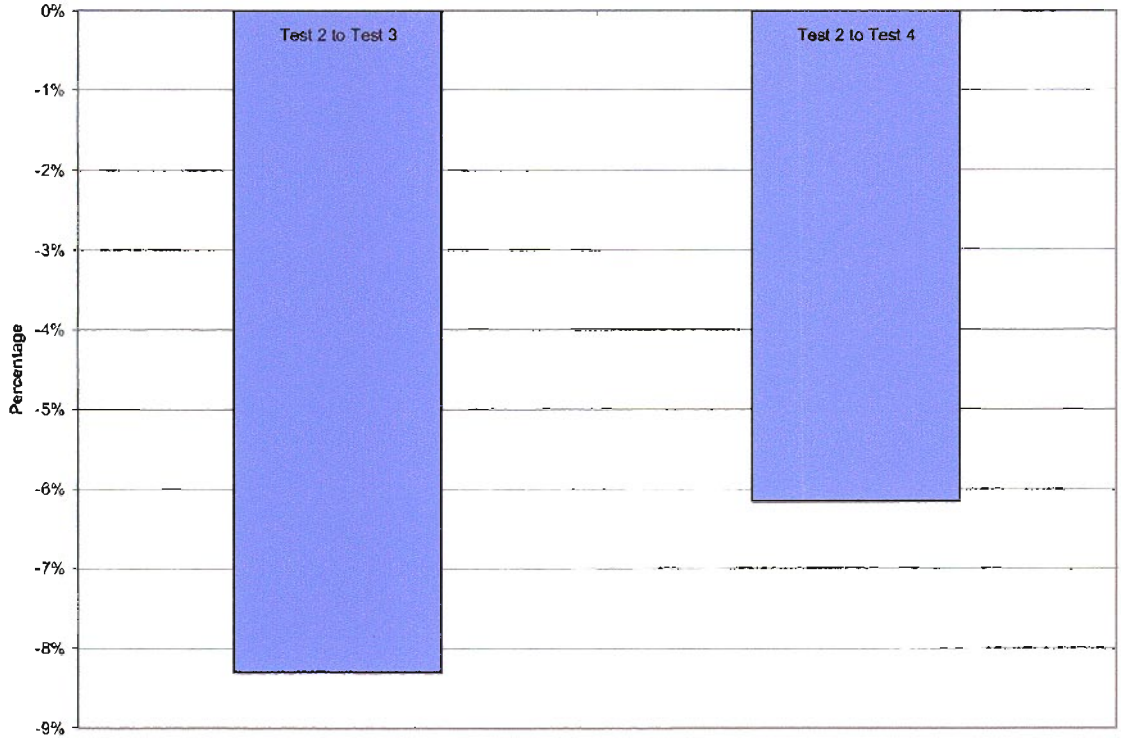

Figure 5-10: Change in Clickstream Database Log Bytes Written per Second

Figure 5-10 displays the percentage difference in the number of bytes written per second to the database log between Test 2 and Test 3, and between Test 2 and Test 4. Consolidating the Web server while redundantly logging clickstream data and the clickstream database server decreases the number of bytes written per second to the log file by about 8%, and decreases the number of bytes written per second by about 6% without the Web server redundantly logging clickstream data. The number of bytes written per second increases about 6% between Test 3 and Test 4.

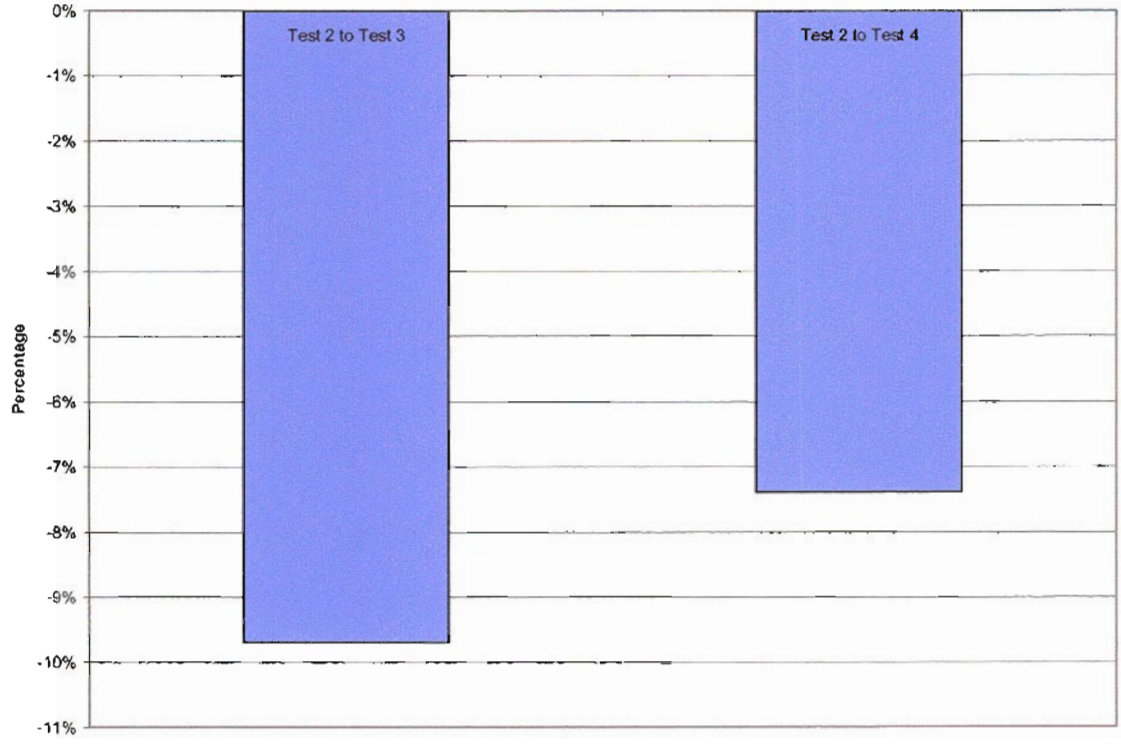

Figure 5-11: Change in Clickstream Database Log Writes per Second

Figure 5-11 displays the percentage difference in the number of writes to the database log per second between Test 2 and Test 3, and between Test 2 and Test 4. Consolidating the Web server while redundantly logging clickstream data and the clickstream database server decreases the number of writes to the database log per second about 9.5%, and decreases the number of writes to the database log per second by about 7.5% without the Web server redundantly logging clickstream data. The number of bytes written per second increases about 2.0% between Test 3 and Test 4.

The results of the hard disk drive activity for the database log do not show any significant performance degradation, which supports the hypothesis that the Web server and the clickstream database server do not compete for the same resources when both components are consolidated onto a single system. The decrease in the number of bytes written per second in Test 2 and Test 3 is supported by the number of bytes per write and in the number of writes per second. The latter two metrics support the result shown by the number of bytes written per second to the database log. The overall decrease in hard disk drive activity also coincides with the decrease in the number of HTTP requests

serviced per second in Test 2 and Test 3. Since there is a decrease in the number of HTTP requests serviced per second, there must be a decrease in the amount of data written to the Web server log file.

5.4 Network Traffic

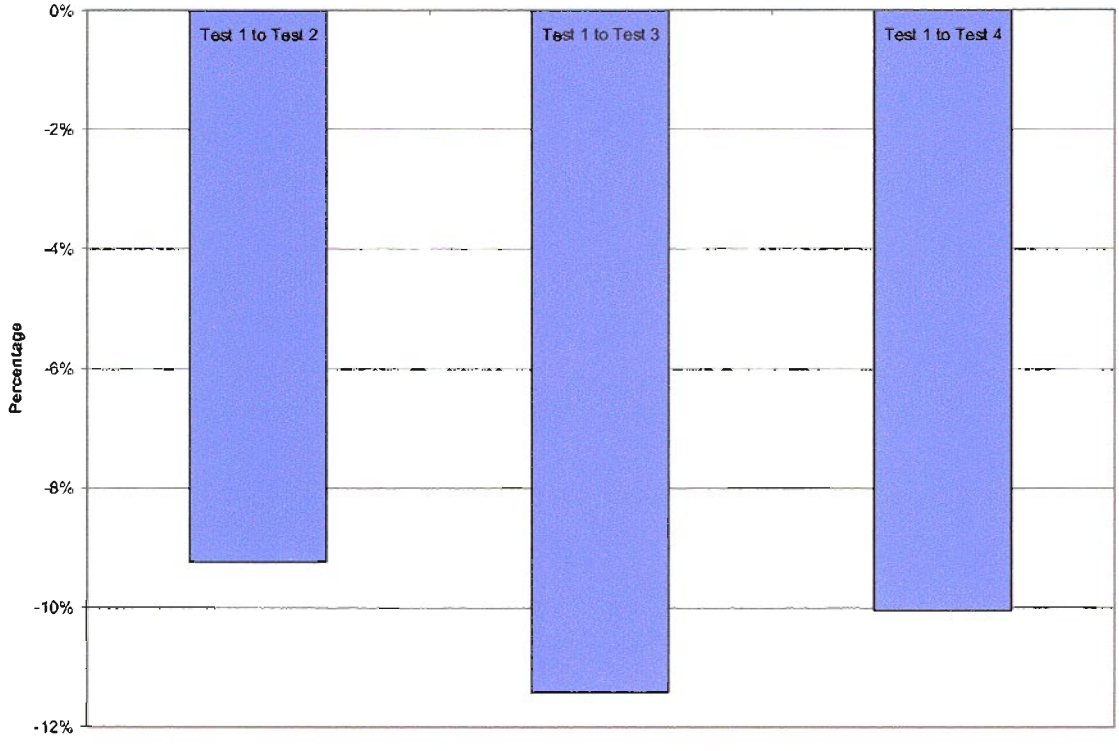

Figure 5-12: Change in Total Network Bytes Transmitted per Second

Figure 5-12 displays the percentage difference in the total number of bytes sent over the network between the RBE and the Web server from Test 1 to Test 2, from Test 1 to Test 3, and from Test 1 to Test 4. The use of a clickstream database server reduces the amount of data transmitted over the network by about 9.0%. Consolidating the Web server while redundantly logging clickstream data and the clickstream database server onto a single system reduces the data transmitted by about 11.5%, and reduces the data transmitted by about 10.0% without the Web server redundantly logging clickstream data. The total number of bytes transmitted per second decreases only 2.5% between Test 2 and Test 3, and increases 1.5% between Test 3 and Test 4.

The network traffic verifies the findings reported by the number of HTTP requests serviced per second as shown in Figure 5-1 and the change in performance between each test follows the changes calculated from the results of the number of HTTP requests per second.

6 Conclusion

In this thesis, I implemented a method of having a Web server directly store clickstream data into a clickstream database with both components consolidated onto a single, highly scalable system. The purpose was to determine if this method could be done without a significant change in performance to the Web server or the clickstream database server. This method was designed based on the ideas of getting clickstream data into a clickstream database as soon as possible and to do this on a single, highly scalable system.

Getting clickstream data into a clickstream database was important for analyzing customer behaviors. While this also allows the data to be queried for simple statistics, the data can be further integrated with a data warehouse. This will allow analyses for customer behaviors to be performed as clickstream data streams into the database.

Using a single, highly scalable system has several advantages that can be utilized. The system can be upgraded to accommodate the demands of any business and all resources in the system can be dynamically allocated between tasks. Also, using a single system reduces maintenance costs associated with having a large number of systems.

The experiments in this study modeled a Web site based on the TPC-W benchmark and used three tests to measure the overall performance of the system under test. The primary resources used by the Web server and the clickstream database server, such as processor utilization and the writing activities to the hard disk drives, were measured. The first test measured the performance of the Web server before a clickstream database was implemented. The second test measured the performance of the Web server inserting clickstream data into a clickstream database before the two components were consolidated onto a single system. The third test measured the performance of the Web server and the clickstream database server when they were consolidated onto a single system with redundant clickstream logging by the Web server,

while the fourth test measured the performance of the same system without redundantly logging clickstream data.

The most significant change in performance was caused by the additional time taken to insert clickstream data into the clickstream database, which was measured by the length of time to receive a Web page. The additional time averaged less than one millisecond, which is hardly noticeable for people surfing the Web. The results also showed that the Web server did not compete with the clickstream database server for hard disk drive resources, and that the clickstream database server did not use very much processing resources. The data collected on the performance of this implementation suggested that consolidating a Web server and a clickstream database server could be done without a significant performance impact to either the Web server or the clickstream database server. The data also showed that redundantly logging clickstream data required slightly more resources then a system that did not log clickstream data redundantly. Since it is a slight amount, the decision of redundantly logging clickstream data can be left to the system administrators. These results made server consolidation a feasible option for companies looking for better manageability of their systems and for reducing costs.

If the Web application used a content database to generate HTML pages, as defined in the TPC-W specification, the results shown in this study should still hold true. The content database has its own requirements of physical resources and it should not conflict with the resource requirements of the other components in the system. For example, hard disk drives could be dedicated to the content database since the hard disk drives do not need to be shared between databases or any other components in the system. Proving this hypothesis has been left for future study, but with the ability to dedicate resources to individual components, it would be likely that introducing a content database would not create resource conflicts between components on the system.

Likewise, if clickstream analysis were implemented on a consolidated system, some system resources could be allocated to minimize conflicts for those resources when performing the analysis. Again, this has been left for future study, but may be a more difficult feat to accomplish than implementing a content database. While a content database was a self-contained component of the system, the task of performing

clickstream analysis would draw from the clickstream database used in this study. Additional components must be implemented to perform clickstream analysis and these components would need to extract the data from the clickstream database. Minimizing conflicts on the clickstream database may be a difficult task.

7 Future Work

This thesis presents a start for three branches of study. First, more work can be done on characterizing the consolidation of more components in a Web site. Also additional work can be done on implementing additional components for clickstream analysis. Finally, studies typically done on Web server log files can be adapted to work with databases.

7.1 Web Site Consolidation

Further study on the performance of system consolidation can be done with a more complete implementation of a Web site. The implementation of this study is simple when compared to all of the components in a real world Web site. A Web site may use reverse proxy servers, Web caches, text search engines, and a database to supply content, as mentioned previously in Sub-subsection 3.3.3. Reverse proxy servers can be implemented for testing stronger security measures. Web caches can be implemented to better represent a real world Web site, which would reduce the workload on the Web servers. The Web caches would also insert clickstream data into the clickstream database because the Web caches also services HTTP requests. Running two databases, one for content and one for clickstream analysis, would verify whether or not they could be configured such that they can run together on a single system with minimal conflicts.

7.2 Clickstream Analysis

The tests presented in this thesis can be modified to simulate users who do not wish to be tracked in addition to users that can be tracked. Companies do not always have the luxury of having readily identifiable customers and some companies are not prepared to track their customers. Spiliopoulou [16] presents a general mining algorithm for identifying a sequence of HTTP requests that represent a user session. Nasraoui, Krishnapuram, and Joshi [17] identify user sessions in Web server log data where cookies and URL tagging have not been used. While cookies are used in this study, the research

done by Nasraoui, Krishnapuram, and Joshi can be adapted for users who refused to accept cookies.

Additional performance studies can be done with more components of the clickstream analysis architecture implemented. This study only implements the first couple of steps in the architecture where clickstream data is cleaned and inserted into a data warehouse staging area. The next steps would be to implement a process to identify transactions, integrate the clickstream data with other data sources, and finally import clickstream data into a data warehouse.

Cooley, Mobasher, and Srivastava [18] have conducted research on grouping sequences of HTTP requests into transactions. They have defined transactions to consist of two types, those for navigational purposes and those for information content purposes. Their paper describes a general model for identifying transactions that could be applied to raw Web server log files or clickstream data that has been inserted into a database.

TPC-H [19] is a decision support benchmark developed by the TPC. It may be possible to use the clickstream data generated by a TPC-W benchmark in conjunction with the TPC-H benchmark to create a data webhouse and to continue studying the performance of the system.

7.3 Using Databases for Clickstream Analysis

Previous research conducted on Web server log files can be adapted to work with clickstream data in a database. This thesis introduces a method for getting clickstream data into a database, which may be a better medium for research than raw Web server log files. The following research has been based on using raw Web server log files and are candidates for using a clickstream database for further study.

Chen, Park, and Yu [20] performed research using association rules to identify sequences of *maximal forward references*. Maximal forward references represent a path a Web user traverses on a Web site without using a browser's back button or without traversing a link that takes a user back to where he has been before. The goal of identifying maximal forward references is to determine how well a Web site is laid out. These maximal forward references are small if it is easy for a Web user to find what he is looking for.

Lo and Ng [21] base some of their research on work done by Chen, Park, and Yu to generate association rules that allow companies to analyze customer behaviors in order to improve profits and services. These association rules determine facts such as "10% of our customers buy bread and 20% of those also buy butter."

Osawa, Asai, and Ohnishi [22] explore creating three-dimensional animations to model Web access behavior. They use the Virtual Reality Modeling Language (VRML) and Java to render a dynamic system model as graphical objects in a three-dimensional space.

References

[1] Richard Winter. More Than You Hoped For. *Intelligent Enterprise*, Vol. 3, No. 6, April 10, 2000, pages 63-65.

[2] RFC 2616: Hypertext Transfer Protocol – HTTP/1.1. URL:<ftp://ftp.isi.edu/in-notes/frc2616.txt>, June 1999.

[3] Logging in W3C httpd. URL:<http://www.w3.org/Daemon/User/Config/Logging.html>, July 1995.

[4] Ralph Kimball and Richard Merz. *The Data Warehouse Toolkit*. John Wiley & Sons, Inc., 2000.

[5] R. Cooley, B. Mobasher, and J. Srivastava. Web Mining: Information and Pattern Discover on the World Wide Web. *Proceedings of the 9th International Conference on Tools with Artificial Intelligence*, Newport Beach, California, 1997.

[6] RFC 2965: HTTP State Management Mechanism. URL:<ftp://ftp.isi.edu/in-notes/rfc2965.txt>, October 2000.

[7] *TPC Benchmark™W*. Specification Version 1.1. June 27, 2000.

[8] I-Yuen Lin, Xin-Mao Huang, and Ming-Syan Chen. Capturing User Access Patterns in the Web for Data Mining. *Proceedings, 11th IEEE Internaional Conference on Tools with Artificial Intelligence*, pages 345-348, Chicago, Illinois, 1999.

[9] *TPC-W Results List*. Transaction Processing and Performance Council Web Site. URL<http://www.tpc.org/tpcw/results/tpcw_results.asp> September 17, 2001.

[10] Ralph Kimball. The Perfect Handoff. *Intelligent Enterprise*, Vol. 2, No. 18, December 21, 1999, pages 27-29.

[11] Mary Edie Meredith, 2000. Private communication.

[12] Lincoln D. Stein. The Joy of SQL, *Web Techniques*, Vol. 3, No. 10, October 1998, pages 14-20.

[13] *Data Warehousing – Scaling the Data Refinement Process*. A Sun Microsystems Executive Brief. URL:<http://www.torrent.com/pages/products/wpform.html>, November 1997.

[14] Adam Warkow. Planning for Data Center Consolidations. *The 20th International Conference for the Resource Management and Performance Evaluation of Enterprise Computing Systems*, pages 1001-1004, Orlando, Florida, December 4-9, 1994.

[15] James Pitkow. In Search of Reliable Usage Data on the WWW. *Sixth International World Wide Web Conference*, pages 451-463, Santa Clara, California, 1997.

[16] Myra Spiliopoulou. The laborious way from data mining to web log mining. *Computer Systems Science and Engineering*, Vol. 14, No. 2, March 1999, pages 113-126.

[17] Olfa Nasraoui, Raghu Krishnapuram, and Anupam Joshi. Relational Clustering Based on a New Robust Estimator with Application to Web Mining. *International Conference of the North American Fuzzy Information Processing Society*, pages 705-709, 1999.

[18] R. Cooley, B. Mobasher, and J. Srivastava. Grouping Web Page References into Transactions for Mining World Wide Web Browsing Patterns. *Proceedings of the 1997 IEEE Knowledge and Data Engineering Exchange Workshop*, Newport Beach, California, 1997.

[19] *TPC Benchmark™H.* Specification Version 1.3.0. April 24, 2000.

[20] Ming-Syan Chen, Jong Soo Park, and Philip S. Yu. Data Mining for Path Traversal Patterns in a Web Environment. *Proceedings of the 16th International Conference on Distributed Computing*, pages 385-392, Hong Kong, 1996.

[21] Charles Lo and Vincent Ng. Discovering Web Access Orders with Association Rules. *IEEE SMC'99 Conference Proceedings*, pages 99-104, Tokyo, Japan, 1999.

[22] Noritaka Osawa, Kikuo Asai, and Hitoshi Ohnishi. Three Dimensional Animation of Web Access Behavior Using Indirect Mapping. *IEEE International Conference on Information Visualization*, pages 502-507, 1999.

Appendix A Raw Data

This section presents graphs of the raw data collected from each of the tests described in the Section 5. Each graph shows approximately one hour of data collected from the performance counters in the operating system. The data are sampled every thirty seconds.

A.1 HTTP Requests per Second

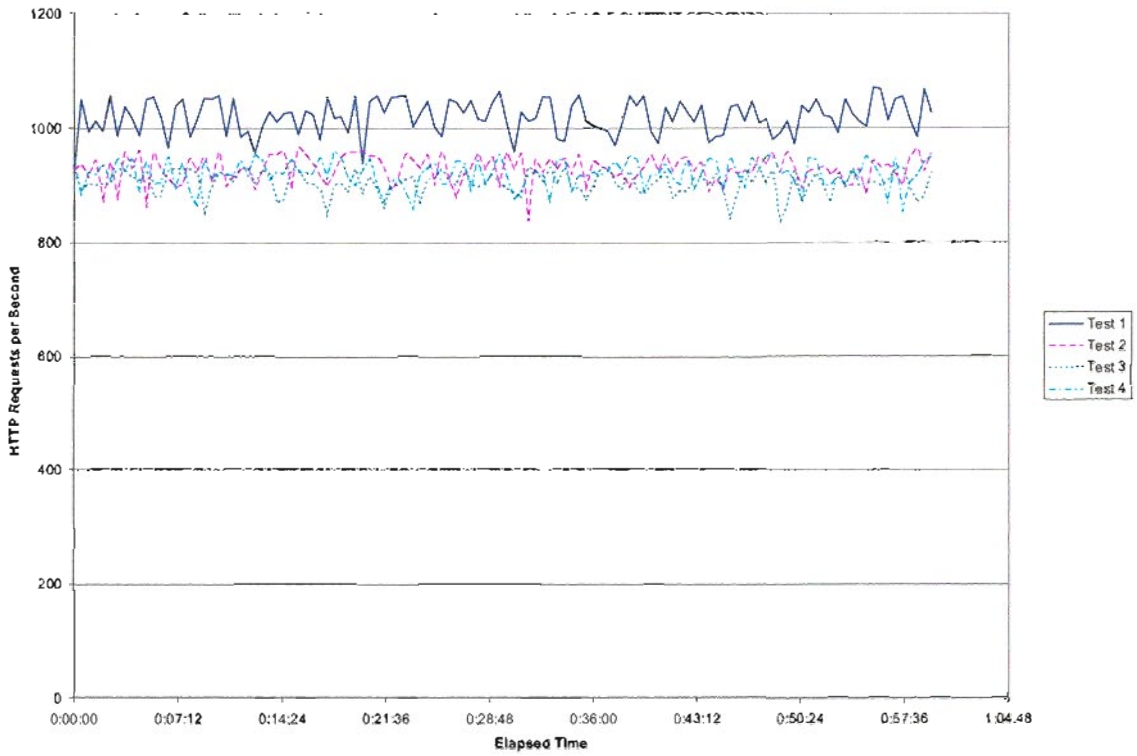

Figure A-1: HTTP Requests per Second

Figure A-1 displays the total throughput of the system for each test. Each test displays a steady and consistent level of throughput. Test 1 services an average of 1,020 HTTP requests per second, Test 2 services 927 HTTP requests per second, Test 3 services 904 HTTP requests per second, and Test 4 services 919 HTTP requests per second.

A.2 Web Server Processor Utilization

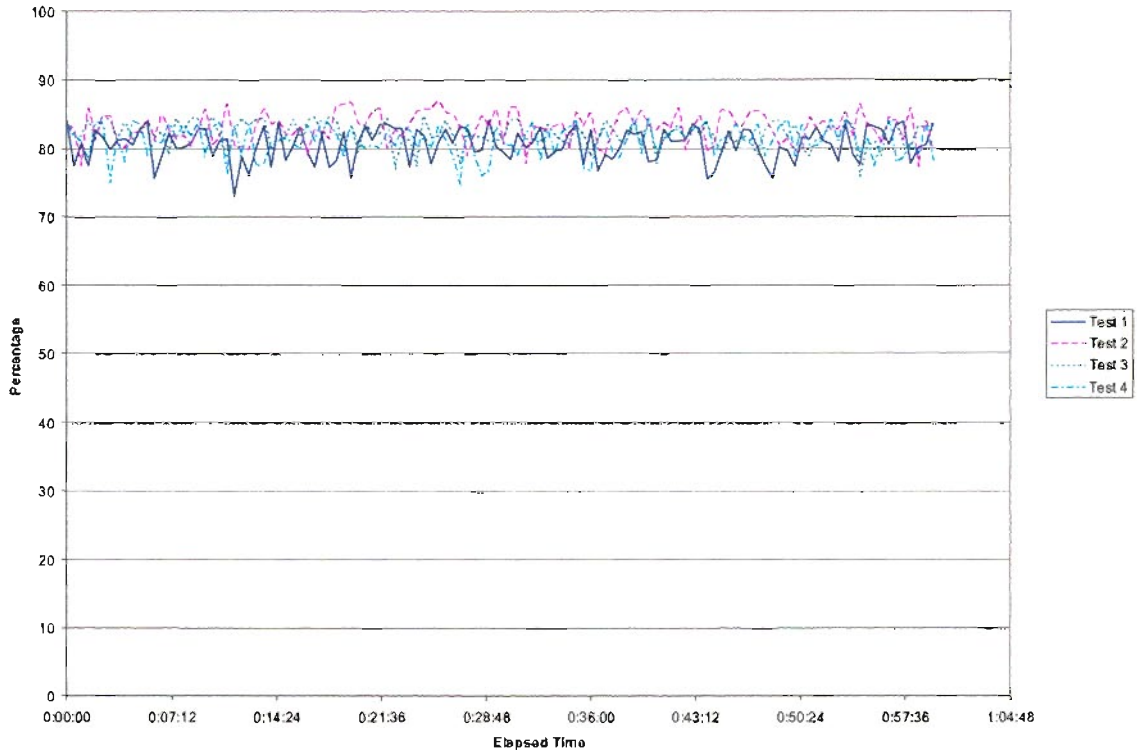

Figure A-2: Web Server Processor Utilization

Figure A-2 displays the processor utilization by the Web server in each test. Each test displays a steady and consistent use of the processors throughout the duration of each test. The Web server is the primary user of the processors, utilizing the processor more than 80% for each test. The Web server in Test 1 utilizes the processors an average of 81%, Test 2 utilizes the processors an average of 83%, Test 3 utilizes the processors an average of 82%, and Test 4 utilizes the processors an average of 81%.

A.3 Clickstream Database Server Processor Utilization

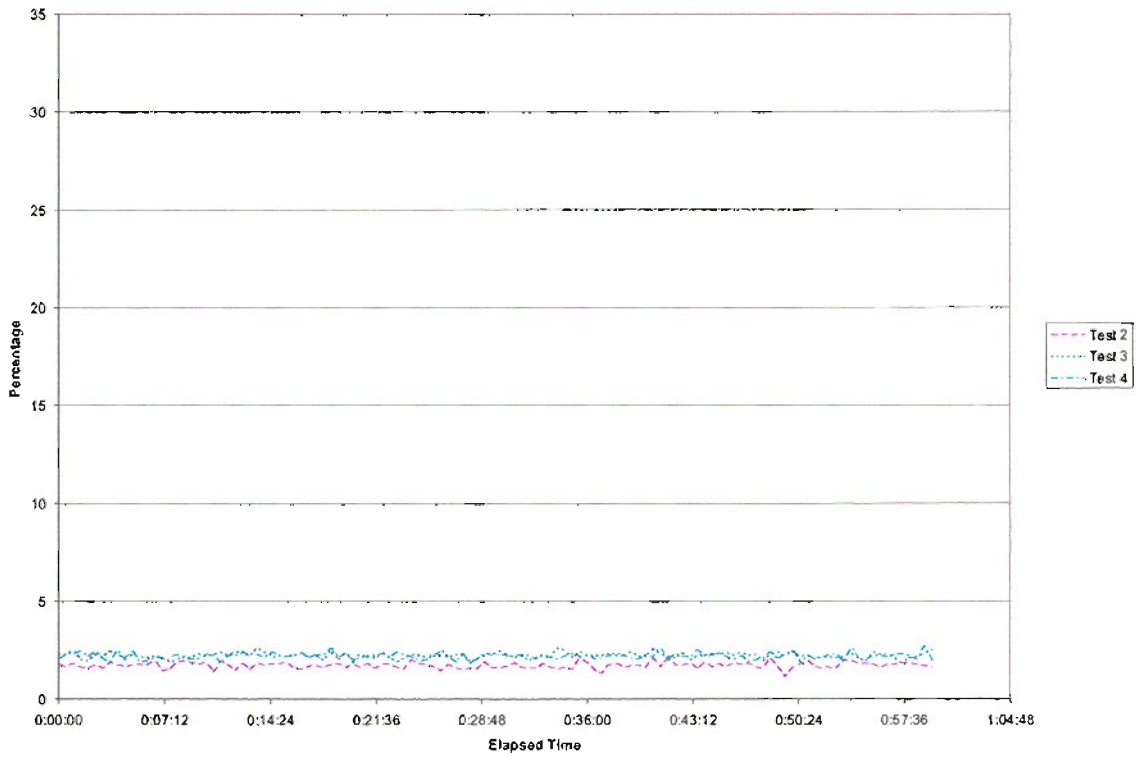

Figure A-3: Clickstream Database Server Processor Utilization

Figure A-3 displays the processor utilization of the clickstream database server for Test 2, Test 3, and Test 4 since the clickstream database server is not used in Test 1. Test 2, Test 3, and Test 4 display a steady and consistent use of the processors by the clickstream database server. The clickstream database server does not primarily require processing power as the processor utilization is below 3.0% for each test. The clickstream database server in Test 2 utilizes the processor 1.7%. Test 3 and Test 4 utilize the processor 2.2%. Consolidating the Web server and the clickstream database server increases the processor utilization of the clickstream database server by a 10% difference, but the change in processor utilization is still 0.1%.

A.4 Web Server Log Hard Disk Drive Activity

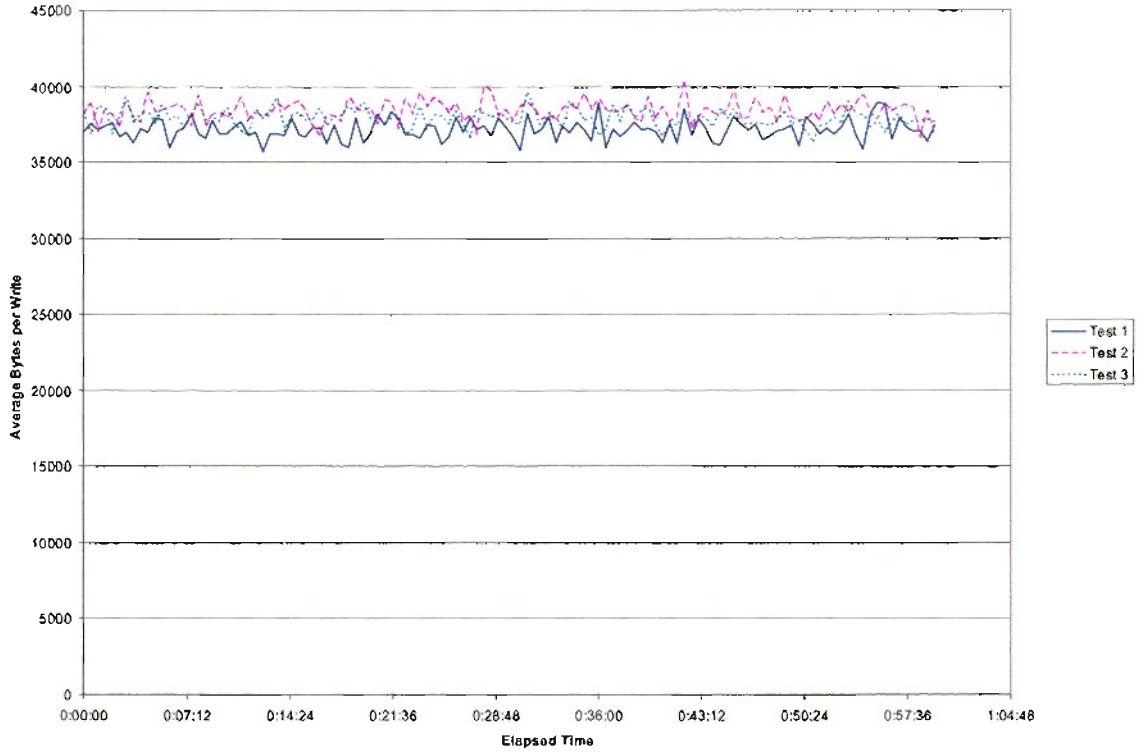

Figure A-4: Web Server Log Bytes per Write

Figure A-4 displays the number of bytes per write to the Web server log file for Test 1, Test 2, and Test 3 since the Web server is not logging clickstream data in Test 4. Each test results in a steady number of bytes per write to the Web server log file. The Web server in Test 1 writes an average of 37,000 bytes per write to the log file. Test 2 and Test 3 write an average of 38,000 bytes per write to the log file.

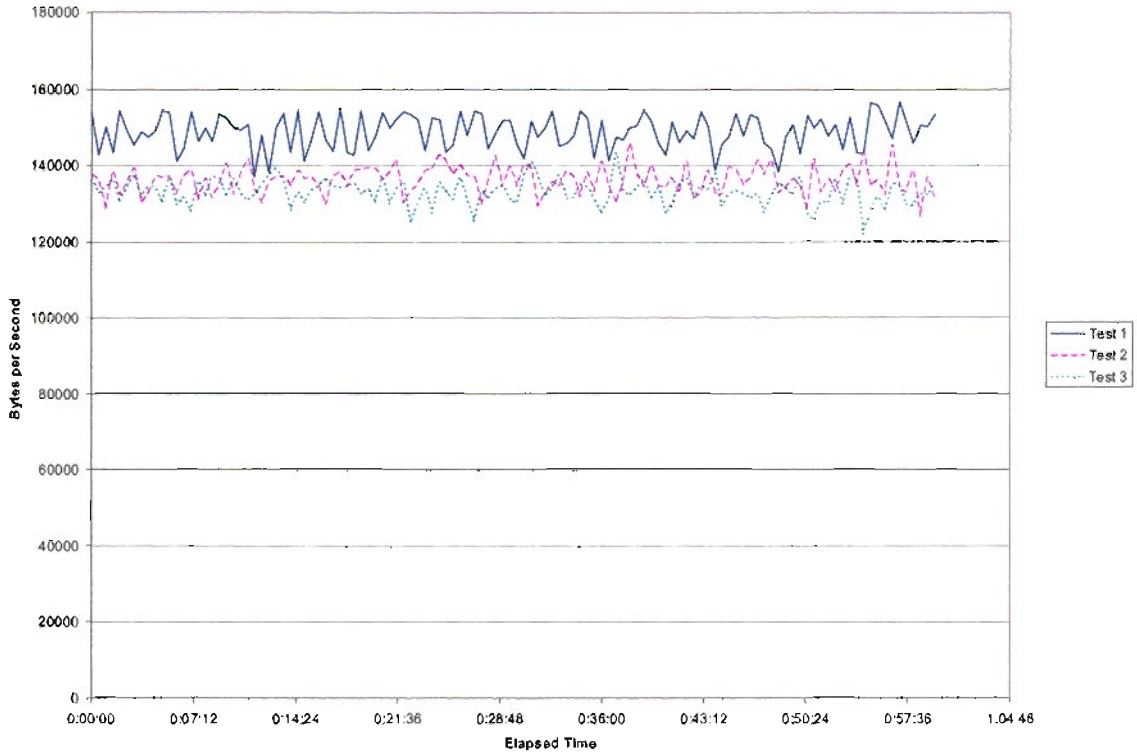

Figure A-5: Web Server Log Bytes Written per Second

Figure A-5 displays the number of bytes written to Web server log file per second for Test 1, Test 2, and Test 3 since the Web server is not logging clickstream data in Test 4. Each test results in a steady and consistent number of bytes written per second to the Web server log file. This verifies the total throughput of each test since the amount of data written to the Web server log file reflects the number of HTTP requests made to and serviced by the Web server. The Web server in Test 1 writes 149,000 bytes per second to the log file, Test 2 writes 136,000 bytes per second to the log file, and Test 3 writes 133,000 bytes per second to the log file.

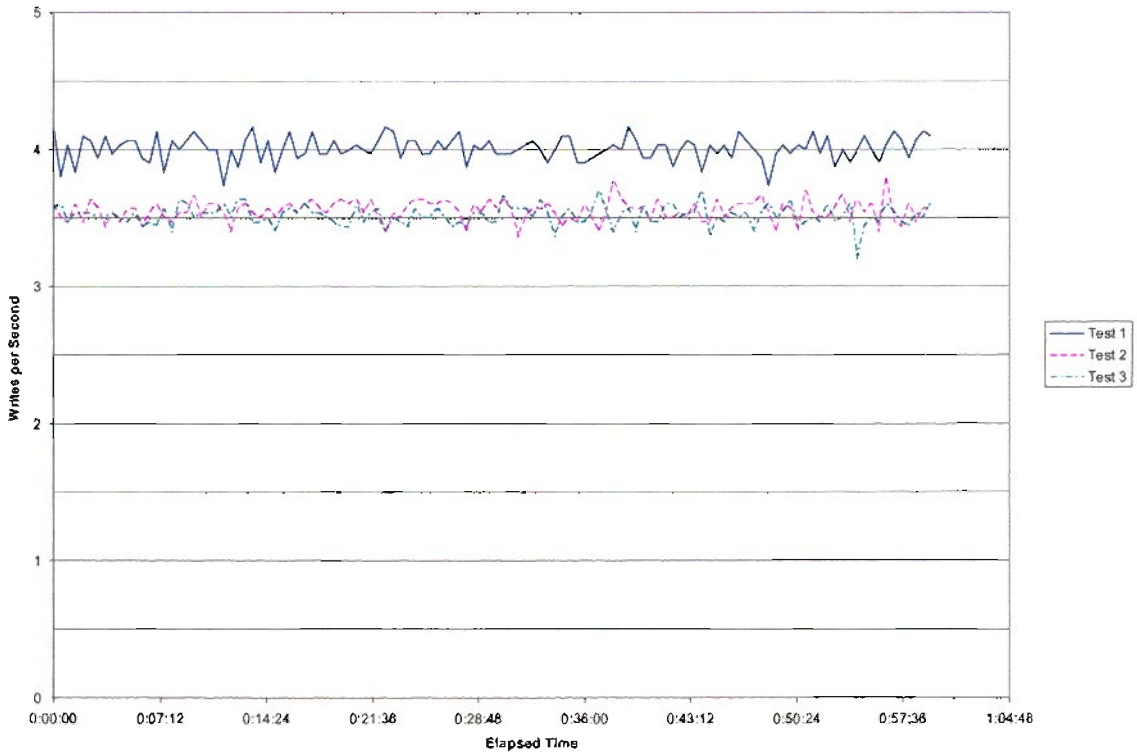

Figure A-6: Web Server Log Writes per Second

Figure A-6 displays the number of the number of times the Web server writes to its log file per second for Test 1, Test 2, and Test 3 since the Web server is not logging clickstream data in Test 4. Each test results in a steady and consistent number of writes to the Web server log file throughout the duration of each test. The Web server in Test 1 performs 4.0 writes per second to the Web server log file. Test 2 and Test 3 perform 3.5 writes per second to the Web server log file.

A.5 Clickstream Database User Tablespace Hard Disk Drive Activity

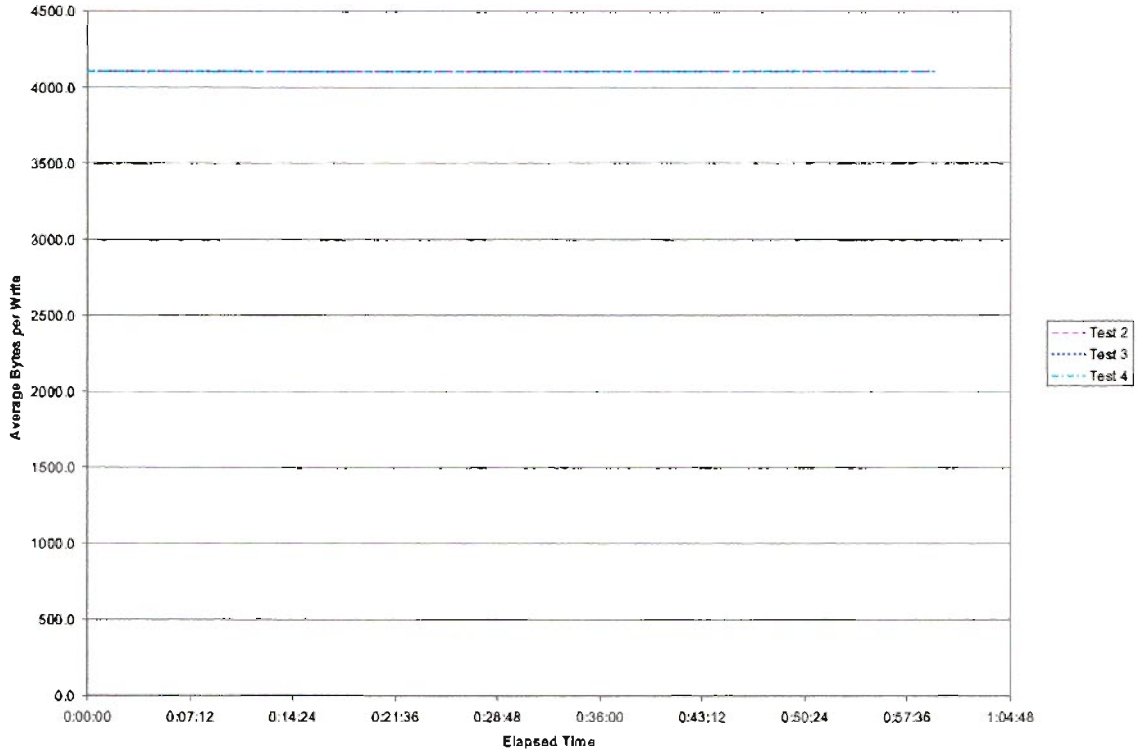

Figure A-7: Clickstream Database User Tablespace Bytes per Write

Figure A-7 displays the number of bytes written to the HTTP_LOG table in the clickstream database for Test 2, Test 3, and Test 4 since the clickstream database server is not used in Test 1. The clickstream database server in Test 2, Test 3, and Test 4 writes 4,096 bytes per write to the user tablespace. Since the data is identical in every test, only one line appears in Figure A-7. This verifies the database setting of a fixed 4KB block size for the tablespace where the HTTP_LOG table resides.

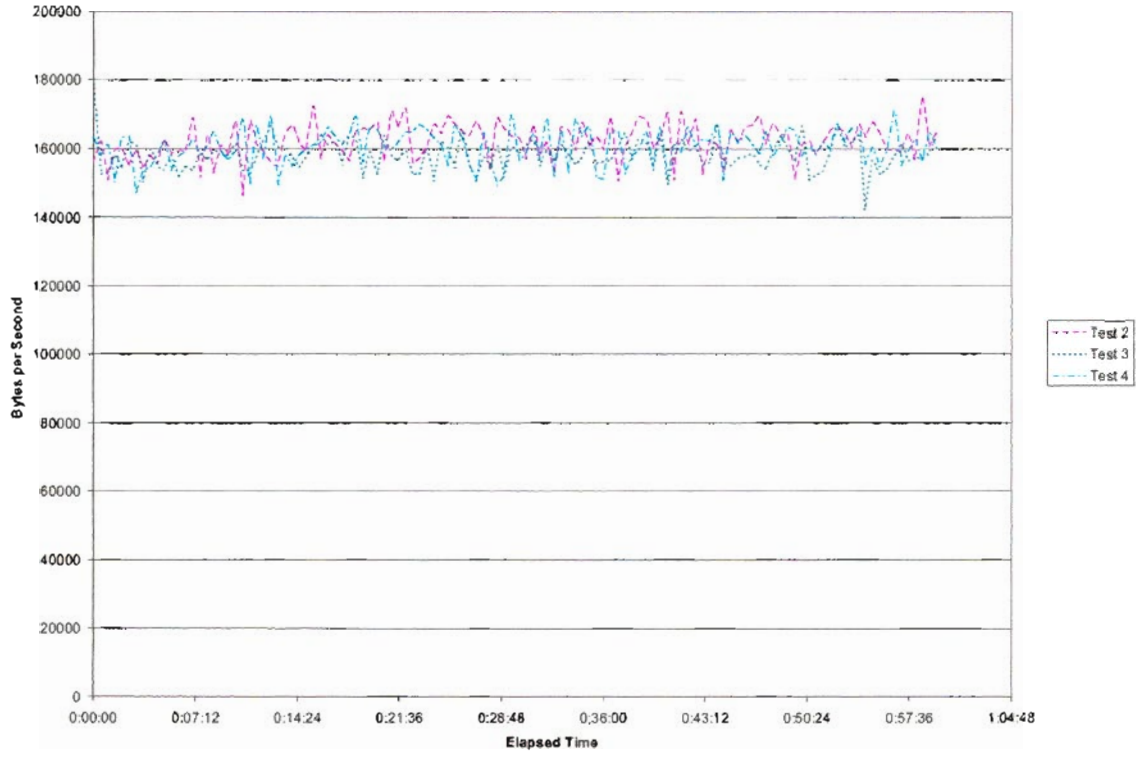

Figure A-8: Clickstream Database User Tablespace Bytes Written per Second

Figure A-8 displays the number of bytes written to the HTTP_LOG table in the clickstream database per second for Test 2, Test 3, and Test 4 since the clickstream database server is not used in Test 1. The clickstream database server in Test 2 writes an average of 162,000 bytes per second to the user tablespace, Test 3 writes an average of 158,000 bytes per second to the user tablespace, and Test 4 writes an average of 160,000 bytes per second to the user tablespace.

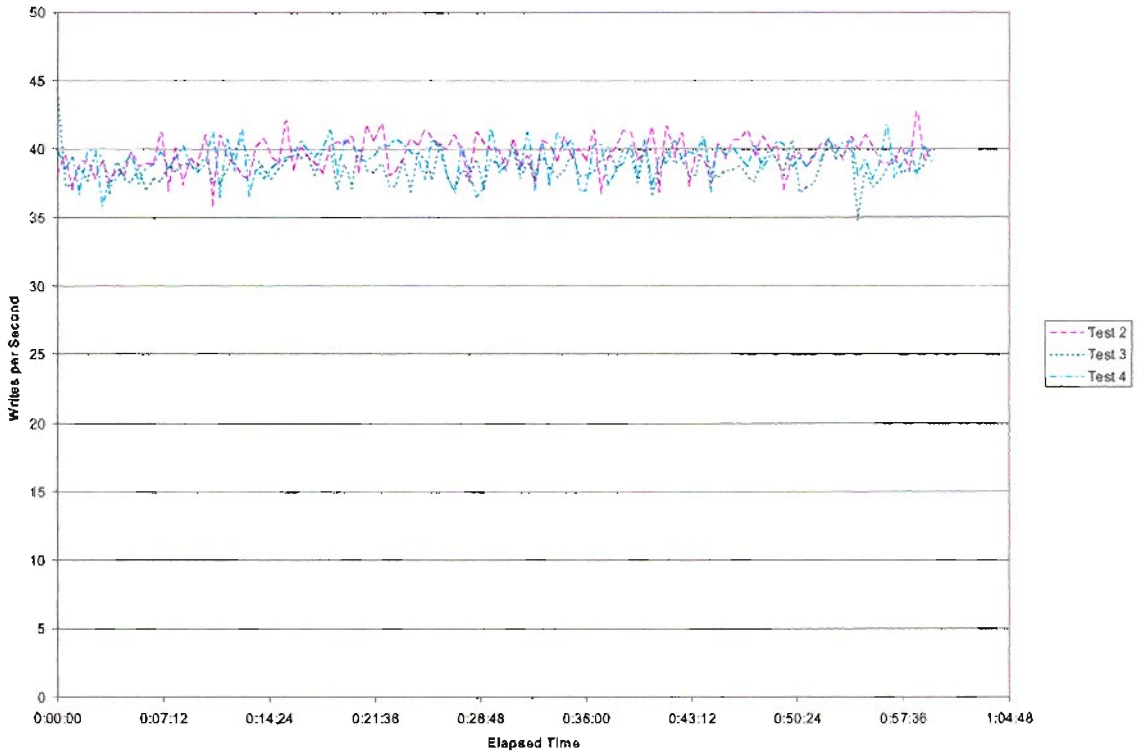

Figure A-9: Clickstream Database User Tablespace Writes per Second

Figure A-9 displays the number of writes to the HTTP_LOG table in the clickstream database per second for Test 2, Test 3, and Test 4 since the clickstream database server is not used in Test 1. The clickstream database server in Test 2 performs an average of 40 writes per second to the user tablespace. Test 3 and Test 4 performs an average of 39 writes per second to the user tablespace.

A.6 Clickstream Database Log Hard Disk Drive Activity

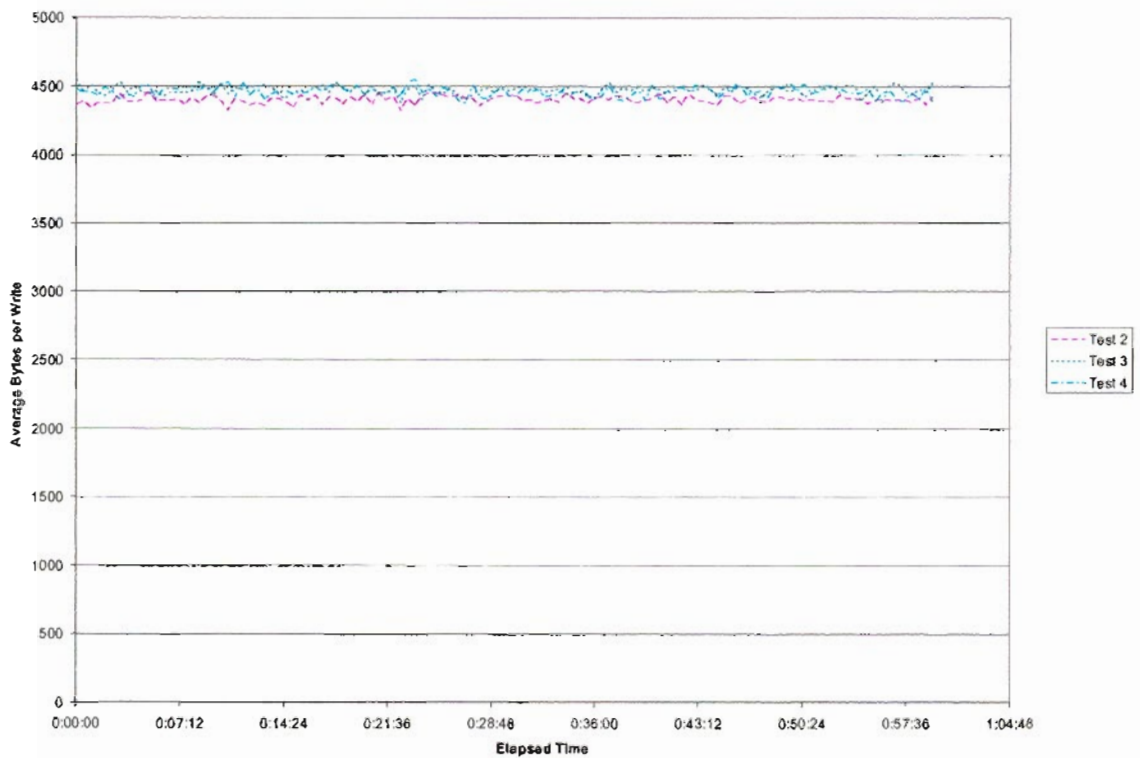

Figure A-10: Clickstream Database Log Bytes per Write

Figure A-10 displays the number of bytes per write to the database log in Test 2, Test 3, and Test 4 since the clickstream database server is not used in Test 1. Test 2, Test 3, and Test 4 result in a fairly steady and consistent number of bytes written per write to the database log throughout each test. The clickstream database server in Test 2 writes an average of 4,400 bytes per second to the database log. Test 3 and Test 4 writes an average of 4,500 bytes per second to the database log.

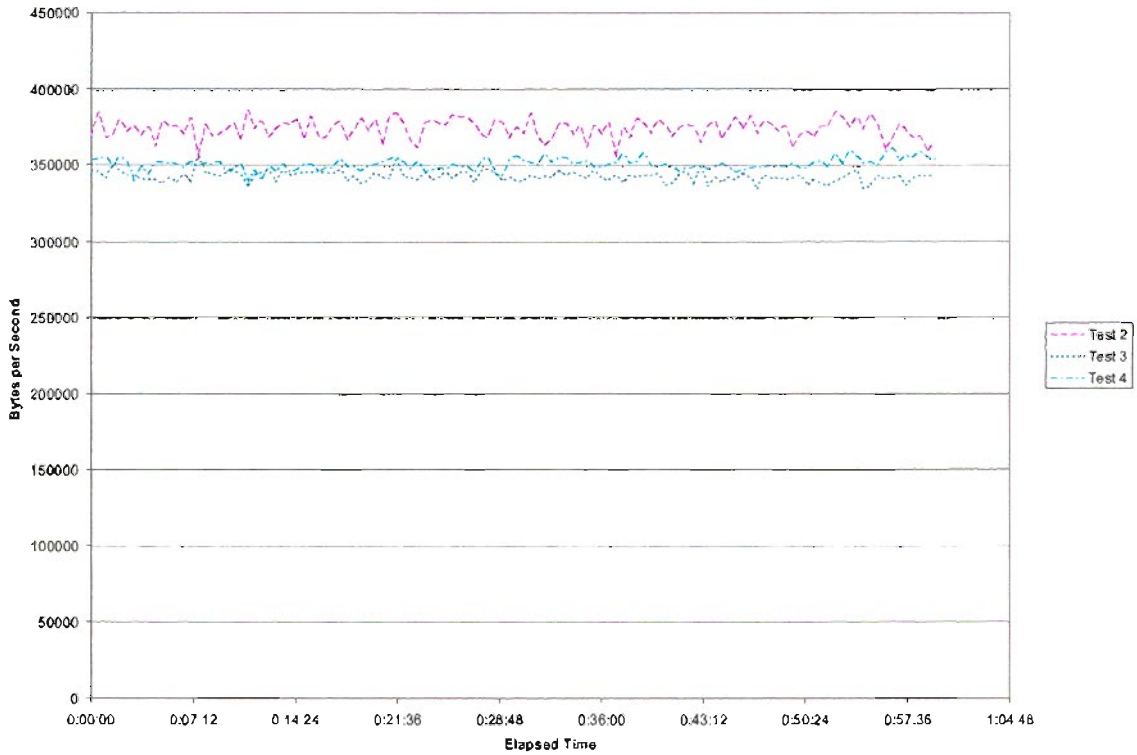

Figure A-11: Clickstream Database Log Bytes Written per Second

Figure A-11 displays the number of bytes written per second to the database log for Test 2, Test 3, and Test 4 since the clickstream database server is not used in Test 1. Test 2, Test 3, and Test 4 result in a fairly steady and consistent number of bytes written to the database log per second. The clickstream database server in Test 2 writes an average of 373,000 bytes per second to the database log, Test 3 writes an average of 342,000 bytes per second to the database log, and Test 4 writes an average of 350,000 bytes per second to the database log.

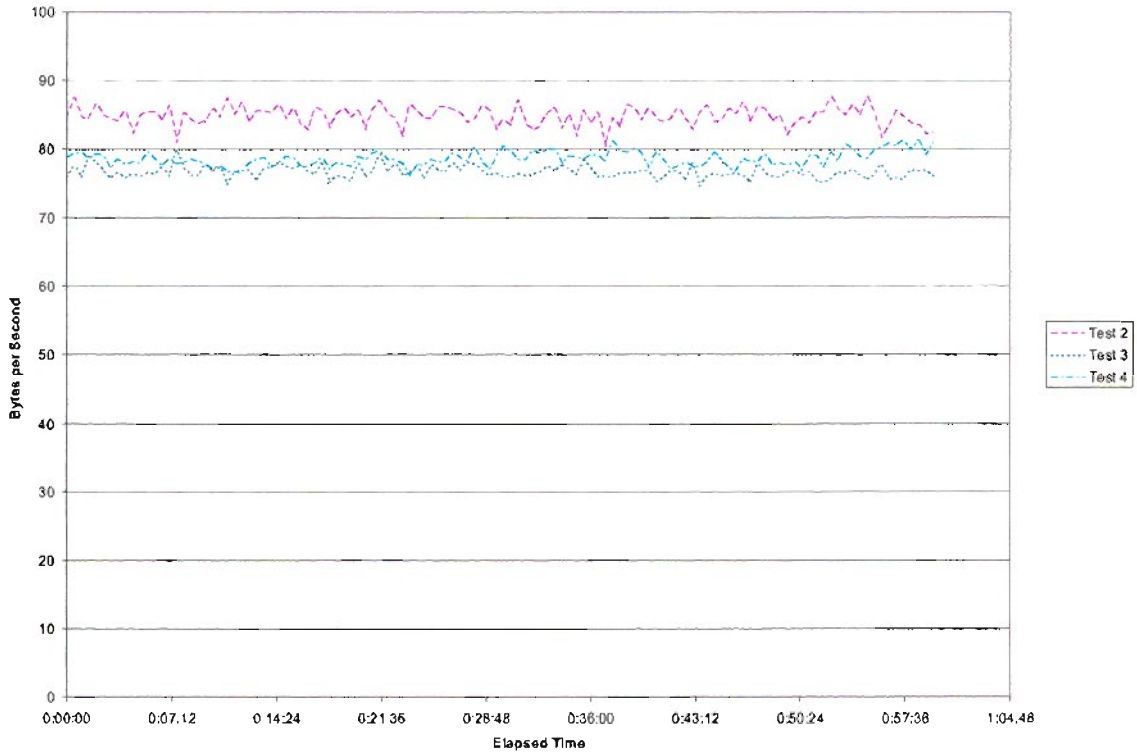

Figure A-12: Clickstream Database Log Writes per Second

Figure A-12 displays the number of writes to the database log for Test 2, Test 3, and Test 4 since the clickstream database server is not used in Test 1. Test 2, Test 3, and Test 4 result in a fairly steady and consistent number of writes to the database log throughout each test. The clickstream database server in Test 2 performs an average of 85 writes per second to the database log, Test 3 performs an average of 77 writes per second to the database log, and Test 4 performs and average of 79 writes per second to the database log.

A.7 Network Bytes Transmitted per Second

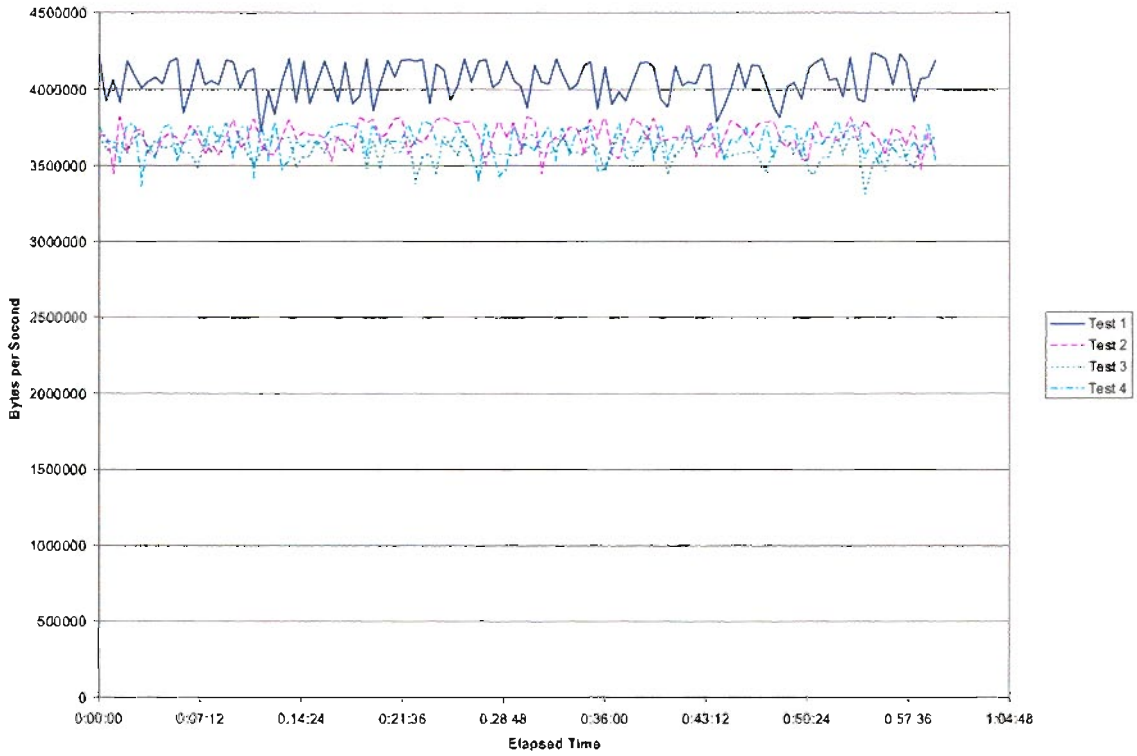

Figure A-13: Total Network Bytes Transmitted per Second

Figure A-13 displays the amount of data transmitted over the network between the RBE and the Web server for each test. Each test displays results that are steady and consistent. The Web server in Test 1 transmits an average of 4.1 million bytes per second, Test 2 transmits an average of 3.7 million bytes per second, Test 3 transmits an average of 3.6 million bytes per second, and Test 4 transmits an average of 3.7 million bytes per second.

A.8 Web Server System Available Memory

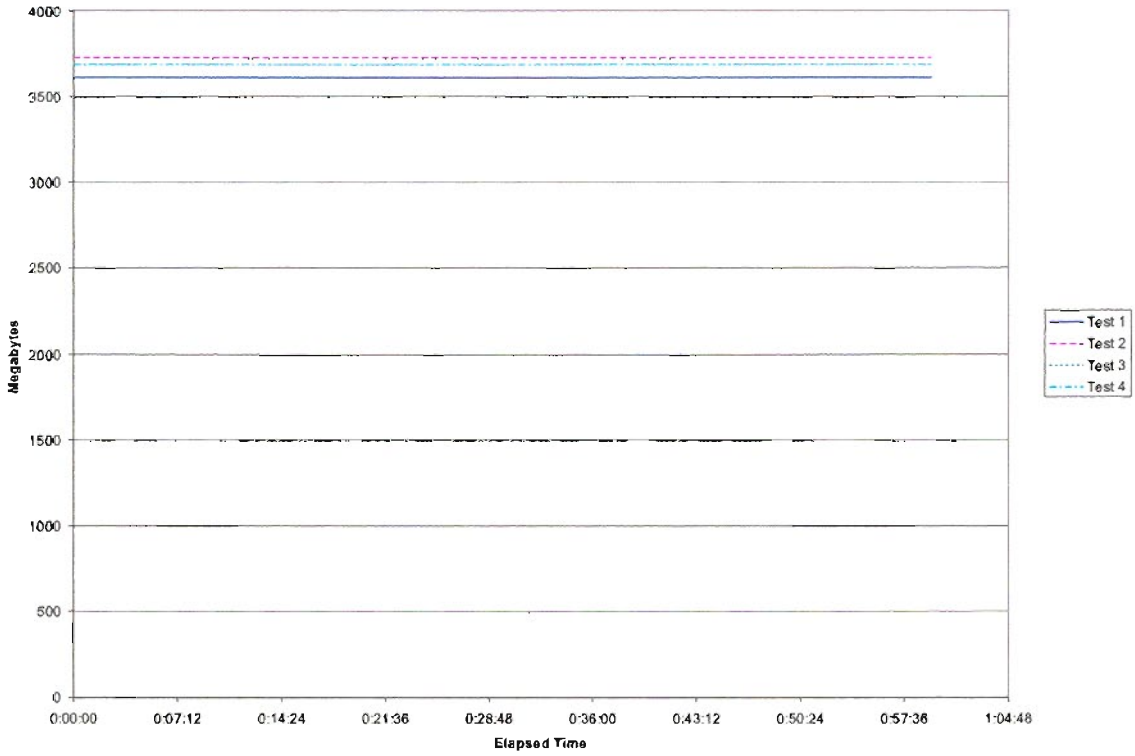

Figure A-14: Web Server System Available Memory

Figure A-14 displays the available memory on the system running the Web server in Test 1, Test 2, Test 3, and Test 4. The purpose of this graph is to verify the queues in the Web application, as shown in Figure 3-3 are not growing without bound. The nearly constant amount of available memory shows that the size of the queues in the Web application is maintaining a constant size throughout the duration of each test. This shows that the Web application is able to insert clickstream data into the clickstream database as it is generated.

A.9 Web Server Log Growth

The data gathered from the hard disk drives verify that there is enough hard disk drive storage for the Web server log and database log to grow over the duration of a test. Table A-1 lists each test and the calculated rate of growth for the Web server log, the HTTP_LOG table, and the database log based on the number of bytes written per second. The partition used for the Web server log has a maximum capacity of 90GB of storage,

the partition used for the HTTP_LOG table also has a maximum capacity of 90GB of storage, and the partition used for the database log has a maximum capacity of 45GB of storage. The partitions created for each of the columns listed in Table A-1 may not have been divided evenly as the partition for the Web log has enough space for 180 hours of data, the partition for the HTTP_LOG table has enough space for 180 hours of data, and the partition for the database log has enough space for 70 hours of data, but there is definitely enough hard disk drive storage available for running each test several hours.

	Web Log Growth (MB per Hour)	**HTTP_LOG Table Growth (MB per Hour)**	**Database Log Growth (MB per Hour)**
Test 1	511	N/A	N/A
Test 2	467	556	1281
Test 3	457	543	1175
Test 4	N/A	551	1202

Table A-1: Growth of the Web Server Log and the Clickstream Database

Biographical Note

Mark Wong was born in Portland, Oregon, on March 4, 1977. He received his Bachelor of Science in Civil Engineering from Tufts University in May 1999. He passed the Fundamentals of Engineering (EIT) exam in 1998. Since graduating from high school in 1995, Mark has interned at Sequent Computer Systems in Beaverton, Oregon, which merged with IBM Corporation in 1999. Mark has also interned at Informix Corporation in Portland, Oregon, for the division that was purchased by IBM Corporation in 2001.