Workload Characterization of a Web Proxy in a Cable Modem Environment

Martin Arlitt, Rich Friedrich, Tai Jin
Internet Systems and Applications Laboratory
HP Laboratories Palo Alto
HPL-1999-48
April, 1999

E-mail: [arlitt,richf,tai]@hpl.hp.com

World-Wide Web, workload characterization, performance, proxy caching

This paper presents a detailed workload characterization study of a World-Wide Web proxy. Measurements from a proxy within an Internet Service Provider (ISP) environment were collected. This ISP allows clients to access the Web using high-speed cable modems rather than traditional dial-up modems. By examining this site we are able to evaluate the effects that cable modems have on proxy workloads.

This paper focuses on workload characteristics such as file type distribution, file size distribution, file referencing behaviour and turnover in the active set of files. We find that when presented with faster access speeds users are willing to download extremely large files. A widespread increase in the transfer of these large files would have a significant impact on the Web. This behaviour increases the importance of caching for ensuring the scalability of the Web.

Internal Accession Date Only

Workload Characterization of a Web Proxy in a Cable Modem Environment

Martin Arlitt, Rich Friedrich and Tai Jin
Hewlett-Packard Laboratories
Palo Alto, CA
{arlitt, richf, tai}@hpl.hp.com

Abstract

This paper presents a detailed workload characterization study of a World-Wide Web proxy. Measurements from a proxy within an Internet Service Provider (ISP) environment were collected. This ISP allows clients to access the Web using high-speed cable modems rather than traditional dial-up modems. By examining this site we are able to evaluate the effects that cable modems have on proxy workloads.

This paper focuses on workload characteristics such as file type distribution, file size distribution, file referencing behaviour and turnover in the active set of files. We find that when presented with faster access speeds users are willing to download extremely large files. A widespread increase in the transfer of these large files would have a significant impact on the Web. This behaviour increases the importance of caching for ensuring the scalability of the Web.

1 Introduction

Since its inception in the early 1990's, the World-Wide Web (WWW or the Web) has grown at an exponential rate. This rapid growth is expected to persist as the number of Web users continues to increase and as new uses for the Web such as electronic commerce become widely accepted. As the Web evolves into a part of the daily lives of people and businesses, Web performance becomes not only desirable but necessary.

There are a number of approaches to improving the performance of the Web. Web sites can utilize clusters of machines to handle incoming requests. Network links can be upgraded to higher bandwidths to handle the increased volume of data traffic. However, these approaches are really only short-term solutions. A more permanent solution to the scalability problems of the Web is file caching [18][27]. File caching is used to store documents[1] closer to the clients that request them. By doing this user latency, network loads and origin server loads can all be reduced. File caching can help eliminate network *hotspots* by reducing the volume of traffic to popular sites. File caching can also dynamically adjust to eliminate *flash crowds* (i.e., temporarily popular sites), something that network upgrades cannot [27].

This paper presents a detailed workload characterization of a Web proxy server. We focus on identifying characteristics that we believe are important for proxy cache performance. We also examine the data set to determine the effects of higher client access speeds on the proxy workload. The main observations from our characterization study are:

- HTTP accounts for almost all of the client requests (99.3%) and most of the data traffic (87.7%)
- more than 73% of all requests are for image files; a further 12% are for HTML files
- the unique file and transfer size distributions are heavy-tailed
- clients are (more) willing to download extremely large files when access speeds are higher
- growth in usage of the service is due to new subscribers and more use by all subscribers
- file referencing patterns are non-uniform
- more than 60% of the unique files were requested only a single time
- the active set of files changes over time, although some files remain popular for extended periods

The remainder of this paper is organized as follows. Section 2 provides an overview of related work. Section 3 describes the methodology of the workload characterization study. Section 4 presents the detailed results of the workload characterization. Section 5 summarizes the paper, presents our conclusions and discusses future directions in Web (proxy) caching research.

2 Related Work

Workload characterization is a crucial component of systems design as it allows us to understand the current state of the system. By characterizing a system over time we can observe the effects that

1. Throughout this paper we use the terms *file*, *document* and *object* interchangeably.

changes to the system have had. Workload characterization is also necessary in the design of new components.

In this paper we focus on the characterization of a Web proxy workload. Several workload studies of Web proxies have already been reported in the literature, including [8],[14],[17], and [21]. We examine our data set for similar characteristics to determine how the workload changes in a cable modem environment. Other studies have examined the workloads of various components of the Web, including clients ([7],[13]), servers ([4],[12],[22]) and the HTTP protocol ([6],[16],[23]).

The main performance benefit of Web proxies is the file caching that they perform. There are two general approaches to file cache management. One approach attempts to use as few resources as possible by making good replacement decisions when the cache is full (we call this the *elegant* approach). The alternative approach is to provide the cache with sufficient resources such that few replacements are needed (this is the *brute-force* approach). Supporters of the elegant approach have utilized workload characterization to develop a number of different replacement policies [3][9][20][26]. Feldmann *et al.* [15] point out that proxies may also improve performance by caching persistent connections.

3 Methodology

This section describes the methodology of the workload characterization study. Section 3.1 presents background information on the data collection site. Section 3.2 discusses the data that was collected at the site. Section 3.3 describes how the collected data was reduced into a more manageable format. Section 3.4 summarizes the assumptions we made to address the limitations of our data set.

3.1 Data Collection Site

The site under study is an Internet Service Provider (ISP) that offers interactive data services to residential and business subscribers. These subscribers utilized cable modems to connect to the ISP's server complex. Several thousand subscribers utilized the system during the data collection period. All subscriber requests for Web objects (e.g., HTTP, FTP, and Gopher requests) were forwarded to a single server running a commercial proxy software package. This proxy server includes a file cache so some of the client requests were satisfied within the server complex. On a cache miss the proxy retrieved the object from an origin server on the Internet. All requests for the ISP's own Web site issued by this group of subscribers went through the proxy. Since the ISP's Web site was quite popular with the subscribers the hit rate in the proxy cache is higher than traditionally seen in proxy caches.

This site provides us with a unique opportunity to (potentially) characterize client access patterns of the future. The cable modems utilized at this site had peak bandwidths reaching several megabits per second. This is several orders of magnitude more than is achieved by traditional dialup modems. The increased access bandwidths made proxy caching important for reducing user latency as the ISP's connection to the Internet was the main bottleneck in the system.

3.2 Data Collection

The access logs of the proxy server described in Section 3.1 were collected for this study. These logs were collected on a daily basis from January 3rd, 1997 until May 31st, 1997. The access logs were not available on 13 days and were incomplete on four other days. Despite these gaps in the data set we have a relatively complete view of the proxy workload for an extended period of time.

Each entry in an access log contains information on a single request received by the Web proxy from a client. Each entry includes the following information:

- **client address**: the IP address that was dynamically assigned to the client upon connection to the ISP
- **timestamp**: the date and time that the request was made
- **request**: contains the method (e.g., GET, HEAD, etc.), the requested URL and the protocol used for client-proxy communication
- **status codes**: indicate the nature of the proxy response and, if necessary, the origin server response
- **header data**: the amount of header data, measured in bytes, passed between the client, the proxy, and, if necessary, the origin server
- **content data**: the amount of content data, measured in bytes, passed between the client, the proxy and, if necessary, the origin server
- **transfer time**: the amount of time (millisecond precision) between the arrival of the request at

the proxy and the end of the response from the proxy

Some of the information that is not specifically recorded in the access logs can be inferred by examining the data. For example, proxy cache hits and misses can be determined by examining both the proxy and origin server response codes. Unfortunately, not all information of interest is available in the access logs. For example, the logs contain no information that would allow us to correctly identify individual users. There is also no information that enables us to correctly identify all of the aborted transfers (i.e., the user becomes impatient with a slow response and presses the Stop button on the browser).

Table 1 contains the summary statistics for the raw data set. The access logs contain a total of 117,652,652 requests for a total of 1,340 GB of content data.

Table 1: Summary of Access Log Characteristics (Raw Data)

Access Log Duration	January 3rd - May 31st, 1997
Total Requests	117,652,652
Avg Requests/Day	840,371
Total Content Data	1,340 GB
Avg Content Data/Day	9.6 GB

3.3 Data Reduction

Due to the extremely large access logs created by the proxy (nearly 30 GB of data) we found it necessary to create a smaller, more compact log due to storage constraints and to ensure that our workload analyses could be completed in a reasonable amount of time. We performed these reductions in two ways: by storing data in a more efficient manner (e.g., we mapped all distinct URLs to unique integers), and by removing information of little value (e.g., we kept only GET requests which accounted for 98% of all requests and 99.2% of the content data transferred). We utilize a database to perform the mapping of URLs to unique integers. This approach allows us to map the integers back to the original URL. During the reduction process it became apparent that there were too many unique URLs to map all of them (the database became a bottleneck). Since the ability to determine the URL from the integer identifier was important to us, we decided to map only the cacheable URLs. We considered a URL to be cacheable if it did not contain substrings such as '`cgi-bin`' or '`?`', if it did not have a file extension such as '`.cgi`', and if the origin server response contained an appropriate status code (e.g., `200 Success`). Any URL that failed one or all of these tests was considered to be uncacheable.

After reducing the access logs in this manner we recalculated the overall statistics. The results are shown in Table 2. The reduced data set contains 115,310,904 requests for 1,328 GB of content data. 16,225,621 unique cacheable URLs and a total of 9,020,632 uncacheable URLs are present in the reduced data set. Assuming that the cache is initially empty, this means that the minimum number of requests that could not be satisfied by the proxy cache is 25,246,253 or 21.9% of all GET requests. The transfer of the unique cacheable responses accounted for 389 GB of the content data; a further 56 GB of content data was transferred in response to requests for uncacheable objects. In total, a minimum of 445 GB or 33.5% of all content data needed to be transferred by the origin servers.

Table 2: Summary of Access Log Characteristics (Reduced Data)

Access Log Duration	January 3rd - May 31st, 1997
Total Requests	115,310,904
Total Content Bytes	1,328 GB
Unique Cacheable Requests	16,255,621
Total Uncacheable Requests	9,020,632
Unique Cacheable Content Bytes	389 GB
Total Uncacheable Content Bytes	56 GB

3.4 Assumptions

In Section 3.4 we stated that not all information of interest is available in the access logs. In particular we cannot accurately identify if a file has been modified at the origin server since the previous request for that file, or if the user aborted the transfer of a file. Since we felt it was important to identify these occurrences, we developed a method for approximating when one of these conditions has happened. Our approach monitors the size reported for a file on every request for that file. We assume that if no change in the size occurs then the file has not been modified. We speculate that if a file is modified that it will result in a relatively small change in the size. An aborted request is identified if the number of bytes transferred is less than our current estimate of the requested file's size. While

these assumptions are not always true we believe they will allow us to get a reasonable estimate. Based on an analysis of this data set, we chose a threshold value of 5% to distinguish between a file modification and an aborted transfer. This means that if a file increases or decreases in size by less than 5% it is considered a modification; otherwise an abort has occurred. Using this threshold value we estimate that 10.3% of all requests in the data set were aborted. Since Feldmann *et al.* [15] found similar values in their study of a traditional dial-up ISP environment, we believe that our approach is not unrealistic.

4 Workload Characterization

In this section we present the results of our workload characterization. We analyze only the reduced data set described in Section 3.3 for these analyses. We begin with an examination of the different protocols in use on the Web.

4.1 Protocols

Our first analysis examines the protocol (e.g., HTTP, FTP, Gopher) contained in each URL in the data set. Table 6 shows that HTTP is the dominant protocol, accounting for over 99% of all requests and almost 88% of the content data. The only other protocol responsible for any significant amount of activity is FTP. Although FTP was seen in only 0.3% of requests it accounted for 12.1% of the total content data. These results suggests that HTTP has all but eliminated the use of the Gopher protocol, and has significantly altered the way in which FTP is used. The primary purpose of FTP in this workload appears to be to provide clients with access to very large files.

For the remainder of this paper we focus exclusively on the HTTP protocol. Due to space constraints we do not include the results from our analysis of the FTP requests. We found that there is little benefit to be gained from caching FTP files, as few clients are interested in these files and they consume a lot of space in the cache. Both of these characteristics degrade the potential hit rates for the cache. Furthermore, the FTP protocol does not provide a consistency mechanism, which further undermines the usefulness of caching FTP files. Finally, we observed problems with the caching of FTP files due to incompatibilities between the HTTP and FTP protocols. For example, the proxy repeatedly returned cached error messages to clients rather than the files that were requested.

Table 6: Breakdown of Requests by Protocol

Item	HTTP	FTP	Gopher	Other
Requests (%)	99.30	0.30	0.02	0.38
Content Data (%)	87.70	12.10	0.03	0.17
Avg Content Size(KB)	10.6	432	14.4	5.7

4.2 Response Status Codes

In this section we analyze the proxy and origin server response codes for the HTTP requests. Table 3 shows the breakdown of the proxy response code distribution. Most of the client

Table 3: Breakdown of Proxy Response Codes for HTTP Requests

Item	Successful(200)	Found(302)	Not Modified(304)	Client Error(4xx)	Server Error(5xx)	No Response	Other
Requests (%)	75.6	3.9	15.9	1.6	1.5	0.6	0.9
Content Data (%)	99.8	0.1	0.0	0.1	0.0	0.0	0.0

Table 4: Breakdown of Origin Server Responses for HTTP Requests

Item	Successful(200)	Found(302)	Not Modified(304)	Client Error(4xx)	Server Error(5xx)	No Response	Other
Requests (%)	52.1	3.9	14.3	1.6	0.1	26.9	1.1
Content Data(%)	77.0	0.1	0.0	0.1	0.0	22.8	0.0

Table 5: Breakdown of HTTP Requests by File Type

Item	HTML	Images	Audio	Video	Format	Text	Comp	Exe	Uncache	Other
Requests(%)	12.4	73.1	0.6	0.2	0.0	0.2	0.2	0.1	8.6	4.6
Content Data(%)	4.8	47.6	3.9	19.9	0.2	0.1	5.8	8.3	4.7	4.7

requests (75.6%) result in the Successful transfer of the requested document. These responses account for almost all of the content data. The next most common response is 'Not Modified' which makes up 15.9% of all HTTP responses but none of the content data. These responses indicate client cache hits that are being validated. This means the client is checking to ensure that the version of the file it has in its cache is consistent with the latest version of that file at the origin server. While these validations do not reduce the number of requests that clients send to the proxy they do reduce the volume of content data that needs to be transferred

Table 4 presents the results for the origin server response codes. This distribution differs from the results in Table 3 in three ways: there are fewer 'Successful' responses; there are significantly more 'No Responses'; and there are slightly fewer 'Not Modified' responses. The first two differences are caused by proxy cache hits that do not require the file to be validated. Thus the 'No Response' column provides an estimate of the hit rate and byte hit rate achieved by the proxy cache for the data set under study. The reduction in 'Not Modified' responses from the origin server is caused by the proxy responding directly to some of the client validation requests.

4.3 File Types

Our next analysis classifies the types of files being requested by clients. We place each file into one of ten categories: HTML, Image, Audio, Video, Formatted (Format), Text, Compressed (Comp), Executable (Exe), Uncacheable, or Other types. The categorization for the cacheable file types is based on the file extension (e.g., `.gif` and `.jpg` are Images, `.pdf` is Formatted, `.gz` is Compressed, `.exe` is Executable, etc.). The criteria for classifying a file as uncacheable are given in Section 3.3. The 'Other' category contains all files that could not be classified based on their extension, although they could potentially belong in one of the defined categories.

Table 5 shows the results of the file type analysis for all of the HTTP requests that resulted in a Successful (i.e., status 200) response from the proxy. These results indicate that Images (73.1%) and HTML files (12.4%) account for most of the requests. 8.6% of the responses were uncacheable. Despite the increased bandwidth available to the clients there does not appear to be a significant number of requests for multimedia files (e.g., Audio and Video). However, this may be due in part to a lack of multimedia objects on popular Web sites rather than just minimal subscriber interest.

HTML and Image files account for just over half of the content data transferred from the proxy to the clients. This is significantly less than the percentage of requests that these types receive, since most of the HTML and Image files are quite small (see Section 4.4). The content data is impacted heavily by the transfer of larger file types such as Audio, Video, Compressed and Executables. While these types make up only 1.1% of the requests, they accounted for 37.9% of the content data traffic.

4.4 File Size Distributions

One of the obstacles for Web caching is working with variable-sized objects. Earlier studies of Web traffic found that Web file sizes span several orders of magnitude, from tens of bytes to tens of MB [4][12]. These studies suggest that the distribution of Web file sizes is *heavy-tailed*. A heavy-tailed distribution has the property that the tail of the distribution declines relatively slowly [11]. This means that when sampling random variables from a distribution of this type, the probability of obtaining extremely large values is non-negligible. The mathematical definition of a heavy-tailed distribution is $P[X > x] \sim x^{-\alpha}$, $x \to \infty$, $0 < \alpha < 2$.

To examine a distribution for evidence of a heavy-tail we plot the complementary distribution (CD) function on log-log axes and examine the results for linear behaviour in the upper tail [10].

Table 7: Unique File Size Information by File Type

Item	All Files	HTML	Image	Audio	Video	Format	Text	Comp	Exe	Other
Number	16,110,226	3,433,769	11,033,981	136,225	43,583	5,884	32,378	49,017	14,488	1,360,901
Mean (bytes)	21,568	6,354	14,032	135,734	1,593,565	247,374	30,026	553,781	1,642,792	21,686
Median (bytes)	4,346	3,051	4,694	37,806	925,735	79,920	5,854	92,263	766,692	5,719
Maximum (MB)	148	12.7	89.2	28.5	148	222.1	27.7	86.0	74.1	49.7
Total Size (GB)	323	20	144	17	65	1	1	25	22	28

In the remainder of this section we examine two different file size distributions. In Section 4.4.1 we analyze the sizes of the unique cacheable files requested in the access log. In Section 4.4.2 we study the sizes of all Successful HTTP responses from the proxy.

4.4.1 Unique File Size Distribution

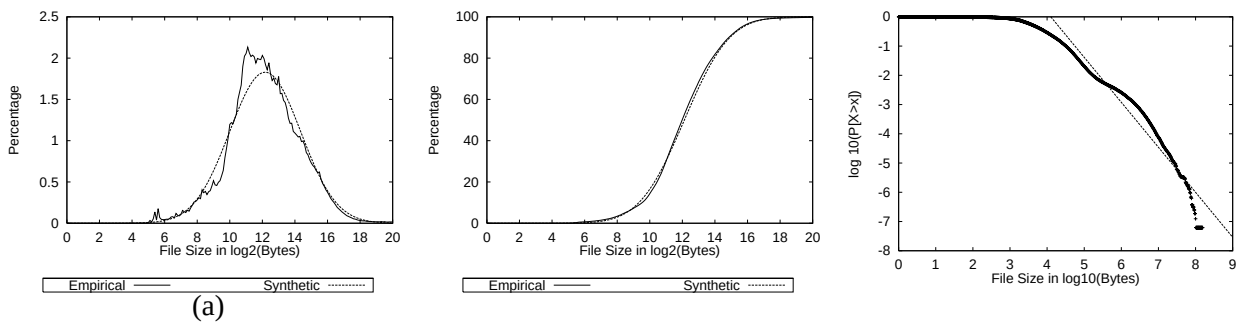
Our first analysis examines the sizes of each file that was successfully transferred (i.e., aborted transfers, as defined in Section 3.4, are not considered) at least once in the data set. For the purpose of this study we utilize the initial size recorded for each unique file. Since some of the unique files change over time so too will the unique file size distribution. However, we believe that the choice of which size to use for a file will only affect the parameters of the distribution and not the distribution itself.

Table 7 presents the overall statistics on the unique files transferred in the data set. These statistics were calculated for the set of all unique files as well as for each class of files. A total of 16,110,226 unique HTTP files were seen in the data set. The combined size of these files was 323 GB. 90% of the unique files were either HTML or Image objects. However, these objects account for only 51% of the total size. 40% of the total size is due to the presence of a few large file classes (Audio, Video, Compressed and Executable).

The median value of the unique file size distribution is quite small at only 4,346 bytes. The mean size is several times larger (21,568 bytes). This skew is caused by the presence of a few extremely large files. However, Table 7 reveals that no one file type is responsible for the presence of these large files. In fact, all of the classes contain some extremely large files. We confirmed that these large files did indeed exist on the Web by downloading them ourselves. We speculate that these extremely large transfers are a direct result of the increased bandwidth available to the clients, since it is unlikely that a subscriber would be patient enough to wait more than 6 hours for a 148 MB file to transfer at 56Kb/s.

Figure 1 shows the analysis of the size distribution for all unique HTTP files in the data set. We have applied a logarithmic transformation to the file sizes to enable us to identify patterns across the wide range of sizes [25]. Figure 1(a) compares the empirical distribution to a synthetic lognormal distribution with parameters μ=12.17 and σ=2.18. Figure 1(b) provides the corresponding cumulative frequency plot. Although there are clearly differences between the empirical and synthetic distributions, the body of the unique file size distribution appears to follow a lognormal distribution.

While most of the unique files requested in the data set were less than 64 KB in size, a few were substantially larger. Figure 1(c) shows the CD plot for the tail of the unique file size distribution. Since this distribution does exhibit some linear behaviour in the upper region we conclude that it is indeed heavy-tailed. We estimate the weight of the tail (α) from the slope of this linear region [10]. This analysis results in an estimate of 1.5 for α.



(a)

Figure 1 Analysis of Unique File Size Distribution: (a) Frequency; (b) Cumulative Frequency

4.4.2 Transfer Size Distribution

Our next analysis focuses on the sizes of all successful transfers from the proxy. We do not include the transfers that we believe to have been aborted. Table 8 presents the overall statistics on the successful transfers, for all transfers (Xfers) as well as by file type. The decrease in mean and median sizes between Table 7 and Table 8 for the overall distributions indicates that the smaller unique files tended to get requested more often than the large ones. However, for the Video and Executable classes, the mean and median sizes increased, indicating that the larger files were slightly more popular in these classes. Table 8 reveals that HTML and Image files accounted for most of the requests (86% of all successful transfers, 94% when only the cacheable responses are considered). The transfer of large file types (Audio, Video, Compressed and Executable) accounted for 38% of the content data transferred even though only 1% of the requests were for files of this type.

Figure 2 shows the frequency and cumulative frequency histograms for the successful transfer size distribution. The body of the successful transfer size distribution appears to follow the lognormal distribution (μ=11.68, σ=2.24) quite closely, as was the case with the unique file size distribution. The tail of the successful transfer size distribution is shown in Figure 2. This distribution appears to be heavy-tailed with α estimated at 1.5, the same as for the unique file size distribution,

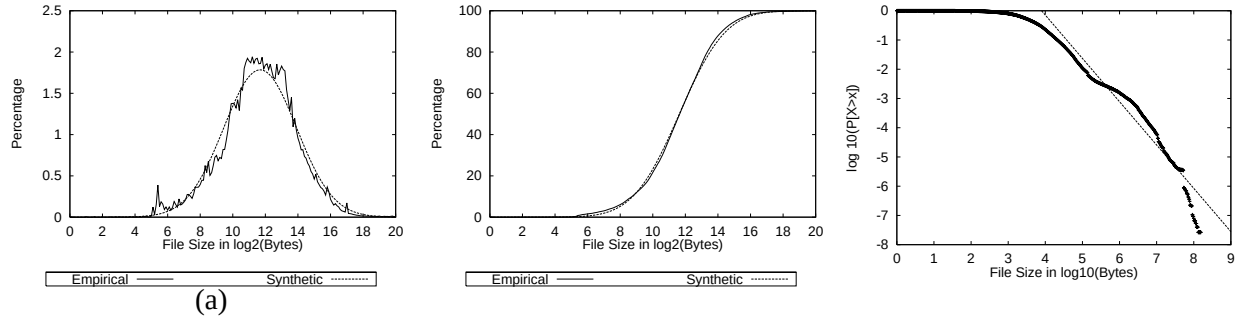
4.5 Usage

Figure 3(a) shows the time-of-day analysis for the HTTP requests and content data traffic. From this figure it is obvious that the proxy workload is affected by the daily routine of the subscribers. The workload is lightest in the early morning when most users are likely sleeping. Usage increases throughout the morning and afternoon, with flat periods around the lunch and dinner hours. Peak usage occurs in the evening, when most subscribers are presumably home from school or work and have time to use their computers. Figure 3(b) shows the breakdown of the proxy workload by the day of the week on which the requests were made. As with the time-of-day results, the day-of-week usage is obviously affected by the routines of the users. For example, the peak usage occurs on weekends when most subscribers are likely at home.

Figure 4(a) shows the level of proxy usage over time. This figure reveals that the number of HTTP requests per day and the daily volume of content data increased during the five month measurement period. Note that the y-axis in Figure 4 ia logarithmic, so even a slight increase in the slope indicates significant growth, During this time the number of HTTP requests per day increased by 83%, or an average of 12.8% per month, while the volume of HTTP content data grew by 92% (13.9% per month).

Table 8: Successful Transfer Size Information by File Type

Item	All Xfers	HTML	Images	Audio	Video	Format	Text	Comp	Exe	Uncache	Other
Number (000s)	75,617	9,387	55,284	454	127	9	117	134	42	6,510	3,553
Mean (bytes)	14,660	5,721	9,540	95,694	1,732,027	252,666	12,319	480,076	2,192,718	8,018	14,466
Median (bytes)	3,450	2,625	3,390	41,638	1,076,836	73,000	779	58,962	776,877	4,835	4,760
Maximum (MB)	148	12.7	89.2	28.5	148	22.1	27.7	86.0	74.1	123	49.7
Total Transfer (GB)	1,032	50	491	40	205	2	1	60	86	49	48



(a)

Figure 2 Analysis of Successful Transfer Size Distribution: (a) Frequency; (b) Cumulative Frequency

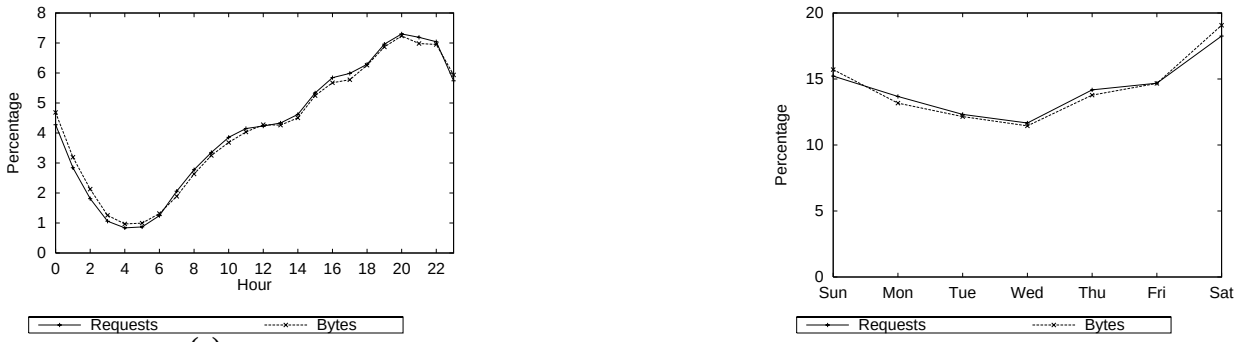


Figure 3 Analysis of Proxy Usage: (a) Time-of-Day; (b) Day-of-Week

Figure 4(b) provides some insights into the growth of the daily proxy usage. This figure shows the number of unique files requested per day, the number of distinct origin servers contacted per day, and the number of unique client IP addresses issuing requests per day. The number of unique files grew at a rate of 14.6% per month (98% overall), while the number of origin servers contacted daily increased by 13.6% per month, or 90% for the five month period. During this time the number of clients rose by 11.7% per month (74% overall). We believe that much of the growth in the workload is due to an increase in the number of clients over time (i.e., new subscribers). This also leads to an increase in both the unique files requested per day and the number of distinct servers contacted. We believe that the growth in the workload is partially due to existing clients using the service on a more regular basis. Since the growth in content data exceeded that of the number of requests, we speculate that users began to download more large files as the subscribers became more comfortable using the Web.

4.6 Recency of Reference

Most Web proxy caches in use today utilize the Least Recently Used (LRU) replacement policy. This policy works best when the access stream exhibits strong temporal locality (i.e., recency of reference). This means that files that have recently been referenced are likely to be referenced again in the near future. To measure the temporal locality present in this data set we utilize the standard LRU stack-depth analysis [1][4]. This analysis determines the depth in the stack at which re-references occur. Reference streams which exhibit a high degree of temporal locality will have a small average stack depth while streams with a low degree of temporal locality will have high average stack depths. In order to compare the degree of temporal locality across data sets we normalize the stack depth by dividing by the number of unique files requested.

Table 9 presents the results of the stack depth analysis. Due to the presence of requests to the ISP server in this data set, the degree of temporal locality is stronger than has been reported for other proxy workloads. For example, Barford *et al.* reported a normalized mean stack depth of 0.2340 for a recent proxy workload [7]. However, the degree of temporal locality reported in Table 9 is still significantly weaker than that of Web servers

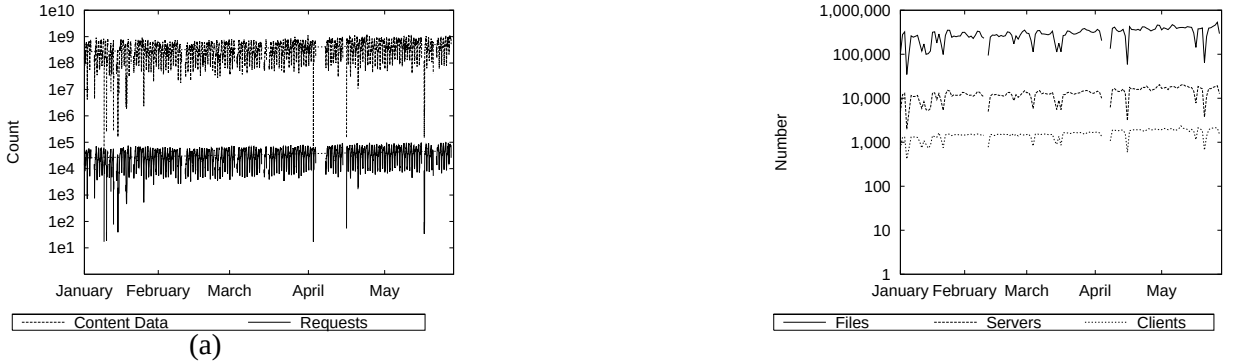


Figure 4 Growth Trends: (a) Proxy Workload; (b) Unique Files, Origin Servers and Clients

(e.g., the normalized mean stack depth for the World Cup Web site was 0.015 [2]).

Table 9: Stack Depth Analysis for HTTP Requests

Median Stack Depth	60,564
Mean Stack Depth	639,412
Standard Deviation	1,489,815
Maximum Stack Depth	16,102,789
Normalized Median Depth	0.004
Normalized Mean Depth	0.04

4.7 Frequency of Reference

Several recent studies, including [4],[8],[13] and [21], have found that some Web objects are substantially more popular than others. That is, Web referencing patterns are non-uniform. We perform two analyses to determine if this characteristic is also present in the data set under study. One analysis examines the concentration of references among the unique files. In this analysis the unique files are sorted into decreasing order based on the number of times that they were requested. We then determine the fraction of all client requests for each of these files. The second analysis determines a file's popularity. The unique files are again sorted into decreasing order based on the number of times they were requested. Each file is then assigned a rank, with rank 1 given to the file with the most references and rank ***N*** (assuming ***N*** unique files) granted to the file with the fewest requests. The results of these analyses are shown in Figure 5.

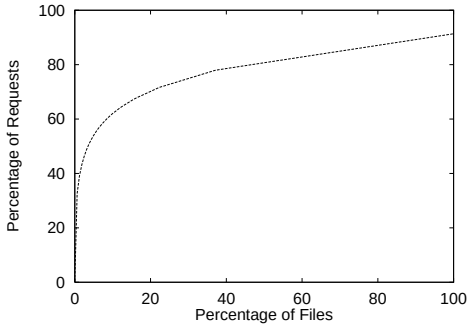
Figure 5(a) indicates that the referencing patterns for HTTP files in this data set are definitely non-uniform. Three distinct groups of files can be identified in this figure: *extremely popular* files (the top 1% of the unique files received 39% of all client requests); *moderately popular* files (the top 37% of all unique files received 78% of the requests); and *unpopular* files (the remaining 63% of unique files were requested only a single time; we refer to these as "one-timers" [4]). The presence of large numbers of one-timers has been observed in numerous other proxy workloads [5][19][20][21]. This characteristic is significant as there is no benefit in caching a one-timer. Cache performance could be improved if these files could be readily identified so that they would not be stored in the cache. A more thorough discussion of one-timers is provided by Mahanti and Williamson [21].

Figure 5(b) indicates the popularity of individual files in the data set. This distribution appears to be Zipf-like with α estimated at 0.79 (a thorough discussion of Zipf-like distributions is provided in [8]). Similar observations have been made for other proxy workloads [7][8][21].

4.8 Origin Servers

Several recent studies have examined the reference patterns to origin servers in proxy workloads [8][24]. These studies found that some of these workloads exhibit non-uniform referencing patterns to origin servers. In this section we analyze our data set for this characteristic.

The results in Figure 6 indicate that the reference patterns to origin servers are non-uniform for the data set under study. Figure 6(a) shows the concentration of client requests to all of the unique origin servers contacted during the measurement period. The disparity between the extremely popular sites and the moderate and unpopular sites is much greater than was the case with the unique files. For example, the top 1% of origin servers (about 2,500) were the intended recipients of 65% of all client requests (caching at the proxy reduces the number that actually reach these servers). The top 10% of the distinct origin servers were the target of 91% of



(a)

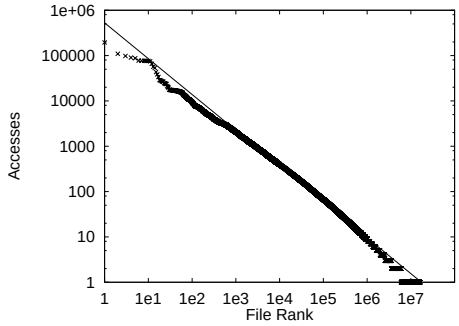


Figure 5 Frequency of Reference Analysis: (a) Concentration of References; (b) File Popularity

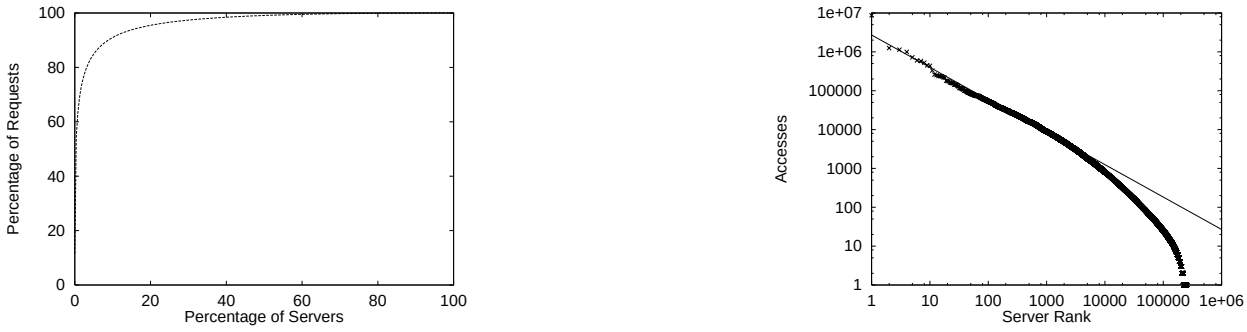


Figure 6 Analysis of Reference Patterns to Origin Servers: (a) Concentration of References; (b) Server Popularity

all client requests. Figure 6(b) shows the number of requests that clients issued for each of the unique servers. For example, more than 500,000 requests were issued for each of the top ten servers. Figure 6(b) appears to be Zipf-like for the servers that were the intended recipients of more than 1,000 requests. For this portion of the graph we estimate α at 0.83.

We also examined the temporal locality in reference stream to the set of origin servers accessed. Since clients typically issue multiple requests to a site within a short period of time (e.g., seconds or minutes) we expected to find a higher degree of locality than for the references to unique files. Table 10 reveals that this is indeed the case.

Table 10: Stack Depth Analysis for Origin Servers

Median Stack Depth	4
Mean Stack Depth	725
Standard Deviation	6,750
Maximum Stack Depth	252,909
Normalized Median Depth	0.00002
Normalized Mean Depth	0.003

4.9 Turnover

In this section we examine the turnover that occurs in the set of *active documents*. In other words, we want to determine how the set of files that users are interested in changes over time. This characteristic is important as one of the responsibilities of a cache management policy is to identify the set of active documents and keep a cached copy. If the active set changes with time then the cache management policy must be able to adjust accordingly.

Our analysis consists of selecting a set of popular files and observing how that set changes in different fixed-length intervals of the data set. Figure 7 shows the results when the 500 most popular files from the first day of the measurement period are compared to the top 500 files for each subsequent day in the data set. The graph indicates the percentage of files that are common to the active sets from Day 1 and Day *X*. Figure 7 reveals that some files have very short popularity, as only about half of the popular objects from Day 1 remain popular on the following days. As time passes fewer and fewer of the objects that were popular from the first day remain popular. However, some of the objects exhibit long-term popularity, as even after five months 20% of the active set from Day 1 are still popular. We also performed this test for different starting days. The results were very similar to Figure 7. Mahanti and Williamson have observed similar behaviour in the proxy workloads that they examined [21].

4.10 Proxy Performance

Our final analysis examines the performance of the proxy for several different cache actions: cache hits; cache hits requiring origin server validation; cache misses; and uncacheable requests (must be handled by the origin server). We measure the performance of these actions by the bandwidth received for the response. The bandwidth is calcu-

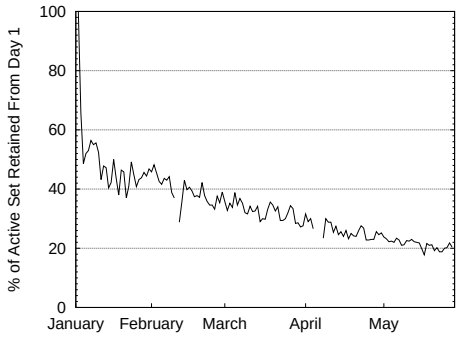


Figure 7 Turnover in the Set of Active Documents

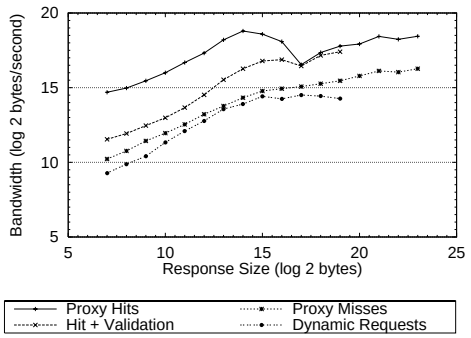


Figure 8 Achieved Bandwidth for Various Cache Actions

lated by dividing the response size by the total response time recorded in the access log.

Figure 8 compares the median achieved bandwidth for the different cache actions when transferring a response of a given size. Unfortunately the recorded response time is missing the time to send the last TCP window (32 KB or 2^{15} bytes on the proxy we examined). This causes our results to overestimate the actual bandwidth for files less than 32 KB, particularly for the cache hits. Since the response time for cache misses is dominated by the time it takes to get the response from the origin server, these results are not affected as much. Despite these problems, Figure 8 still indicates that the proxy cache was able to significantly improve the "user experience" (at least for large transfers), as the achieved bandwidths for cache hits are 2-4 times higher than for cache misses. As expected the bandwidths for uncacheable responses are the lowest of all the cache actions. These responses incur an extra penalty as the origin server must create the response rather than just return a static file.

5 Summary and Conclusions

This paper has presented a detailed workload characterization study of a busy Web proxy server. Our results suggest that user access patterns change when the client's connection to the Internet is not a bottleneck. In particular, users appear to be more willing to request extremely large files (e.g., tens or hundreds of MB in size) when access bandwidths are increased substantially. This behaviour places even more load on Web servers and Internet backbones. The importance of proxy caching increases under such circumstances.

The results of this workload characterization study revealed numerous characteristics that could impact the performance of a Web cache. Several of these characteristics, including file sizes, recency of reference, frequency of reference and turnover in the set of active files, are important to consider when choosing a cache replacement policy. A number of the other characteristics, such as the protocol, the file type and the origin server referencing patterns suggest that alternative approaches to cache management, such as partitioning, may be appropriate. We address both of these issues in a separate paper [3].

Many of the characteristics we observed in this proxy workload were similar to those found in different environments by other researchers. We believe that some of these similarities are influenced by the current design of Web sites. We speculate that as content providers redesign their sites to target users with high-speed Web access, characteristics like the file type, file size and transfer size distribution will also change, and may vary more substantially between users with high access speeds (e.g., cable modems, DSL) and those with lower speed access (e.g., dialup modems, wireless devices).

In order for caching to be a viable long-term solution for the scalability problems of the Web the majority of client requests must be for cacheable files. During our study we observed that the percentage of cacheable responses increased over time, albeit relatively slowly. Feldmann *et al.* [15], using traces of HTTP request and response headers, have shown that in addition to responses that are dynamically generated (those that were deemed uncacheable in this study) many other responses are also uncacheable at proxy caches due to the headers issued by either the client or the origin server. Obviously this trend, should it proceed unchecked, could severely limit the usefulness of Web caching. We believe that there are two major causes for this trend: the failure of the existing caching architecture to provide the functionality needed by users, ISPs and content providers, and a lack of understanding regarding how to properly use the existing technologies. Much effort is required to address both of these issues.

Acknowledgments

The authors would like to thank Mike Rodriquez of Hewlett-Packard Laboratories and all the people in HP's Telecommunication Platforms Division who supplied us with the access logs used in this research.

References

[1] V. Almeida, A. Bestavros, M. Crovella and A. de Oliveira, "Characterizing Reference Locality in the WWW", Proceedings of the *1996 International Conference on Parallel and Distributed Information Systems (PDIS '96)*, pp. 92-103, December 1996.

[2] M. Arlitt and T. Jin, "Workload Characterization of the 1998 World Cup Web Site", Hewlett-Packard Laboratories Technical Report HPL-99-35, February 1999.

[3] M. Arlitt, L. Cherkasova, J. Dilley, R. Friedrich and T. Jin, "Evaluating Content Management Techniques for Web Proxy Caches", *Proceedings of the 2nd Workshop on Internet Server Performance (WISP '99)*, Atlanta, GA, May 1999.

[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] M. Baentsch, L. Baum, G. Molter, S. Rothkugel, and P. Sturm, "Enhancing the Web's Infrastructure: From Caching to Replication", *IEEE Internet Computing*, Vol. 1, No. 2, pp. 18-27, March-April 1997.

[6] P. Barford and M. Crovella, "A Performance Evaluation of HyperText Transfer Protocols", *Proceedings of ACM SIGMETRICS '99*, Atlanta, GA, May 1999.

[7] P. Barford, A. Bestavros, A. Bradley and M. Crovella, "Changes in Web Client Access Patterns", to appear in *World Wide Web Journal*, Special Issue on Characterization and Performance Evaluation, 1999.

[8] 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.

[9] P. Cao and S. Irani, "Cost-Aware WWW Proxy Caching Algorithms", *Proceedings of USENIX Symposium on Internet Technologies and Systems (USITS)*, Monterey, CA, pp. 193-206, December 1997.

[10] M. Crovella and M. Taqqu, ''Estimating the Heavy Tail Index from Scaling Properties", to appear in *Methodology and Computing in Applied Probability*, Vol. 1, November 1, 1999.

[11] M. Crovella and L. Lipsky, "Long-Lasting Transient Conditions in Simulations with Heavy-tailed Workloads", *Proceedings of the 1997 Winter Simulation Conference*, 1997.

[12] M. Crovella and A. Bestavros, "Self-Similarity in World-Wide Web Traffic: Evidence and Possible Causes", *IEEE/ACM Transactions on Networking*, Vol. 5, No. 6, pp. 835-846, December 1997.

[13] 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.

[14] B. Duska, D. Marwood, and M. Feeley, "The Measured Access Characteristics of World-Wide Web Client Proxy Caches", *Proceedings of USENIX Symposium of Internet Technologies and Systems (USITS)*, Monterey, CA, pp. 23-35, December 1997.

[15] A. Feldmann, R. Cáceres, F. Douglis, G. Glass and M. Rabinovich, "Performance of Web Proxy Caching in Heterogeneous Bandwidth Environments", *Proceedings of IEEE Infocom '99*, New York, NY, pp. 107-116, March 1999.

[16] H. Frystyk-Nielsen, J. Gettys, A. Baird-Smith, E. Prud'hommeaux, H. Wium-Lie and C. Lilley, "Network Performance Effects of HTTP/1.1, CSS1 and PNG", *Proceedings of ACM SIGCOMM '97*, Cannes, France, September 1997.

[17] S. Gribble and E. Brewer, "System Design Issues for Internet Middleware Services: Deductions from a Large Client Trace", *Proceedings of USENIX Symposium on Internet Technologies and Systems (USITS)*, Monterey, CA, pp. 207-218, December 1997.

[18] V. Jacobson, "How to Kill the Internet", *SIGCOMM '95 Middleware Workshop*, Cambridge, MA, August 1995.

[19] M. Kurcewicz, W. Sylwestrzak and A. Wierzbicki, "A Filtering Algorithm for Proxy Caches", *3rd International Web Cache Workshop*, http://wwwcache.ja.net/events/workshop/.

[20] P. Lorenzetti and L. Rizzo, "Replacement Policies for a Proxy Cache", Technical Report, Universita di Pisa, December 1996.

[21] A. Mahanti and C. Williamson, "Web Proxy Workload Characterization", Technical Report, Department of Computer Science, University of Saskatchewan, February 1999.

[22] S. Manley and M. Seltzer, "Web Facts and Fantasy", *Proceedings of USENIX Symposium on Internet Technologies and Systems (USITS)*, Monterey, CA, pp. 125-133, December 1997.

[23] J. Mogul, "The Case for Persistent-Connection HTTP", *Proceedings of ACM SIGCOMM '95*, Cambridge, MA, pp. 299-313, 1995.

[24] Network Appliance White Paper, "NetCache Web Caching", available at `http://www.netapp.com/products/level3/netcache/webcache.html`.

[25] V. Paxson, "Empirically-Derived Analytic Models of Wide-Area TCP Connections", *IEEE/ACM Transactions on Networking*, Vol. 2, No. 4, pp. 316-3336, August 1994.

[26] S. Williams, M. Abrams, C. Standridge, G. Abdulla, and E. Fox, "Removal Policies in Network Caches for World-Wide Web Documents", *Proceedings on ACM SIGCOMM '96*, Stanford, CA, pp. 293-305, August 1996.

[27] World-Wide Web Consortium, "Replication and Caching Position Statement", August 1997. Available at `http://www.w3.org/Propogation/activity.html`