# Characterizing the Scalability of a Large Web-based Shopping System

**Martin Arlitt**
**Diwakar Krishnamurthy**
**Jerry Rolia**

Hewlett-Packard Laboratories
1501 Page Mill Road
Palo Alto, CA 94304
{arlitt, diwa, jar}@hpl.hp.com

This paper presents an analysis of five days of workload data from a large Web-based shopping system. The multi-tier environment of this Web-based shopping system includes Web servers, application servers, database servers and an assortment of load-balancing and firewall appliances. We characterize user requests and sessions and determine their impact on system performance and scalability. The purpose of our study is to assess scalability and support capacity planning exercises for the multi-tier system. We find that horizontal scalability is not always an adequate mechanism for supporting increased workloads and that personalization and robots can have a significant impact on system scalability.

# 1 INTRODUCTION

Over the past several years interest in e-commerce (i.e., the use of the Internet to buy and sell goods and services) has grown substantially. While e-commerce itself is relatively new, the underlying business principles for its success are not. In order to be successful, a business must not only attract new customers but retain existing ones, as profits rise with extended customer relationships [21]. In fact, retaining just 5% more customers has been shown to increase profits by up to 100% [21]. This result suggests that businesses should do whatever they can to retain their customers.

The fundamental motivation for this work is the need to ensure a profitable business. Reichheld and Sasser state that quality is the most profitable way to run a business, and that in order to improve quality it must first be measured [21]. Unfortunately there are many aspects of a business that must be evaluated in order to measure quality. For example, a business must market itself effectively to attract (and retain) customers. A business must ensure that it has the products or services that its customers are looking for at acceptable prices and in sufficient quantities [14]. A business must find ways to “know” who its customers are in order to provide them with better (i.e., personalized) service [21]. To retain customers a business must also provide a pleasurable shopping experience (e.g., no long delays to make a purchase, ease and promptness in finding goods or services, etc.). In this paper we are primarily concerned with the last of these issues, that is, providing a pleasurable shopping experience. The others, while vital business practices, are (mostly) beyond the scope of this paper.

Our purpose is to investigate the issues affecting the performance and scalability of a large Web-based shopping system. Workload data was obtained from an e-commerce site. Workload characterization and clustering techniques are used to determine the impact of user requests and sessions on system performance, to assess system scalability, and to support capacity planning exercises for the multi-tier system.

Our research has identified several key results. First, there are tradeoffs among business issues that must be considered. In particular, providing personalized service to customers can severely impact the shopping experience provided to customers, as personalization may reduce the performance and scalability of the system. Thus it may be necessary for the system to adapt to its workload, providing more personalization when relatively few customers are using the system and less (or no) personalization when the system is supporting many concurrent customers. Second, we confirm that, as in other studies [2][18], a non-negligible fraction of requests are issued by robots (i.e., non-human users). We find that robots can have a significant impact on system performance. Steps should be taken to positively identify robots so that their presence does not degrade the service provided to customers. Finally, caching is vital for ensuring the scalability of large Web-based shopping systems, even though much of the workload may appear to be inherently uncacheable.

The remainder of this paper is organized as follows. Section 2 provides additional background information and introduces related work. Section 3 describes the system under study, while Section 4 presents the measurement data used in our work. Section 5 discusses the results of our (HTTP-level) workload characterization. Classes of requests are characterized in Section 6, which also explores the impact of workload changes and personalization on system scalability. Section 7 characterizes classes of user (i.e., human customers) and robot sessions for the purpose of capacity planning. A summary of performance and scalability issues is given in Section 8. Concluding remarks are in Section 9.

# 2 Background Information and Related Work

In the previous section we identified a number of issues (e.g., personalized service, a pleasurable shopping experience) that must be addressed in order to ensure a successful (i.e., profitable) business. Unfortunately, tradeoffs among many of these issues must be considered. For example, successful promotions can cause dramatic increases in the number of customers that visit a site. If system capacity is inadequate, system performance can degrade substantially, leading to a loss of customers. Poor site responsiveness (i.e., long response times) can also result in a loss of customers and revenue to competitor sites. The performance and scalability of a site are therefore vitally important issues that must be carefully managed to avoid blunders that risk a business's success.

Capacity management for Web-based shopping systems is governed by the notion of horizontal scalability. That is, as the number of concurrent users that the system must support increases, more servers are added. Such scaling must be cost effective, otherwise it will not be possible for a system to support the large number of users that can be attracted by a successful promotion. In order to improve the performance and scalability of e-commerce systems, a thorough understanding of the system workload is required. Numerous characterization studies of Web server workloads have been conducted, including [3], [4], and more recently, [20]. Due to the limited availability of data, few characterization studies of e-commerce workloads have been done. Menascé *et al.* performed one of the first [18]. Our work builds on the characterization studies of Web server workloads and extends the current knowledge of e-commerce workloads.

E-commerce provides businesses with a great opportunity to quantify the quality they provide their customers, since the underlying use of computer systems enables businesses to collect the data necessary for analysis. Lee and Podlaseck utilized data from online shopping systems to develop tools for improving Web marketing and merchandising [14]. Many Web-based shopping sites today are interested in personalizing online shopping as a means of differentiating themselves from their competitors. VanderMeer *et al.* [22] propose a method for improving the scalability of personalization systems. Our work is complementary. Challenger *et al*. developed an approach for consistently caching dynamic Web data that became a critical component of the 1998 Olympic Winter Games Web site [7].

# 3 THE SYSTEM UNDER STUDY

The Web-based shopping system under study has a multi-tier architecture typical of e-commerce sites. The tiers of this system are illustrated in Figure 1. All customer requests access the system via the Internet. Load balancers divide incoming requests across a tier of Web servers. The Web servers serve all requests for non-dynamic resources and propagate requests for dynamic resources to application servers. A dynamic resource is one that is typically created by executing a program or script each time the resource is requested. Application servers make use of database servers to support information retrieval, personalization, and transactional order management. The application servers prepare responses in HTML format for rendering on users' browsers. These responses are then propagated to the customer via the Web server tier. Networking equipment such as routers and firewalls are not shown in Figure 1.

The system under study handles only the requests for the HTML portions of each Web page. All requests for the embedded graphics in each page are provided by a separate "graphics" system. No workload data on this graphics system is available; as a result, this system is not analyzed in this paper. Application servers can either generate HTML responses for users dynamically or serve responses from an application server request cache (each application server has its own request cache, as shown in Figure 1). Dynamically generating responses creates the opportunity for personalization, but can use significantly greater processing resources than serving responses from a request cache. We explore the performance and scalability impact of application server request caching in this paper.

A number of our analyses in Section 7 characterize user *sessions*. We define a session as a sequence of requests made by a single "source" during a visit to the site [17]. For the system under study each session is associated with a unique identifier in order to maintain state (e.g., shopping cart contents) between requests. The first request of a session does not have a session identifier - we define it as a *session id-less request*. A Web server that receives a session id-less request uses a load-balancing algorithm to choose an application server. It then forwards the request to the application server, which assigns a session identifier. Subsequent requests in the session may go to any Web server but are always directed back towards the application server that issued the session identifier.

In this paper we consider initial session id-less requests to be part of their sessions. However it is not possible to positively identify the access log entry of the initial request of all distinct sessions. Although the Web access logs do contain the IP address for the client that issued each request, there are a significant number of proxies present which may represent any number of users.

When characterizing session lengths, we compensate for the initial session id-less request by simply adding one to all reported session lengths. We assume that the time between this initial request and the second request in the session (i.e., the initial "inter-request" time) is equal to the mean inter-request time across all sessions. An analysis determined that these start of session requests accounted for most of the session id-less requests.

Finally, to conserve system resources, application servers time-out sessions (and reclaim their resources) after 15 minutes of inactivity. All requests associated with an expired session identifier receive a ‘time-out’ response. At this point the user may return to the home page to obtain a new session identifier, or they may choose to depart.

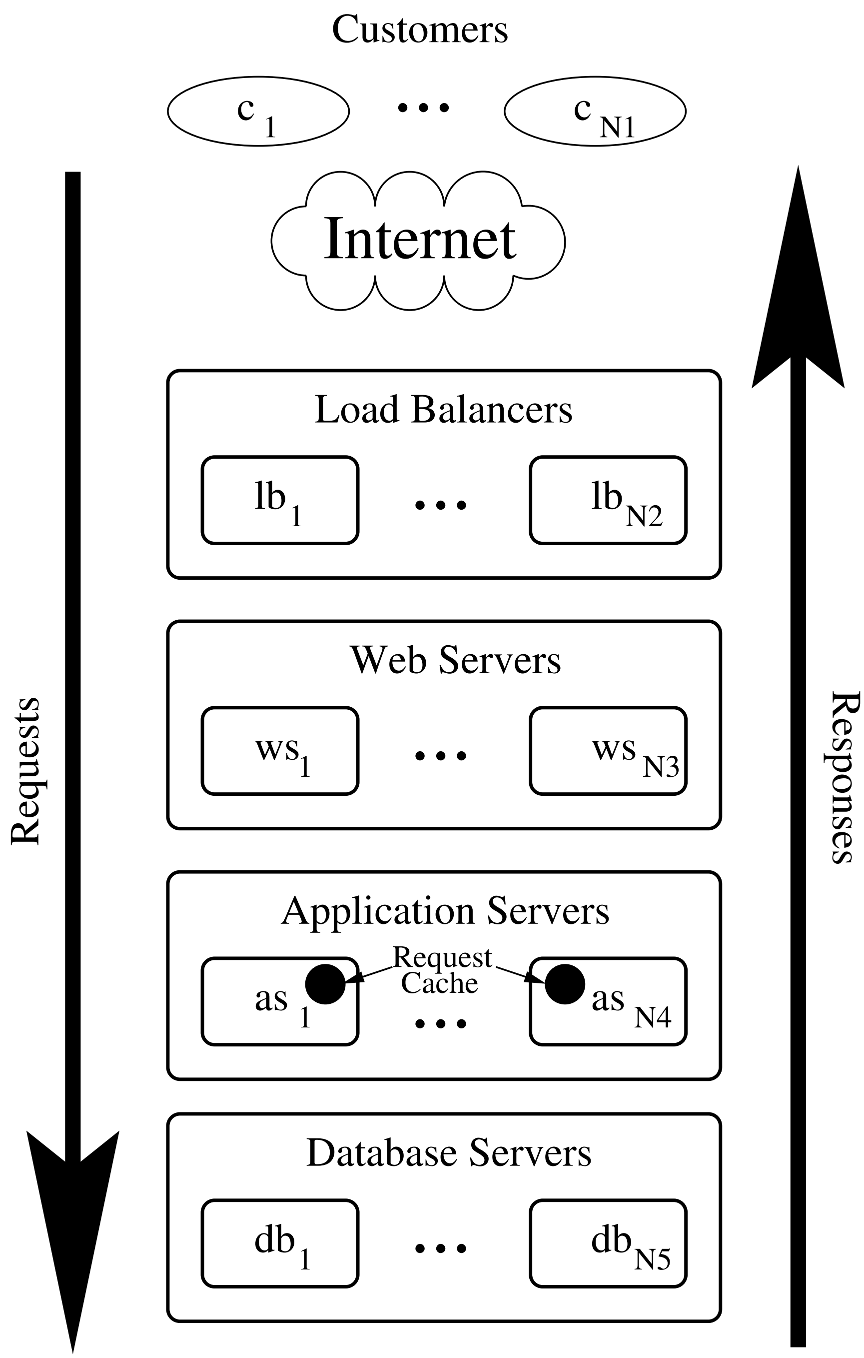


Figure 1 Logical Diagram of the System under Study

# 4 MEASUREMENT DATA

The measurement data for this study was collected from four different sources. Three of these sources were part of the site under study. The primary source of workload data were the access logs collected from each of the Web servers in the site. Additional workload information was provided by the access logs of each of the application servers. Supplemental performance data for servers was available from MeasureWare[1] logs. These logs were available for all of the servers in the site.

The fourth source of data was a test environment that was a replica of the actual site, albeit on a much smaller scale. This test environment enabled us to further characterize the performance of various requests. For example, it was possible to obtain estimates for the mean CPU demands of important requests using exercisers (i.e., mini-benchmarks). Unless explicitly stated we do not use this data when reporting how the actual site was used.

Measurement data was collected from the actual site on two separate occasions. The first measurement period was on March 9$^{th}$, 2000 (the *March data set*); the second measurement period lasted from midnight on July 21$^{st}$, 2000 until midnight on July 25$^{th}$, 2000 (the *July data set*). There were no significant changes to either the software or hardware configurations of the system under study between the two measurement periods.

The Web server access logs contain a single line of information for each HTTP request issued to the site. The recorded information includes: the IP address of the client that issued the request; the time the request was received (1 second resolution); the method

1. MeasureWare is also known as the Vantage Point Performance Agent [23].

(e.g., GET) to be applied to the resource; the name of the resource (i.e., the URL); a session identifier (for sessions already in progress); the HTTP protocol version; and the HTTP response code. The Web server logs are used in the (HTTP-level) workload characterization described in Section 5 and the characterization of sessions presented in Section 7.

The application server access logs contain a single line of information for each request handled by the application server. This information includes: the timestamp of the response completion; the elapsed time for completing the response; a flag to indicate if the response was served from the application-level cache; and the (application-level) resource name. These logs are used to calculate CPU demands in Section 6.

The MeasureWare logs provide a wide range of data on server performance. For this study we primarily utilized “global” metrics (aggregated over five minute intervals). The data we used includes: a timestamp for the start of the measurement interval; the mean CPU utilization over all server CPU; the peak disk utilization across all disks; the physical memory utilization; and the number of packets sent and received on all network interfaces. Information from these logs was used in Section 6 to compute CPU demands.

Despite the vast amount of data collected, a lot of interesting information is still missing. For example, the Web server logs do not contain any information on the size of request or response headers or bodies, nor do they include a measure of the time required to complete a response. In addition, the timestamps for request arrivals have only a one second timestamp resolution, which limits the accuracy of several of our analyses. The application server access logs do not include a session identifier that would allow us to match the requests in the Web server access logs to those in the application server access logs.

The collected data contains information that can be used to evaluate the site's effectiveness from many different perspectives. In this paper our focus is on the performance and scalability of the site. Other important perspectives, such as marketing and merchandising [14] are orthogonal to the issues that we pursue in this paper. In addition, we do not report any absolute values (e.g., the number of requests during the measurement period) that could be used to derive information on the site's effectiveness in terms of marketing and merchandising (e.g., the number of customers visiting the site or the number of purchases made during the measurement period).

# 5 WORKLOAD CHARACTERIZATION

This section presents the results of our (HTTP-level) workload characterization. In our effort to identify methods for improving scalability, we focus on the most significant characteristics. Section 5.1 discusses the distribution of requests by resource type. Section 5.2 analyzes the usage of the site during the measurement periods. Section 5.3 looks at the resource referencing patterns. Section 5.4 describes client request behaviours.

## 5.1 Resource Types

Table 1 provides the breakdown of requests by the resource type. Two approaches are utilized to determine the resource type. One approach examines the extension of the resource. For example, resources ending in '`.jpg`' or '`.gif`' are placed in the Image cate-

**Table 1 Breakdown of Requests by Resource Type**

| Resource Type | March 2000 (%) | July 2000 (%) |
|---|---|---|
| Dynamic | 96.58 | 95.73 |
| HTML | 2.58 | 3.67 |
| Image | 0.37 | 0.16 |
| Text | 0.03 | 0.01 |
| Other | 0.44 | 0.43 |
| **Total** | **100.00** | **100.00** |

gory. The second approach searches for the presence of a parameter list (e.g., `/example.html?parameter_list`). All requests containing a parameter list are placed in the dynamic category.

Table 1 reveals that almost all requests (over 95% in both data sets) were for dynamic resources. Even after accounting for the absence of graphics, this represents a significant difference between this e-commerce workload and 'traditional' Web server workloads. As we will see in Section 6, the on-demand creation of dynamic resources can have a significant impact on system scalability. In the remainder of the paper, when analyzing system workload, we consider only the requests for dynamic resources.

## 5.2 Usage Analysis

Our next analysis examines the usage of the site during each of the two measurement periods. Understanding site usage is important for numerous reasons, including administrating the site (e.g., scheduling backups or upgrades so that there is minimum impact on the customer experience). Monitoring site usage over time is important for planning capacity.

Figure 2 shows the usage of the site during each of the two measurement periods. The curve in each graph represents the request rate to the site during each minute of the measurement period. The request rate has been normalized by the mean request rate of the measurement period. Figure 2(a) shows the site usage for a single day in March. This figure indicates that the workload follows a typical time-of-day pattern; i.e., the site is busiest during the day and least busy during the early morning. The peak request rate during this measurement period is twice the mean request rate, and almost nine times the lightest load, which occurred between 3am and 6am. During this measurement period there were a number of occasions when the request rate appears to decrease rapidly for a short period of time. All of these occurrences were followed by a sudden increase in the request rate. We do not know if this behaviour was due to a problem with the system or a problem with the collection of the workload data.

Figure 2(b) shows the usage of the site during the July measurement period. This figure reveals that in addition to the time-of-day factors, the day of the week also affects the request rate to the site. For example, the site is busier on weekdays than on weekends. On weekdays the workload is busier during the day than the evening, which is still significantly busier than the early morning. On Saturday the request rate is quite consistent throughout the day and evening hours. On Sunday the evening hours are busier than the daytime hours. In all cases, the lightest loads occur in the early morning hours. In Figure 2(b) the peak request rate is more than twice the mean request rate and nine times the minimum request rate. While both of our data sets are quite consistent in terms of the peak to mean ratios, it is important to note that there were no special events during either of these periods that would have caused a “flash crowd” (i.e., a sudden and significant increase in the number of user arrivals). Examples of flash crowds can be seen in the 1998 World Cup Web site workload [3]. The flash crowds in that data set were caused by user interest in several of the key football (i.e., soccer) matches. Examples of special events that can affect e-commerce workloads are new advertising campaigns, special promotions (e.g., issuing coupons), or the approach of holidays such as Valentine’s Day, Easter, Mother’s Day, Father’s Day and Christmas.

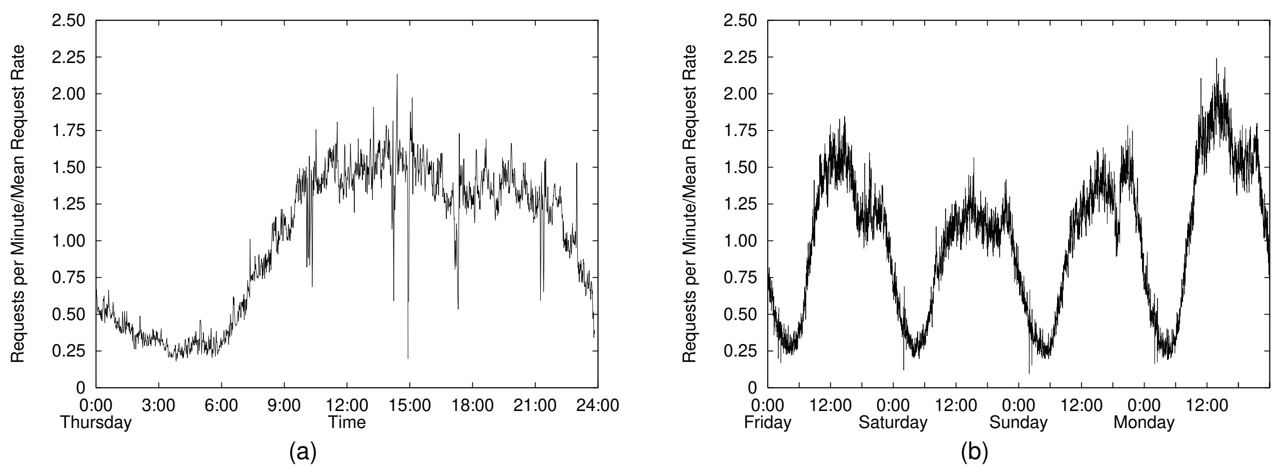


Figure 2 Traffic Volume: (a) March 2000 (b) July 2000

As we have already described, the request rate during the two measurement periods varied by up to a factor of nine indicating that at times the system has excess capacity. This suggests that the system could be enhanced to provide more personalized service to the smaller number of customers visiting the site during these periods. This additional customer service could be a feature for improving customer loyalty, and ultimately increasing profits.

## 5.3 Resource Referencing Behaviour

In this section we analyze the referencing behaviour to the unique resources (i.e., the distinct URLs, including the name of the resource and any parameters) requested during a given measurement period. In particular, we examine the popularity of resources for this site.

In several studies, the popularity of resources on the Web has been characterized by a Zipf-like distribution [4][5][6][8]. A distribution is considered to be Zipf-like if the relative probability of a request for the $\mathbf{i^{th}}$ most popular resource is proportional to $\mathbf{1/i^{\beta}}$ [6]. The more concentrated the references are to a set of popular resources, the greater the value of $\beta$. To test if a distribution is Zipf-like, a log-transformed plot of the number of references for each resource as a function of the rank of the resource is created. The most frequently referenced resource is assigned a rank of 1; the $N^{th}$ most frequently referenced resource is assigned a rank of $N$. If the distribution is Zipf-like, the plot should appear linear with a slope near -$\beta$ [5].

Figure 3(a) and (b) show the relative popularity of the most referenced resources in the March and July data sets respectively. These plots indicate that the most popular files do follow a Zipf-like distribution. The estimates of the slope are $\beta$=1.06 for the March data set (Figure 3(a)) and $\beta$=1.04 for the July data set (Figure 3(b)). In both cases the coefficient of determination ($R^2$) is 0.99, which indicates a very good fit between the empirical and synthetic distributions.

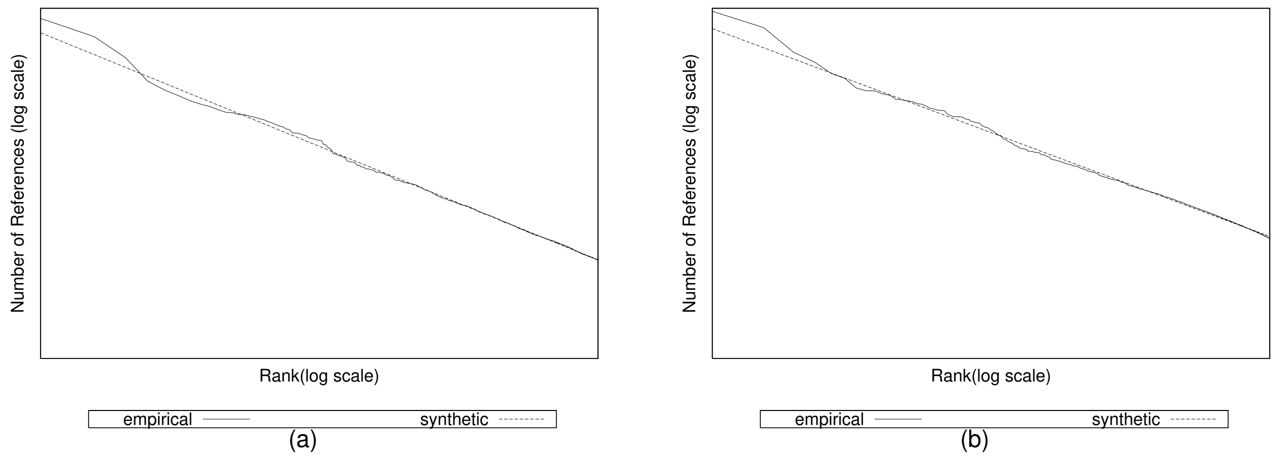


Figure 3 Concentration of Resource References (reference count vs. rank): (a) March 2000 (b) July 2000

These results suggest that even a small cache could significantly improve system performance, as a small number of resources account for a significant fraction of all requests. In the case where a small cache is utilized, an intelligent replacement policy should be used to maximize performance. Such a replacement policy would need to exploit workload characteristics such as resource popularity in order to achieve the best possible performance. For reasons explained in Section 8, we discuss utilizing a cache large enough to store all of a site's responses; a side effect of this approach is that it negates the need for a replacement policy. As a result, we do not investigate other resource referencing characteristics (such as temporal locality) that could be useful in designing/selecting good replacement policies.

## 5.4 Client Request Behaviour

In this section we analyze the requests (or references) by the client IP address, rather than by the name of the requested resource. Figure 4 indicates the relative popularity of the clients based on the number of requests issued by each unique client. The results show that the popularity of clients also follows a Zipf-like distribution. In the March data set the slope is estimated at $\beta$=0.66 (Figure 4(a)); in the July data set (Figure 4(b)) the slope is estimated at $\beta$=0.78.

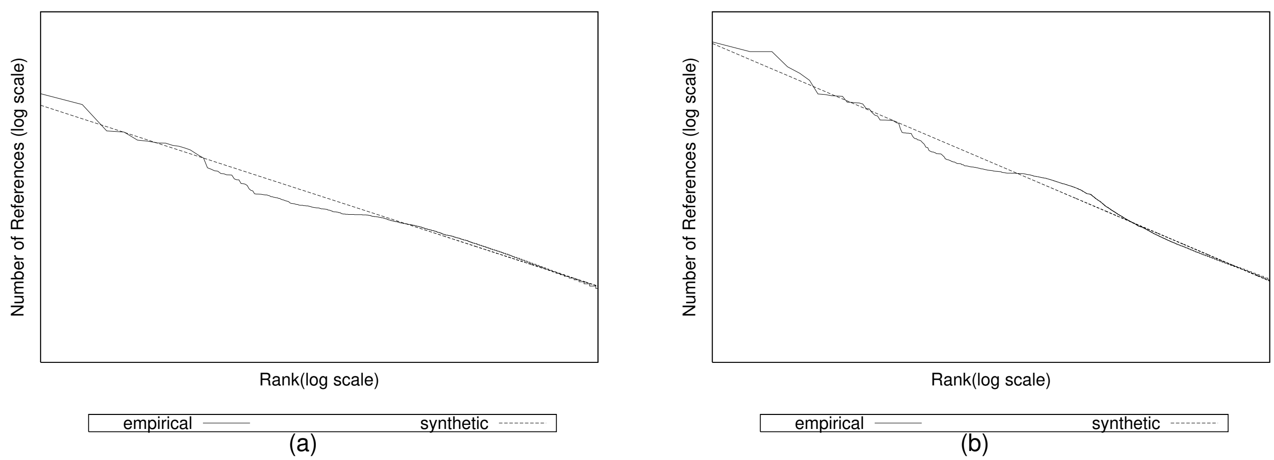


Figure 4 Concentration of Requests from Clients (request count vs. rank): (a) March 2000 (b) July 2000

These results reveal that some clients issue significantly more requests than others. Many of the more popular clients are actually proxy servers that forward requests on behalf of a number of customers. This verifies that caching at a small number of locations within the Internet would be an effective enhancement for improving the scalability of this system. We discuss this issue in more depth in Section 8. A second group of popular clients are robots. We discuss robots and their impact on the system in more detail in Section 7.

# 6 Classes of Requests

In this section we characterize classes of requests based on the impact of their performance on the system. We then discuss the impact of these request classes on system scalability. For the system under study, nearly all requests are for dynamic resources (not HTML files) and pass from the Web server tier to the application tier for processing. Application servers in the system under study support many kinds of requests. However, there are essentially three classes of requests, distinguished by their significantly different resource demands. We partition the requests into these classes to better support reasoning about the scalability of the system.

The three classes of requests are *cacheable*, *non-cacheable*, and *search*. As examples, cacheable requests return responses that describe products for sale. Non-cacheable requests include such functions as adding an item to a shopping cart and preparing order details. Search requests let users search for products by keyword. Non-cacheable and search requests always cause dynamic page generations[2]. Cacheable requests, as the name implies, can have their responses stored in an application server request cache. Subsequent requests for the same response can be served from the cache.

2. The caching of search responses is not uncommon in these environments but was not exploited during this study.

A drawback of a cached response is that it must not contain any personalized information. As discussed in the Introduction, personalization is desirable to boost repeat visits and increase revenue [21]. Personalization is therefore desirable for Web-based shopping sites. Requests for cacheable documents that are not in the cache or that are personalized cause the dynamic generation of responses. We define these as *request cache misses*. They are different from *uncacheable requests* (e.g., requests in the non-cacheable and search classes) which are never cached.

Table 2 lists the request classes and gives the relative mean CPU demand for each. CPU demand does not include the time spent waiting for input-output operations or any other remote access; it is time spent consuming CPU resources alone. These demand values are estimated based on the use of exercisers and MeasureWare CPU demand reports from within the testbed environment. The reported values were verified with respect to elapsed time measurements for the system under study during low load conditions.

**Table 2 Relative Mean CPU Demands for Request Classes**

| Request class name | Ratio of CPU demand |
|---|---:|
| Cacheable (response can be cached) | 1 (cache hit) |
| Cacheable (response can be cached) | 500 (cache miss) |
| Non-Cacheable (response can not be cached) | 100 |
| Search | 40 |

For the system under study, the application server CPUs presented a system bottleneck. The wide range of ratios shown in Table 2 suggest that the request mix has a significant impact on the average application server CPU demand per request. For the system under study, generating a cacheable response dynamically requires approximately 500 times the application server CPU demand of serving a response from the request cache. This is due to the use of interpreted scripts, and illustrates the performance cost of personalization and other request cache misses. Mechanisms that integrate fragments of cached pages during personalization are also possible and may have lower costs, but they were not exploited by the system under study [7].

The remainder of this section characterizes the request classes for our data sets and demonstrates the sensitivity of system scalability to request class mix and request cache hit rate. Request class mix defines the fraction of total requests, over some time interval, that belong to each class. Table 3 gives a breakdown of requests by request class for the March and July data. The table shows that the request class mix remained relatively stable over the five month period.

**Table 3 Request Class Mix**

| Request class name | March 2000 (%) | July 2000 (%) |
|---|---:|---:|
| Cacheable (response can be cached) | 73.30 | 70.53 |
| Non-Cacheable (response can not be cached) | 15.89 | 17.94 |
| Search | 10.81 | 11.94 |

A simple utilization analysis demonstrates the sensitivity of application server capacity to the mix of requests for the three classes. Utilization analysis exploits the utilization law [11] that states:

$$U = X\,D \qquad \text{(EQ 1)}$$

where $U$ is the utilization of a resource, $X$ is the number of completions per time unit, and $D$ is the demand in time units per completion. Since $U$ has a maximum value of 1 per resource the maximum value for $X$ is determined by $D$. As $D$ increases, the maximum number of completions per time unit decreases. Conversely, as $D$ decreases, the maximum number of completions per time unit increases. Calculating the ratio of demand for different planning scenarios quantifies the change in system capacity:

$$\textit{Change in capacity} = D_{old} / D_{new} \qquad \text{(EQ 2)}$$

As an example, suppose the system has the request class mix for the July data in Table 3. Based on the product of this mix (weights) and the ratios of average application server CPU demands ($D$) of Table 2, the weighted average CPU demand $D$ per request is 23.4 times that of a cacheable request. If the percentage of cacheable requests varies by ±5% with the differences applied equally to non-cacheable and search then the average CPU demand per request for the workload varies from 20.0 to 26.9 times that of a cacheable request. Therefore, depending on the perturbation in request class mix, the capacity of each application server varies by a factor of 1.17 to 0.87.

Next suppose that the request cache is not large enough to cache all possible responses or that the frequent personalization of responses is required. Suppose again we have the workload mix of the July data from Table 3. We perturb the workload mix of Table 3 by considering a 1%, 4% and 8% increase in response cache misses to reflect increased personalization. The weighted average CPU demand $D$ per request for these perturbation cases are, respectively, 28.4, 43.4 and 63.3 times that of the average CPU demand of a request satisfied by the cache. For these cases the relative capacity per server decreases from 1 (for no misses), to 0.82, 0.54, and 0.37, respectively.

For large numbers of users an 8% request cache miss rate combined with a 5% decrease in cacheable requests leads to very high demand. Approximately 2.7 times as many servers are required compared to the scenario of no request cache misses. This is a concern when exploiting horizontal scalability. Even small changes in the cache hit rate or the workload mix can have a significant effect on the number of servers needed for horizontal scalability under high load conditions.

# 7 SESSION ANALYSIS

The previous section contained an analysis of individual requests. This section presents a session-level characterization of the system under study. Section 7.1 introduces issues that pertain to the two kinds of sources that make use of the system, namely users and robots. Section 7.2 discusses the session-level characteristics for the March and July data sets. Section 7.3 applies clustering techniques to categorize the individual sessions based on their performance impact on the system to support the evaluation of system scalability. The resulting clusters are used to create workload classes appropriate for multi-class queuing models.

### 7.1 Users and Robots

The primary source of requests for large Web-based systems (such as the one under study) are users. Robots are another source of requests. Common uses of robots include determining a site's liveliness, measuring a site's responsiveness, and collecting a site's pricing information. It is important to distinguish these sources of requests from those of users when reporting on user behaviour and when assessing scalability.

It is difficult to positively distinguish between robots and consumer sessions. While proper 'netiquette' requires robots to access the resource '/robots.txt' to identify themselves to the site as being a robot (and to learn about the site rules for robots) [13], our results suggest that most robots do not access this file. While some robot implementors/users may be unaware of this expected behaviour, we speculate that many choose not to access this resource because numerous sites refuse (further) service to positively identified robots. A few robots can be identified by the user-agent information they provide, while others can be identified from their fully qualified domain name. Unfortunately, these methods do not allow us to identify many of the (suspected) robots. Thus, for this work, we utilize the overly-simple assumption that robot sessions issue significantly more requests than do typical human users. Sessions consisting of more than 30 requests are deemed to be generated by a robot; shorter sessions are deemed to be human users. A detailed analysis found that 9% of those sessions characterized as robots performed checkout operations. This offers a lower bound on false positives for our simple approach.

## 7.2 Descriptive Statistics for Sessions

In this section we analyze two characteristics for both user and robot sessions using the March and July data sets. Section 7.2.1 examines the inter-request time distribution, while Section 7.2.2 analyzes the number of requests per session. These characteristics are measures of the workload intensity.

As mentioned in Section 4, the Web server (typically) records a unique session identifier for each request handled by the server. We utilize this unique identifier to reconstruct and analyze the distinct sessions. Sessions seen in the first or last hour of the (aggregated) Web server access log are discarded from our analyses to reduce the impact of end effects.

We assume a session time-out length of 15 minutes in our analyses. This is the same value that is used by the system under study. We used a time-out value in our analyses in order to prevent a few, long inter-request times from skewing our results (e.g., the mean inter-request time). Many of these long inter-request times are caused by robots that restart their "crawl" of the site. There are a number of sessions that get fragmented as a result of using this time-out value in our analyses. Typically, the fragmented portion of the session consists of only a single request because the application server issues a response requesting that users restart their sessions (by clicking on a link to the home page, thereby acquiring a new session identifier). Thus we ignore sessions of length one.

### 7.2.1 Inter-Request Times within Sessions

In this section we analyze the session inter-request times (i.e., the time between subsequent request arrivals within a session). The inter-request times for each unique session are measured, and the results are then aggregated for all user and all robot sessions, respectively.

Figure 5 shows the inter-request time distribution for both users and robots in March (Figure 5(a)) and July (Figure 5(b)). The x axis in Figure 5 uses a log scale in order to show the full range of values for the inter-request times. In both measurement periods the results are quite similar. For example, in March the mean inter-request time for users was 53.9 seconds and the median (i.e., the 50th percentile) was 28 seconds. In July the values were slightly lower, with a mean of 48.2 seconds and a median of 26 seconds. The inter-request times for robots were typically shorter than for users. In March the mean inter-request time for robots was 38.9 seconds, while the median was 22 seconds. In July the results were almost identical, with a mean of 40.4 seconds and a median of 22 seconds. In all cases the maximum inter-request time was limited to 900 seconds.

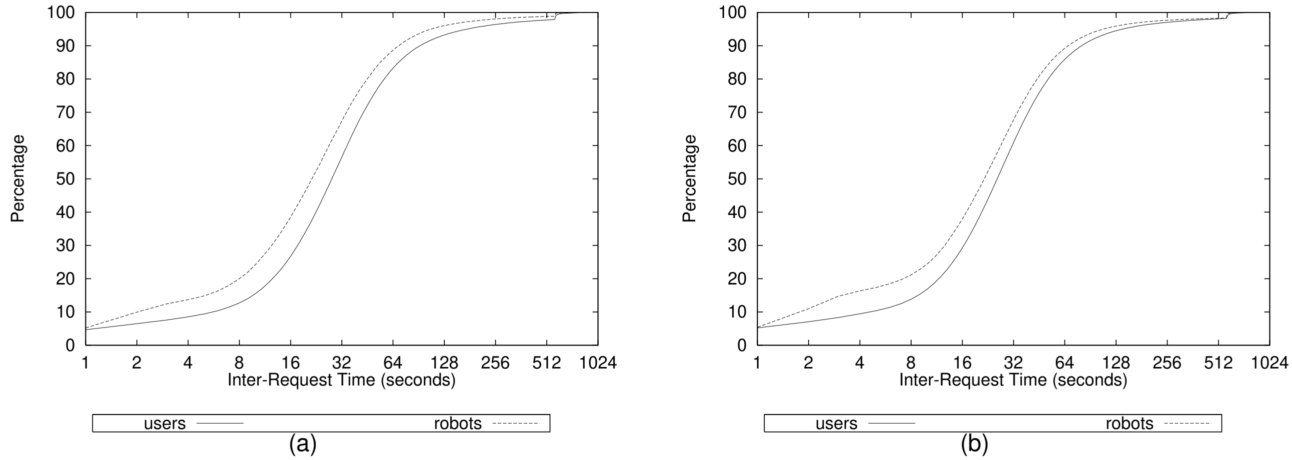


Figure 5 Cumulative Distribution of Session Inter-Request Times: (a) March 2000 (b) July 2000

### 7.2.2 Number of Requests Per Session

Figure 6 shows the distribution of the number of requests per session for both users and robots. In both Figure 6(a) (March data set) and Figure 6(b) (July data set) the x-axis uses a log scale in order to show the full range of values. The maximum number of requests in a user session is limited to 30 due to how we chose to distinguish between users and robots.

Figure 6 reveals that there are a few robot sessions with fewer than 30 requests. This occurs for the following reasons. We began our analysis by dividing all of the unique sessions into two sets (users and robots) based on the number of requests that contained each distinct session identifier. Sessions with more than 30 requests were assigned to the set of robot sessions. We then analyzed the characteristics of all sessions in a particular set, this time utilizing the 900 second limitation on time between subsequent requests in a session. As a result of this approach, a single session can become fragmented. With a well-behaved client the fragment will always be of length one, as the response would direct the client to restart the session and obtain a new identifier. However, some robots are not well behaved, and instead continue to reuse an expired session identifier, even though the application server continually informs the client to restart the session and acquire a new session identifier. This is an example of a robot consuming resources at the site while not performing any useful service for the robot operator, as the response simply contains information on how to acquire a new session identifier.

Figure 6 shows that the number of requests per session for both users and robots is quite consistent across the March and July measurement periods. In March the average number of requests for a user session was 8.0 and the median 6.0, while in July the mean was 7.14 and the median 6.0. For robot sessions the mean number of requests per session was 51.0 in March and 52.6 in July, while the median in both data sets was 42.0. In March the maximum number of requests in a robot session was 473; in July it was 3,312 (recall that the measurement period in July was 4 times longer than in March).

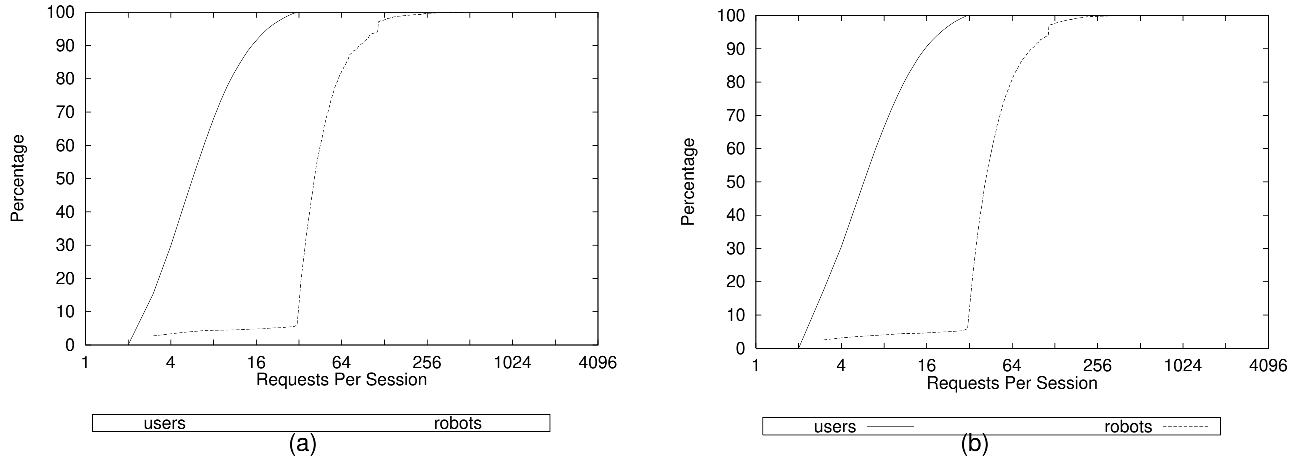


Figure 6 Cumulative Distribution of the Number of Requests per Session: (a) March 2000 (b) July 2000

## 7.3 Classes of Users and Robots

The previous section characterized user and robot sessions as a whole. In this section we use clustering techniques to describe classes of users and robots. The analyses presented in this section employ the general approach described in [16]. For our study we are primarily interested in scalability. Thus we partition the sessions based on their performance impact on the system. For example, those sessions that cause more non-cacheable requests are distinguished from those that issue cacheable requests only, since they have much higher application server resource demands. Clustering techniques [12][10] are used to determine a set of workload classes. Characterizing workload mix based on classes of users and robots enables precise capacity planning and scalability analysis exercises.

Clustering [12] is a well-known technique. A basic clustering algorithm systematically merges clusters that begin with one observation each into other clusters, until only a small number of clusters remain. The merging decision is based on a distance metric that establishes the closeness between cluster pairs. The choice of distance metric and method for choosing a representative observation from a cluster for use with the distance metric are decided based on the problem at hand. The space and computation complexity are $\mathbf{O}(N^2)$ where $N$ is the number of observations. For our study sessions are observations and resulting clusters are workload classes.

The k-means clustering algorithm [10] is more efficient than the basic clustering algorithm. The k-means algorithm is practical for dealing with the large numbers of sessions (observations) typical of e-commerce systems. It is an iterative method that creates k clusters (workload classes), and has a space complexity of $\mathbf{O}$(N) and a per-iteration time complexity of $\mathbf{O}$(k N). For a cluster to truly represent its members, an appropriate value for $k$ must be chosen. Observations within a cluster should indeed be similar to one another, while the clusters identified are as dissimilar as possible. $\beta_{cv}$ is a measure of quality for a clustering exercise [16]. It is defined as the ratio of the coefficient of variation of the mean intra-cluster distance divided by the coefficient of variation of the mean inter-cluster distance. The variation in the mean intra-cluster distance should be small and the variation in the mean inter-cluster distance should be large. In general we look for a small number of clusters ($k$=2 or more) with a small $\beta_{cv}$. When reporting results we provide the $\beta_{cv}$ value for the chosen value of $k$ and the minimum value when considering $k$ from three to twelve.

To apply the clustering algorithm, a distance metric must also be chosen. Each session is mapped onto a vector that describes its use of system features. We use a Euclidean distance measure to determine the closeness between cluster pairs. A first approach towards defining a vector is to consider all different kinds of requests as entries in the vector. The number of requests for each kind gives the value of its corresponding entry in the vector. However, for the system under study, there were many kinds of requests. With such a broad characterization of sessions it is very difficult to assess similarity. Another way to simplify the problem is to treat groups of requests as being similar. Menascé *et al.* [16] group requests based on their functional purpose. Requests are categorized as either browse, search, shopping cart, order, select, or home requests. A transition matrix gives the transition counts from one group to another. With six groups there are 36 attributes used to distinguish the clusters. In this way the transition matrix characterizes session length, navigational behaviour, and request mix within sessions. A Euclidean distance metric is used to compute the distance between matrices for the clustering algorithm. The distance metric is as follows:

$$d = \sqrt{\sum_{i=1}^{n}\sum_{j=1}^{n}(a_{(i,j)} - b_{(i,j)})^2} \qquad \text{(EQ 3)}$$

where $n$=6 is the number of groups and $a$ and $b$ are matrices. A weighted average of matrices is used to form new clusters with number of observations per cluster giving the weights.

The purpose of our study differs from [16]. Our purpose is primarily to characterize the impact of a request class mix on system performance and scalability. As a result the classes of requests presented in Section 6, cacheable, non-cacheable, and search, are chosen as groups to support clustering exercises. These help distinguish workload classes with respect to the use of system features that have significantly different resource demands. Furthermore, for our analysis, the use of counts in the distance metric (EQ 3) and the use of a matrix have two undesirable effects. These distinguish classes based on session length and navigational behaviour, as well as on workload mix. These were desirable for [16] but are not for our study. For large systems, the characterized differences in session length and navigational behaviour are not likely to have a significant impact on overall system performance.

As a result we use the following vector for describing each session:

$$a = \langle a_1/n,\ a_2/n,\ a_3/n \rangle \qquad \text{(EQ 4)}$$

The vector $a$ has three attributes. It describes the fraction (normalized counts) of requests in the session that belong to each of the three request classes; $a_1$ is the number of cacheable requests; $a_2$ is the number of non-cacheable requests; $a_3$ is the number of search requests; and $n$ is the total number of requests in the session, i.e., the sum of $a_1$, $a_2$, and $a_3$. The distance measure is as follows:

$$d = \sqrt{\sum_{i=1}^{n} (a_i - b_i)^2} \qquad \text{(EQ 5)}$$

where *a* and *b* are vectors.

We consider two clustering exercises. Section 7.3.1 and Section 7.3.2 provide a multi-class characterization of user and robot sessions, respectively. Based on the characterization, Section 7.3.3 gives parameters for a multi-class capacity planning model. The model can be used to assess the scalability of the system based on workload intensity and mix.

### 7.3.1 Characterizing User Sessions

This section characterizes user sessions from the July data set. We perform a clustering analysis using request class mix (with three attributes) and give two examples of clustering with respect to a single attribute. The single attribute cases consider the ratio of cacheable request counts to total request counts and search request counts to total request counts. The results of the clustering exercises are shown in Figure 7 to Figure 9.

Figure 7 shows four workload classes for users. 43% of user sessions are characterized as Heavy Cacheable, with an average 96% of requests being cacheable. These users do not rely on the search engine and cause little backend activity. They are "window shoppers." The mean inter-request time is 40 seconds and the average number of URLs per session is 7.5. We refer to the next two classes as Moderate Cacheable and Search. These classes represent 23% and 15% of sessions, respectively. These sessions are less skewed towards any one kind of request. The last class is referred to as Non-Cacheable. It represents 19% of sessions; 68% of these requests are not cacheable. The mean inter-request time is 75 seconds, which is significantly higher than for the other classes.

Next we consider clustering with respect to the ratio of cacheable requests to total requests. From a capacity planning perspective, this distinguishes classes based on their use of the system's caching infrastructure. Figure 8 shows the characteristics of the resulting four classes. The classes range from Highest cacheable with 95% of requests being cacheable to Low with 5% of requests cacheable. The results are similar to those of the characterization by request class mix, but there is a clear increase in mean inter-request time, as sessions make fewer cacheable requests, because those requests divide across Non-Cacheable and Search. From Figure 7 we see that non-cacheable requests are associated with high inter-request times. The inter-request times in Figure 8 range from 40 to 59 seconds.

Figure 9 distinguishes classes based on the ratio of search requests to total requests. Six classes of user sessions are reported. The percentage of search requests per session are 91, 67, 48, 29, 14, and 0 percent, respectively. As the percentage of searches decreases the mean inter-request time increases, again due to more non-cacheable requests. 64% of sessions do virtually no searching at all. The remaining classes have between 2 and 11% search requests.

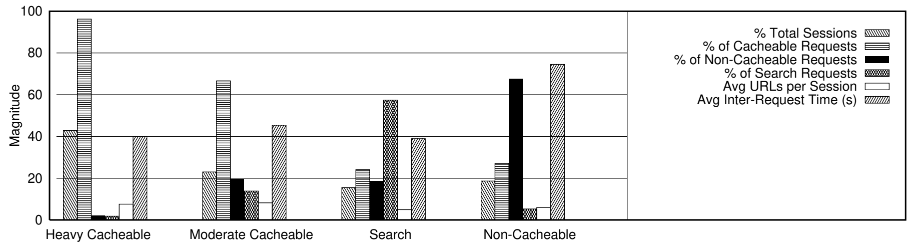


Figure 7 Clustering based on Request Class Mix for Users, July Data ($\beta_{cv}$ @ k = 4 is 1.67 ) (Min $\beta_{cv}$ @ k = 7 is 1.12)

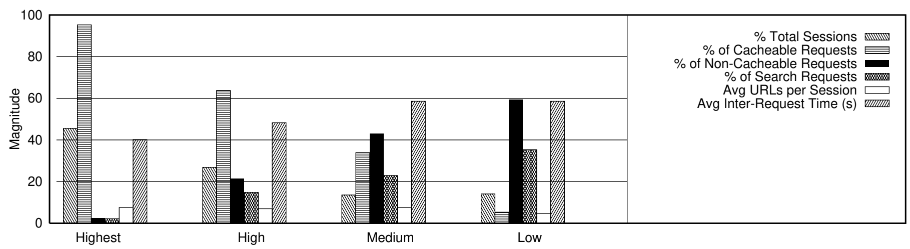


Figure 8 Clustering based on Ratio of Cacheable to Total Requests for Users, July Data ($\beta_{cv}$ @ k = 4 is 0.46 ) (Min $\beta_{cv}$ @ k = 4 is 0.46 )

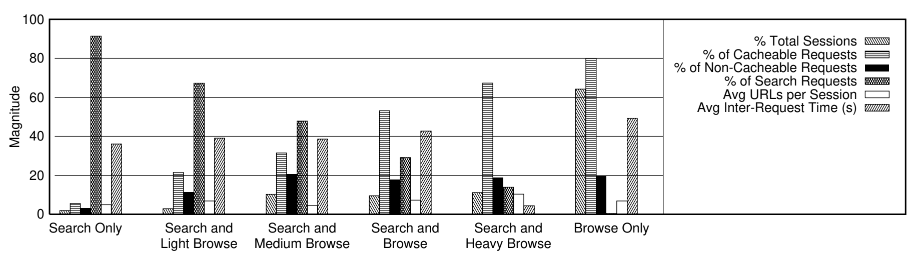


Figure 9 Clustering based on Ratio of Search to Total Requests for Users, July 24 Data ($\beta_{cv}$ @ k = 6 is 1.11 ) (Min $\beta_{cv}$ @ k = 10 is 0.89 )

### 7.3.2 Characterizing Robot Sessions

This section presents a second set of clustering exercises. It characterizes and compares classes of robot sessions obtained using request class mix and a transition matrix (with 3 rows/columns) based on counts. The latter example illustrates the advantages and disadvantages of using counts rather than normalized counts. The results for the July data set are given in Figure 10 and Figure 11, respectively.

Figure 10 presents three classes of robot sessions. 59% of robot sessions have an average of 92% cacheable requests. These robots are looking at known items, most likely observing pricing information. For this reason we refer to them as "window shopper robots." The average number of URLs per session is 53. 10% of robot sessions have an average of 91% non-cacheable requests. We refer to these as Backend robots. The remaining 30% of requests have a less skewed mix. Their behaviour is more user-like, with a request class mix and think time similar to the Moderate Cacheable and Search classes of Figure 7. However, these sessions have an average of 50 URLs per session. Some of these sessions may be real users with particularly long sessions; as noted earlier we have at least 9% false positives for our simple robot identification technique. Future work includes a better characterization of long user sessions and their differences from positively identified robot sessions.

A transition matrix with counts, as in [16], confounds session length and request class mix. This approach discovers four classes of user sessions as shown in Figure 11. The first class has an average of over 1000 URLs per session. For presentation, it is shown in the figure as 150. The fraction of sessions in this class is small, but virtually all of these requests are for cacheable documents. With this approach, 82% of sessions appear as user-like sessions. The mix and mean inter-request times of this class are similar to the Moderate Cacheable class of Figure 7. The mean number of URLs per session for the clusters are over 1000, 111, 102, and 44. The middle classes differ because one class is biased towards search requests and the other towards non-cacheable requests.

Menascé *et al.* [17] suggest that high session lengths combined with highly skewed access to a particular class of requests is a good indicator of robot sessions. This is compatible with the data from our study. We believe, however, that there is a very strong likelihood of false positives and false negatives. It will always be very difficult to recognize robots that intentionally behave like users.

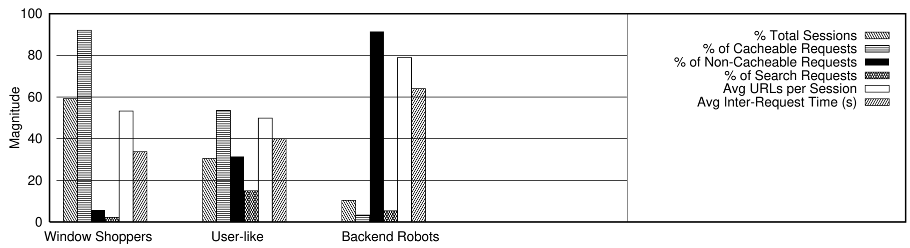


Figure 10 Clustering based on Request Class Mix for Robots, July Data ($\beta_{cv}$ @ k = 3 is 1.11 ) (Min $\beta_{cv}$ @ k = 3 is 1.11 )

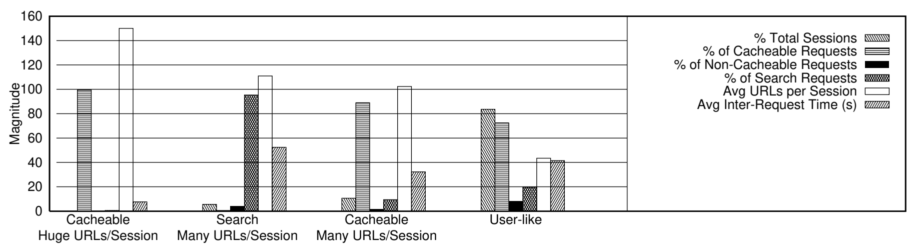


Figure 11 Clustering based on Transition Matrix and Counts for Robots , July Data ($\beta_{cv}$ @ k = 4 is 1.83 ) (Min $\beta_{cv}$ @ k = 10 is 1.55 )

### 7.3.3 Capacity planning example

This section demonstrates how the results of clustering exercises can be used to create multi-class queuing models [11]. Without loss of generality, we consider a multi-class model for application server CPU resources.

For the model, each class requires a demand value. This paper offers ratios of demand with respect to the CPU demand of a request cache hit. The product of a request class mix (from the clusters in Figure 7 and Figure 10) and the application server CPU demand ratios of Table 2 give a normalized per-class demand value for the application server CPU resources for classes of users and robots, respectively. These values are given in Table 4 for the user classes and Table 5 for the robot classes. For example, Table 4 indicates that a heavy cacheable user session requires 365 times the resource demand of a single request cache hit.

**Table 4 Per Class Session Demand Value for User Classes**

| Workload Class | Ratio of Session Demand to Request Cache Hit |
|---|---|
| Heavy Cacheable | 365 |
| Moderate Cacheable | 2570 |
| Search | 4175 |
| Non-Cacheable | 7002 |

**Table 5 Per Class Session Demand Values for Robot Classes**

| Workload Class | Ratio of Session Demand to Request Cache Hit |
|---|---|
| Window Shoppers | 753 |
| User-like | 3795 |
| Backend Robots | 9347 |

The demand ratios of Table 4 and Table 5 support utilization and queuing analyses. In Section 6, an example was given to show the sensitivity of cost-effective system scalability to request class mix and response personalization. The same method can be used to assess the sensitivity of system performance and cost-effective scalability to the above workload classes.

Each workload class imposes significantly different resource demands on the application servers. The classes offer insights into the impact of users and robots on system performance and scalability. The following examples illustrate why multi-class models are important for this system.

- A large increase in Search users per unit time will cause more than ten times the demand on application servers than a similar increase in heavy cacheable users. Such insights can be used when recommending changes for site design or implementation.
- The system should be sized for increases in user sessions per unit time, not the heavier robot sessions. Robot load is not likely to increase with promotions in the same way as user load will.

# 8 PERFORMANCE AND SCALABILITY ISSUES

The system under study had moderate changes in capacity requirements during the March and July measurement periods. During these periods the minimum to maximum capacity requirements differed by a factor of nine. The mean and maximum capacity requirements differed by a factor of two. It is important to note, however, that there were no shopping promotions in place during our measurement periods, and neither of our measurement periods overlapped with traditional busy periods in consumer shopping (e.g., Valentine's Day, Mother's Day, Father's Day, Easter, and Christmas). Thus we expect that the range in capacity requirements could be (significantly) higher at times. Anecdotal evidence from the networking community suggests bandwidth usage on busy Internet links can vary by a factor of 15 over a 24 hour period. In the workload of the 1998 World Cup Web site, the average request rate during the busiest 15 minute interval was 19 times greater than the average request rate for the entire measurement period [3].

For the Web-based shopping system under study, the application servers were a system bottleneck because they performed all of the required response generations. As a result, much of our characterization and modeling focussed on the application server tier.

We found that changes in request class mix of only ±5% can affect capacity per application server by a range of 1.17 to 0.87. Request cache misses or personalization causing page generation up to 8% of cacheable responses can decrease capacity per server by a factor of 2.7. This suggests that 2.7 times as much application server infrastructure is needed to support a system with 8% personalization of cacheable requests, as opposed to 0% personalization. This can be an issue for sites that target many users and have wide fluctuation in workload intensity.

One approach for designing a scalable e-commerce system is to deploy an application server cache that is large enough to store all of a site's responses (or static portions thereof). With this approach, a cache replacement policy is not necessary. This strategy is recommended due to the large difference in resources needed to serve a response from cache compared to generating a response on-demand (a difference of 500x for cache hits and cache misses for the system under study, as shown in Table 2). Furthermore, this cache should be ’primed’ (i.e., filled in advance of customer demand) to avoid the risk of any additional response generations during intervals of peak capacity requirements.

To summarize, Web-based shopping sites must have adequate capacity to support peak loads. This presents a planning challenge for large systems due to:

- the lack of control over how many users may arrive;
- the large number of servers that may be needed;
- the sensitivity of resource demands to workload mix; and
- the sensitivity of resource demands to features such as personalized responses.

Systems are often said to be scalable if they present mechanisms for adding capacity as load increases. The architectures for typical Web-based shopping sites support near linear scalability by enabling the addition of servers as capacity requirements increase. However, linear scalability is not always adequate for these systems. The uncertainty of workloads and the fluctuation in capacity requirements can make linear scalability unattractive. Other mechanisms are also needed for managing capacity. We consider Quality of Service (QoS) management techniques for dealing with system overload and caching techniques to reduce the likelihood of such overload.

For improving the service a business provides, features such as personalization of service to customers are important. However, such features come at a cost. Thus, there is a trade-off to be considered. During periods when ample system resources are available, personalization can be offered for all customers. As system resources become scarce due to the arrival of additional customers, the site could decide to offer full personalization to only a select group of customers and reduced personalization (or none at all) to others. In the traditional retailing world, customers expect better service during non-peak periods. Similarly, software systems for Web-based shopping environments should provide for cost-effective scalability by employing QoS management techniques as described in the literature [1].

Efforts should be made to recognize robots. In most cases requests from robots should be considered as lower priority tasks than requests from all other customers. This is important -- the quality of service to customers should not be impacted by robots.

Another approach for further improving the scalability of large Web-based shopping systems is to extend caching technologies (e.g., content distribution networks) developed for static Web workloads [24]. Utilizing distributed caching techniques is appealing for numerous reasons. As we discussed in Section 5.3, even a relatively small cache could have a positive impact on system performance, since many customer requests are for a small percentage of the site's resources. In Section 5.4 we observed that many of the requests came from a small number of clients. This suggests that even a few caches located at strategic points in the network could be very effective for reducing the load on the site's servers. Pushing content closer to the edge of the network (where the customers are) can also improve the customer experience by reducing response latency.

Unfortunately, caching in an e-commerce environment is more complicated than in a static Web environment. For example, session state management, privacy, security, personalization, information consistency (e.g., pricing and availability) and the ability to monitor customer behaviour are all issues that must be addressed. While these problems are difficult, they are not impossible to solve. One approach is to separate the static portion of each Web page from the personalized (i.e., dynamic) parts. The static portions could then be cached, reducing the amount of dynamic content that must be generated on demand. Techniques such as these were utilized to improve the performance of the 1998 Olympic Games Web site [7]. A related technology is delta encoding, which reduces the amount of data that must be retrieved when a static Web page has been modified [19]. Server-push based cache consistency mechanisms could also be used to give a business control over what content gets cached, where it gets cached, and for how long it gets cached. Existing cache consistency mechanisms designed for static Web pages could be used or modified for use in an e-commerce environment [9][15][25][26].

# 9 Conclusions

For businesses intent on providing the best possible service to their customers, system scalability for Web-based shopping systems is an important issue. For such systems linear scalability is not always adequate. These systems are sensitive to many factors beyond a planner's control. As a result, application- and system-level QoS mechanisms are needed to properly manage overload conditions.

Web-based shopping systems often have significantly different resource demands for application server request cache hits and misses. Replacement algorithms are not the primary issue for such caches. These (server-side) caches must be sized in an appropriate manner to avoid all misses, particularly during periods of significant customer demand.

Effective network-based caching techniques enhance system scalability by limiting the likelihood of system overload. The results of our workload analysis of a Web-based shopping system suggest that caching mechanisms developed for the scalable delivery of static Web content can be exploited by e-commerce systems as well. Section 8 lists numerous challenges that must be addressed before these techniques can be fully exploited.

In this study, cluster analysis was used for the purpose of capacity planning. We employed an approach based on classes of requests with significantly different resource demands. This approach has the advantage of forming workload classes in terms of system features that affect performance most.

Additional workload characterization studies are required for several reasons. First, our measurement data did not contain all of the information of interest to us. Thus, a number of interesting questions remain unanswered (e.g., what was the response size distribution for this workload?). Characterizing the workloads of additional e-commerce systems is needed to validate the hypotheses we proposed based on the characterization of a single e-commerce site. The characterization of additional e-commerce sites may also provide insights into additional strategies that can help improve the performance and scalability of e-commerce sites. Performing workload characterization of a set of sites on an ongoing basis would allow us to understand trends in e-commerce workloads and to evaluate the impact of changes to systems. Finally, we believe that additional study is needed in identifying robots, so that these processes do not degrade the experience of actual customers.

## Acknowledgments

The authors are extremely grateful to the administrators of the e-commerce site under study for providing us with the workload data and information on the system architecture. The authors would also like to thank Doug Grumann for the information he provided on MeasureWare, and the anonymous reviewers for their exceptionally constructive feedback.

# 10 REFERENCES

**[1]** T. Abdelzaher and N. Bhatti, "Web Server QoS Management by Adaptive Content Delivery", Hewlett-Packard Laboratories Technical Report HPL-1999-161, December 1999.

**[2]** V. Almeida, R. Riedi, D. Menascé, W. Meira, F. Ribeiro and R. Fonseca, "Characterizing and Modeling Robot Workload on E-Business Sites", to appear as a poster at *ACM SIGMETRICS 2001*, Cambridge, MA, June 2001.

**[3]** M. Arlitt and T. Jin, "Workload Characterization of the 1998 World Cup Web Site", *IEEE Network*, Vol. 14, No. 3, pp. 30-37, May/June 2000.

**[4]** M. Arlitt and C. Williamson, "Internet Web Servers: Workload Characterization and Performance Implications", *IEEE/ACM Transactions on Networking*, Vol. 5, No. 5, pp. 631-645, October 1997.

**[5]** P. Barford, A. Bestavros, A. Bradley and M. Crovella, "Changes in Web Client Access Patterns", *World Wide Web Journal Special Issue on Characterization and Performance Evaluation*, Vol. 2, pp. 15-28,1999.

**[6]** L. Breslau, P. Cao, L. Fan, G. Phillips and S. Shenker, "Web Caching and Zipf-Like Distributions: Evidence and Implications", *Proceedings of IEEE Infocom '99*, New York, NY, March 1999.

**[7]** J. Challenger, A. Iyengar and P. Dantzig, "A Scalable System for Consistently Caching Dynamic Web Data", *Proceedings of IEEE Infocom '99*, New York, NY, March 1999.

**[8]** C. Cunha, A. Bestavros, and M. Crovella, "Characteristics of WWW Client-Based Traces", Technical Report TR-95-010, Boston University Department of Computer Science, April 1995.

**[9]** J. Dilley, M. Arlitt, S. Perret and T. Jin, “The Distributed Object Consistency Protocol Version 1.0”, Hewlett-Packard Laboratories Technical Report HPL-1999-109, September 1999.

**[10]** J. Hartigan, *Clustering Algorithms*, John Wiley & Sons, Inc., New York, NY, 1975.

**[11]** R. Jain, *The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling*, John Wiley & Sons, Inc., New York, NY, 1991

**[12]** L. Kaufman and P. Rousseeuw, *Finding Groups in Data*, John Wiley & Sons Inc., New York, NY, 1990.

**[13]** M. Koster, “A Standard for Robot Exclusion”, June 1994.

**[14]** J. Lee and M. Podlaseck, “Visualization and Analysis of Clickstream Data of Online Stores for Understanding Web Merchandising”, *Special Issue of the International Journal of Data Mining and Knowledge Discovery on E-Commerce and Data Mining*, Kluwer Academic Publishers, 2000.

**[15]** C. Liu and P. Cao, “Maintaining Strong Cache Consistency in the World Wide Web”, *Proceedings of the 17$^{th}$ IEEE International Conference on Distributed Computing Systems*, May 1997.

**[16]** D. Menascé and V. Almeida, *Scaling for E-Business*, Prentice Hall PTR, Upper Saddle River, NJ, 2000.

**[17]** D. Menascé, V. Almeida, R. Fonseca and M. Mendes, “A Methodology for Workload Characterization of E-Commerce Sites”, *Proceedings of ACM Conference on Electronic Commerce*, Denver, CO, Nov. 1999.

**[18]** D. Menascé, V. Almeida, R. Riedi, F. Ribeiro, R. Fonseca and W. Meria, “In Search of Invariants for E-Business Workloads”, *Proceedings of ACM Conference on Electronic Commerce*, Minneapolis, MN, Oct. 2000.

**[19]** J. Mogul, F. Douglis, A. Feldmann, and B. Krishnamurthy, “Potential Benefits of Delta Encoding and Data Compression for HTTP”, *Proceedings of ACM SIGCOMM ‘97*, Cannes, France, pp. 181-194, September 1997.

**[20]** V. Padmanabhan and L. Qui, “The Content and Access Dynamics of a Busy Web Site: Findings and Implications”, Proceedings of ACM SIGCOMM 2000, Stockholm, Sweden, pp. 111-123, August 2000.

**[21]** F. Reichheld and W. Sasser, “Zero Defections: Quality Comes to Services”, Harvard Business Review, September-October 1990.

**[22]** D. VanderMeer, K. Dutta, A. Datta, K. Ramamritham and S. Navathe, “Enabling Scalable Online Personalization on the Web”, *Proceedings of ACM Conference on Electronic Commerce*, Minneapolis, MN, Oct. 2000.

**[23]** Vantage Point Performance Agent, `http://www.openview.hp.com/products/vpperformance/`

**[24]** J. Wang, “A Survey of Web Caching Schemes for the Internet”, Computer Communication Review, Vol. 29, No. 5, pp. 36-46, October 1999.

**[25]** J. Yin, L. Alvisi, M. Dahlin and C. Lin, “Hierarchical Cache Consistency in WAN”, Proceedings of the 2$^{nd}$ USENIX Symposium on Internet Technologies and Systems, Boulder, CO, pp. 13-24, October 1999.

**[26]** H. Yu, L. Breslau and S. Schenker, “A Scalable Web Cache Consistency Architecture”, *Proceedings of ACM SIGCOMM ‘99*, Cambridge, MA, pp. 163-174, September 1999.