Characteristics of WWW Client-based Traces

Carlos R. Cunha   Azer Bestavros   Mark E. Crovella

Computer Science Department
Boston University
111 Cummington St, Boston, MA 02215

{carro,best,crovella}@cs.bu.edu

BU-CS-95-010

July 18, 1995

Abstract

The explosion of WWW traffic necessitates an accurate picture of WWW use, and in particular requires a good understanding of client requests for WWW documents. To address this need, we have collected traces of actual executions of NCSA Mosaic, reflecting over half a million user requests for WWW documents. In this paper we describe the methods we used to collect our traces, and the formats of the collected data. Next, we present a descriptive statistical summary of the traces we collected, which identifies a number of trends and reference patterns in WWW use. In particular, we show that many characteristics of WWW use can be modelled using power-law distributions, including the distribution of document sizes, the popularity of documents as a function of size, the distribution of user requests for documents, and the number of references to documents as a function of their overall rank in popularity (Zipf's law). Finally, we show how the power-law distributions derived from our traces can be used to guide system designers interested in caching WWW documents.

1 Introduction

The recent proliferation of new information services on the Internet along with the enormous potential for commercial use of the Internet has increased the need for efficient protocols to reduce Internet traffic. The introduction of the World Wide Web (WWW) and the explosion in network traffic attributed to it [9] means that an accurate picture of WWW traffic is extremely important for evaluating various bandwidth-reduction techniques — such as information caching, dissemination, and discovery protocols. Records of traffic to any particular server are readily available, as each server typically logs the requests it serves. However, server logs do not reflect the access patterns of individual users. Thus, they have been used to evaluate system-level performance metrics such as network bandwidth demand and server load, but they cannot be easily used to study user-level performance metrics such as service time. Client-based traces, in contrast, have limited value in the evaluation of system-level performance metrics or the study of global properties, but are central in evaluating user-level performance metrics and access patterns.

Unfortunately, client-based traces of WWW traffic are not widely available. To address this need, and to support our own research into client-level techniques for WWW traffic reduction, we captured a large number of client traces, recording the document reference patterns and times resulting from real users accessing the WWW.

In this paper we describe our method of trace collection, which is based on instrumenting the WWW browser *NCSA Mosaic*. In addition, we present a descriptive statistical summary of the traces we collected, which identifies a number of trends and reference patterns in WWW use. In particular, we show that many characteristics of WWW use can be modelled using power-law distributions, including the distribution of document sizes, the popularity of documents as a function of size, the distribution of user requests for documents, and the number of references to documents as a function of their overall rank in popularity (Zipf's law). In addition, we show how summary statistics derived from our traces can be used to guide system designers interested in caching WWW documents.

To our knowledge, these are the first such traces generally available to the research community. The traces are available to the public via anonymous FTP.[1]

2 Data Collection

The first step in understanding patterns of WWW use is the collection of trace data. Previous file-oriented studies have focused on reference patterns established based on logs of proxies [8, 14], or servers [13]. The authors in [3] captured client traces, but they concentrated on events at the user interface level in order to study browser and page design. In contrast, our goal in data collection was to acquire a complete picture of the reference behavior and timing of user accesses to the WWW. To do this, we modified a WWW browser so as to log all user accesses. The browser we used was Mosaic, since its source was publicly available and permission has been granted for using and modifying the code for research purposes. A complete description of our data collection methods and the format of the logfiles is given in [6]; here we only give a high-level summary.

We modified Mosaic to record the Uniform Resource Locator (URL) [1] of each file accessed by the Mosaic user, as well as the time the file was accessed and the time required to transfer the file from its server (if necessary).[2] For completeness, we record all URLs accessed whether they were served from Mosaic's cache or via a file transfer; however the distinction is preserved in the collected data.

At the time of our study (November 1994 through February 1995) Mosaic was the WWW browser preferred by nearly all users at our site. Hence our results reflect the behavior of nearly all the WWW users at our site. Since the time of our study, the preferred browser has become Netscape [5], which is not available in source form. As a result, capturing an equivalent set of WWW user traces at the current time would be significantly more difficult.

In this paper we will refer to a single execution of Mosaic as a *session.* We will refer to the record of all URLs accessed in a session as a *trace.* Each trace is stored in a separate file called a *log.* Any file pointed to by a URL we call an *object*; we refer to a collection of objects that are displayed together in a page of the browser as a *document.*

2.1 Trace Strategy

To avoid extensive internal modifications to Mosaic, we kept no extra information in Mosaic internal data structures. As trace information was produced, it was written to log files on a local disk. At the end of a session the log was transferred to a central machine where it was stored and later processed to generate the statistics.

To instrument Mosaic we identified the routines that are invoked during requests for an object. This required source changes to the files `main.c`, `img.c`, `mo-www.c`, `history.c`, `gui-documents.c`, `HTAccess.c`, `HTFTP.c`, `HTFormat.c`, `HTGopher.c`, `HTNews.c`, `HTTP.c`, and

[1] From `ftp://cs-ftp.bu.edu/techreports/95-010-web-client-traces.tar.gz`.

[2] We recognized, as discussed in [4], that, technically, a URL does not uniquely identify a document under the HTTP protocol; but like those authors we adopt this simplifying convention.

`HTML.c`. A large number of files was modified because each module responsible for one of the possible transfer protocols (FTP, HTTP, *etc.*) had to be modified to write to disk the URL, the size in bytes of the object, the time the request was issued, and how long it took to retrieve the object. In addition we had to include 2 extra routines: one to verify if logs were requested; and a second to transmit the logs to the central machine.

We now realize that the decision not to maintain information inside Mosaic was not the best one, since the version of Mosaic we used has the following behavior when loading a document: after loading a document, Mosaic recalculates the display space required. If the whole document fits in the display area, Mosaic reloads the document again after modifying the display not to show the sliding rule. This reload causes an extra call to the routines that display texts and in-line images. As a result we need to carefully distinguish between the original display of an object and its redisplay if the sliding rule is eliminated. To address this, an additional tool was developed for log post processing. All of the published logs have been processed in this manner to remove artifacts of Mosaic's document re-display.

2.2 Types of Traces

We generated logfiles in three formats, which we called: *condensed, window* and *structure*. We designed the log files for ease of post-processing, so some information is contained in more than one log format. The condensed logs contain the sequence of object requests (again, whether the object was served from the local cache or from the network). The window logs summarize the sequence of documents viewed in each window of the browser (since the browser can present multiple windows). Finally, the structure logs contain the internal structure of each document requested (consisting of the URLs for the textual part, any inlined images, and the embedded links).

The log files are named according to their contents. Each name consists of 3 segments separated by dots. The first segment describes the type of trace it contains: `con`, `win` and `mos`, for condensed, window and Mosaic document structure, respectively; this concatenated with a user id number. User ID numbers are converted from Unix `UIDs` via a one-way function that allows user IDs to be compared for equality but not to be easily traced back to particular users. The next segment consists of the the machine on which the session took place. The last segment consists of the time (in seconds since 1/1/70) when the session started.

For example, a file named `con1.cs20.785526125` is a condensed log, and is a log of a session from user 1, on machine *cs20,* starting at time `785526125`. Corresponding to this file there would also exist a window and a structure log; their names would differ only their first thee characters.

2.2.1 Format of the Condensed Log

Each line of a condensed log corresponds to a single URL requested by the user; it contains the machine name, the time stamp when the request was made (seconds and microseconds since January 1, 1970), the URL, the size of the document (including the overhead of the protocol) and the object retrieval time in seconds (reflecting only actual communication time, and not including the intermediate processing performed by Mosaic in a multi-connection transfer). An example of a line from a condensed log is:

```
cs20 785526142 920156 "http://cs-www.bu.edu/lib/pics/bu-logo.gif" 1804 0.484092
```

In the condensed logs, lines with the number of bytes equal to 0 and retrieval delay equal to 0.0 mean that the request was satisfied by Mosaic's internal cache. By logging internal cache accesses, one can understand the behavior of Mosaic, in order to mimic its actions or to compare them to those of a different caching strategy. Such an approach was taken in [2].

2.2.2 Format of the Window History Log

The second log file reports the window context of the user when using the browser, *i.e.*, the sequence of documents visited and the windows in which those documents are visited. This tracks behavior such as going "forward" or "backwards" within a window, or "jumps" between windows. This data is useful for behavioral studies, and could be useful in determining an empirical model of user behavior for generating synthetic loads in simulators.

Each line in this file contains: a two digit number separated by a dot, reflecting the window and the document number inside that window; the URL; and the date/time when it was accessed. Here is an example of four lines from a window history log:

```
1.1 "http://cs-www.bu.edu:80/" "Tue Nov 22 12:42:24 1994"
1.2 "http://www.zyxel.com/" "Tue Nov 22 12:46:42 1994"
1.3 "http://www.zyxel.com/2864.html" "Tue Nov 22 12:48:21 1994"
1.2 "http://www.zyxel.com/" "Tue Nov 22 12:53:33 1994"
```

2.2.3 Format of the Document Structure Log

The third log file contains a structural summary of each of the documents viewed by the user. For each document visited, the log contains a summary of the embedded images (`<IMG SRC=...>`) and the embedded references (`<A HREF=...></A>`).

The type of a line in the document structure log is identified by a number in the first position on the line. The number 1 signifies that the line is a URL directly requested by the user. Subsequent lines specify items contained in that URL, continuing until another line type 1 is encountered. The number 2 signifies that the line represents a contained URL, normally an in-line image. The number 3 means that the line represents a reference (anchor). The number 4 is used to indicate that the line represents a redirection; what follows is the redirected URL.

Lines in the document structure log consist of two formats. For type 1 and 2 lines, the format is: the line type; the machine name; the URL; seconds and microseconds since 1/1/70 when the request was made; the round trip retrieval time; and the item's size (including the protocol overhead). If the object's size and time are zeroes, no transfer took place, possibly because the objest was shared by other documents. For type 3 lines, the format is: the line type; the sequence number of the reference within the document (numbered sequentially starting from 0 for each document); and the reference itself. A typical segment of a document structure log is shown in Figure 1.

Sometimes Mosaic is not able to fetch parts of the whole document; for example, some in-line images. This is indicated in the document structure log by the absence of the time and size information (not shown in Figure 1). This was not considered to be a serious problem, since the main objective of this log is to discover document structure.

A possible use of the document structure log is to partially reconstruct the Web graph. With such a graph it could be possible to model access patterns for individual users with the goal of developing an intelligent agent that could help in document prefetching.

2.3 Data Collection Environment

To collect our data we installed our instrumented version of Mosaic in the general computing environment at Boston University's Computer Science Department. This environment consists principally of 37 SparcStation 2 workstations connected in a local network, which is divided in 2 subnets. Each workstation has its own local disk; logs were written to the local disk and subsequently transferred to a central repository.

32 workstations were primarily used by undergraduates in the Computer Science program. We refer to these as the *Room B19* workstations (and traces and users) from the name of the room they are in. 5 workstations were primarily used by graduate students. These are the *Room 272* workstations (and users and traces).

```
1 cs20 "http://cs-www.bu.edu/" 785526141 715240 0.715116 1935
2 cs20 "http://cs-www.bu.edu/lib/pics/bu-logo.gif" 785526142 920156 0.484092 1804
2 cs20 "http://cs-www.bu.edu/lib/pics/bu-label.gif" 785526143 788613 0.470034 716
3 0 "gopher://software.bu.edu/11/Resources%20At%20Your%20Fingertips/
3 1 "http://web.bu.edu/pagetwo.html"
3 2 "http://cs-www.bu.edu/faculty/bulletin/Home.html"
3 3 "http://cs-www.bu.edu/courses/Home.html"
3 4 "http://cs-www.bu.edu/labs/Home.html"
3 5 "http://cs-www.bu.edu/techreports/Home.html"
3 6 "http://cs-www.bu.edu/colloquium/Home.html"
3 7 "http://cs-www.bu.edu/faculty/Home.html"
3 8 "http://cs-www.bu.edu/staff/Home.html"
3 9 "http://cs-www.bu.edu/students/grads/Home.html"
3 10 "http://cs-www.bu.edu/students/acm/Home.html"
3 11 "http://cs-www.bu.edu/help/Home.html"
3 12 "http://cs-www.bu.edu/cgi-bin/finger"
3 13 "http://cs-www.bu.edu/misc/wusage/usage/index.html"
3 14 "http://cs-www.bu.edu/pointers/Home.html"
3 15 "http://cs-www.bu.edu/faculty/best/Home.html"
1 cs20 "http://www.zyxel.com/" 785526307 307174 13.194868 1664
2 cs20 "http://www.zyxel.com/zyxellogo.gif" 785526320 644309 4.094732 1512
2 cs20 "http://www.zyxel.com/line-green-blue-grad.gif" 785526325 224441 4.119299 2547
2 cs20 "http://www.zyxel.com/group.gif" 785526329 874274 69.037422 32994
2 cs20 "http://www.zyxel.com/line-green-blue-grad.gif" 0 0 0.0 0
3 0 "http://www.zyxel.com/b.html"
3 1 "http://www.zyxel.com/bp.html"
3 2 "http://www.zyxel.com/e.html"
3 3 "http://www.zyxel.com/ep.html"
3 4 "http://www.zyxel.com/sp.html"
3 5 "http://www.zyxel.com/p.html"
3 6 "http://www.zyxel.com/r.html"
3 7 "http://www.zyxel.com/re.html"
3 8 "http://www.zyxel.com/cell.html"
3 9 "http://www.zyxel.com/2864.html"
1 cs20 "http://www.zyxel.com/2864.html" 785526419 767702 14.792571 8191
2 cs20 "http://www.zyxel.com/zyxellogo.gif" 0 0 0.0 0
2 cs20 "http://www.zyxel.com/line-green-blue-grad.gif" 0 0 0.0 0
2 cs20 "http://www.zyxel.com/um288ec.gif" 785526434 936426 63.574387 44393
2 cs20 "http://www.zyxel.com/line-green-blue-grad.gif" 0 0 0.0 0
3 0 "http://www.zyxel.com/index.html#1"
```

Figure 1: Typical entries in a document structure log file.

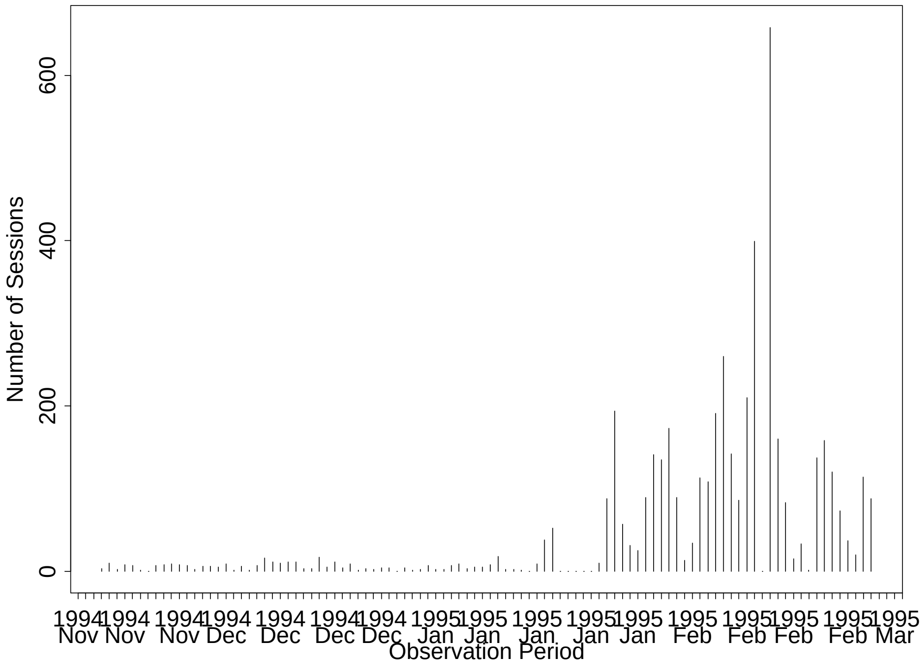


Figure 2: Histogram of the number of sessions per day.

We began by collecting data on the Room 272 workstations only, while testing our data collection process. This period lasted from 21 November 1994 until 17 January 1995. When we were statisfied that data collection was occurring correctly, we extended the data collection process to include the 32 undergraduate workstations; data collection then took place until 8 May 1995. Since Mosaic ceased to be the dominant browser in use by early March 1995, all of the statistics presented in subsequent sections of this report are based only on data from the period 21 November 1995 through 28 February 1995. When our statistics could be affected by the difference in collection strategies during this timeframe, we consider the 272 and B19 data separately.

3 Statistical Summary of the Traces

During the data collection period a total of 4,700 sessions were traced, representing a population of 591 different users. A histogram of the number of sessions per day is shown in Figure 2. The Figure shows considerable variation over time. The enabling of tracing in B19 is clearly visible in the middle of the plot. However, even within each data collection regime (before or after B19 was enabled) there is considerable day-to-day variation in the WWW demand at our site as measured by user sessions.

3.1 Population Observed

The users of the graduate terminal room, Room 272, consisted mainly of graduate students. There were a total of 42 users, and 527 sessions in the Room 272 data. Room 272 was open 24 hours per day.

The population using Room B19 encompasses a broader spectrum of users, including both students doing homework and less-focused users. There were 558 different Room B19 users,

Sessions	272	B19
Minimum	1	9
Maximum	18	658
Mean	5.27	97.05
St.Dev.	4.16	118.17

Table 1: Number of sessions in 272 and B19 daily.

Users	Total	272	B19
Minimum	1	1	1
Maximum	71	71	38
Mean	7.95	12.55	7.48
St.Dev.	8.17	18.58	6.58

Table 2: Number of sessions per user in 272 and B19.

responsible for 4173 sessions. Room B19 was open from 9 am to 10 pm from Monday to Friday, and 12 pm to 6 pm on weekends.

Table 1 shows summary statistics on the number of sessions in each room on a daily basis. The table reflects the fact that the larger number of workstations in B19 led to a much larger number of sessions per day in that room.

Table 2 shows summary statistics on the number of sessions per user in each room. The table shows that members of the more general population in B19 used the Web fewer times overall, while the primarily graduate student population in 272 used the Web much more often.

Finally, Table 3 shows the distribution of requests among the different access protocols. The row *Queries* refers to requests containing `cgi-bin` in the URL. The first three columns include all the references, whether they were served by a WWW server or from Mosaic's internal cache. The second three columns consider only the requests that were statisfied by a WWW server.

3.2 Remote vs. Local Data Volume

An important distinction is between those requests that went to servers outside the Boston University campus (*Remote*) and those requests that went to BU servers (*Local*). Table 4 summarizes requests based on that distinction.

The table additionally distinguishes between *single* references, which each occur only once in our traces, and *repeated* references, which occur at least twice in our traces. A reference is considered repeated if it occurs twice or more regardless of user or session boundaries. In the Table, the *Unique URLs* row corresponds to the number of distinct URLs while the *All URLs* row corresponds to all references in our dataset. Finally, the *Unique Bytes* row corresponds to bytes pointed to by Unique URLs while the *All Bytes* row corresponds to all bytes transferred in our dataset.

The distinction between Local and Remote documents can be useful, for example, in caching

	All			Cache Misses		
	Total	272	B19	Total	272	B19
`http`	568181	65764	502417	125550	15580	109970
`gopher`	5236	727	4509	3077	383	2694
`ftp`	2358	514	1844	1513	332	1181
Queries	7509	1129	6380	5099	699	4400
Other	7888	110	7778	119	20	99

Table 3: Distribution by Protocol.

	All			Remote			Local		
	Total	Single	Repeated	Total	Single	Repeated	Total	Single	Repeated
Unique URLs	46,830	19,454	27,376	44,251	18,658	25,593	2,579	796	1,783
All URLs	575,775	19,454	556,321	452,864	18,658	434,206	122,911	796	122,115
Unique Bytes (MB)	1,088	794	294	1,037	776	261	51	18	33
All Bytes (MB)	2,714	794	1,920	2,163	776	1,387	551	18	533
Sites	3026			2987			39		
Domains	86								

Table 4: Comparison of Remote vs. Local Requests.

	Freq	Site
Local	97477	cs-www.bu.edu
	9283	www.bu.edu
	4566	web.bu.edu
	3385	lobster.bu.edu
	2184	spiderman.bu.edu
	1513	gopher.bu.edu
	862	conx.bu.edu
	593	acs2.bu.edu
	333	eng.bu.edu
	330	miranda.bu.edu
Remote	129649	akebono.stanford.edu
	23315	american.recordings.com
	14621	csugrad.cs.vt.edu
	9675	www.galcit.caltech.edu
	9177	sunsite.unc.edu
	7149	gnn.com
	5036	www.ncsa.uiuc.edu
	4746	www.timeinc.com
	4531	www.tcp.com
	4202	www.mit.edu

Table 5: Most Popular Sites, Local and Remote.

policies, since with the advent of fast LANs the cost associated with information retrieval inside the same institutional domain may become cheaper compared with retrievals that traverse the wide are network. An example exploration of such a distinction is discussed in [2].

The Table shows that from a total of 575,775 requests, 452,862 were for servers outside of the BU domain. This corresponds to 78.6% of all requests. The requests include references to 3026 different sites, of which 2987 were remote and 39 were local. Table 5 summarizes the 10 top most popular sites, local and remote. In addition, Table 6 lists the most popular top-level domains.

3.3 User Profile and Speed of New Site Discovery

Since we are interested in using user profiles and user past history in directing prefetching schemes, we studied the rate of user access to new objects in the Web. To do so, we plotted diagrams showing the patterns of individual users access to new and previously seen URLs. Two such diagrams are shown in Figure 3. In these figures, the horizontal (x) axis represents successive WWW requests by a single users. Each time the user requests a previously unseen URL, that URL is assigned a new ID, which is shown on the vertical (y) axis. If the user were

Freq	Domain
376306	edu
120784	com
17863	net
9361	org
8190	uk
6267	gov
4547	ch
3914	mil
3725	nl
3547	se

Table 6: Most Popular Top-Level Domains.

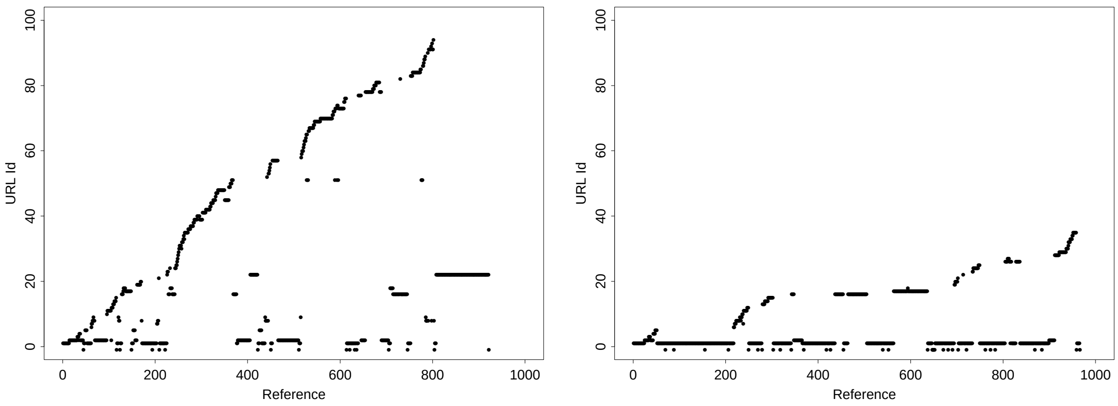


Figure 3: User Profiles. The user on the left navigates through the Web much faster than the user on the right.

continually accessing new URLs (pure "surfing") the diagram would have a slope of 1. If the user were continually accessing the same URL, the diagram's slope would be 0. These figures show that different users exhibit different degrees of temporal locality and make different demands for new URLs, which affects the performance of document caching.

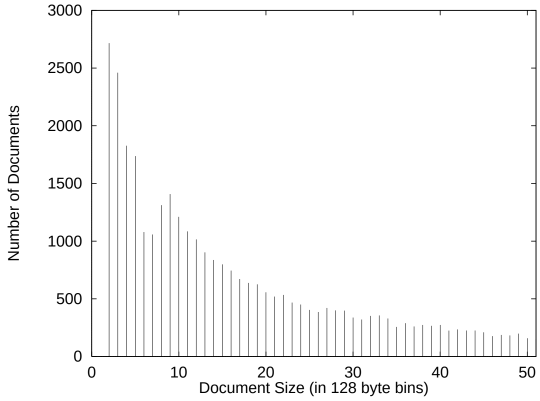
The differences between these diagrams suggest that the slope of each user's URL discovery curve could be used to parameterize a document prefetching scheme.

4 Distributional Characterization of the WWW

A primary goal of our study was to obtain a clear picture of the size distribution of the documents available on the web, of the requests made by users, and of the relationship between document size and popularity.

Previous studies of file sizes in general purpose Unix systems has shown that small sizes are much more common than large sizes [7, 12]. However, in view of the potential use of the Web to publish multimedia objects, documents on the Web could have a significantly different distribution from those in general purpose file systems.

Overall, the most striking aspect of our studies of the WWW is the dominance of the power



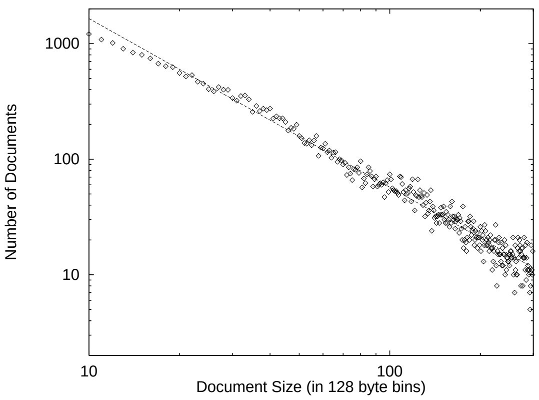


Figure 4: Document size distribution; Left: Linear scale of sizes up to 6400 Bytes; Right: Log-Log scale of sizes 1280 bytes or more.

law (or Pareto) distribution. The shape of the power-law distribution is a hyperbola; with parameter α its probability mass function is

$$p(x) = \alpha k^{\alpha} x^{-\alpha-1}$$

and its cumulative distribution function is given by

$$P[X \leq x] = 1 - (k/x)^{\alpha}, \quad \alpha, k \geq 0, \ x \geq k.$$

Power-law distributions have a number of properties that are qualitatively different from distributions more commonly encountered such as the exponential, normal, or Poisson distributions. If $\alpha \leq 2$, then the distribution has infinite variance; if $\alpha \leq 1$ then the distribution has infinite mean. So depending on α, an arbitrarily large portion of the probability mass may be present in the tail of the distribution — hence the name *heavy-tailed.* In practical terms a random variable that follows a heavy-tailed distribution can give rise to extremely large values with non-negligible probability.

Figure 4 shows the size distribution of objects referred to at least once in our logfiles. Although this is not a random sample of documents available on the Web, it provides a reasonable estimate of the actual size distribution of Web documents. On the left of the Figure is a histogram of file sizes up to 6400 bytes; on the right is a histogram on a log-log scale of file sizes of 1280 bytes or more.

Figure 4 shows the pronounced hyperbolic distribution of file sizes. The linear fit to the log-transformed data ($y \sim x^{-1.35}$) is very strong, with an R^2 value of 0.96.

A comparison of WWW document sizes with the file size distribution that might be found in a typical Unix file system is instructive. While there is no truly "typical" Unix file system, an aggregate picture of file sizes on over 1000 different Unix file systems is reported in [10]. In Figure 5 we compare the distribution of document sizes we found in the Web with that data. The Figure plots the percentage of all files in exponentially increasing bins, on a log-log scale.

Surprisingly, Figure 5 shows that in the Web, there is a *stronger* preference for small files than in Unix file systems.[3] The Web strongly favors documents in the 256 to 512 byte range, while Unix files are more commonly in the 1KB to 4KB range. More importantly, the tail of the distribution of WWW files is not nearly as heavy as the tail of the distribution of Unix files. Thus, despite the inclusion of multimedia in the Web, we conclude that Web documents are currently more biased toward small files than are typical Unix file systems.

Related to the question of document size distribution are questions involving user preferences: the relationship between size and popularity, and the popularity of individual documents.

[3] However, not shown in the figure is the fact that while there are virtually no Web files smaller than 100 bytes, there are a significant number of Unix files smaller than 100 bytes, including many zero- and one-byte files.

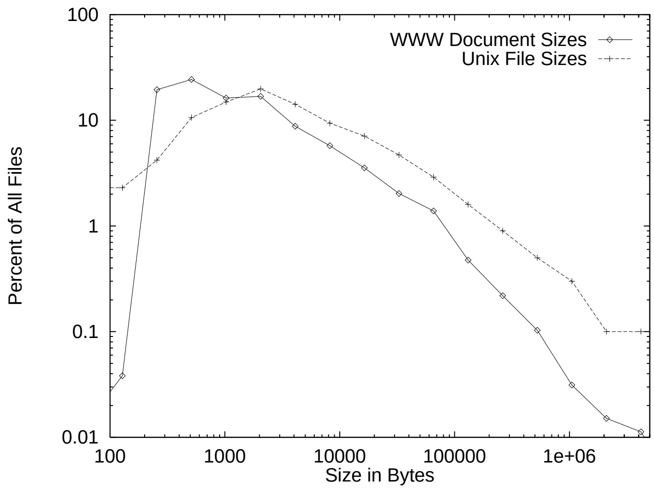


Figure 5: Comparison of Unix File Sizes with WWW File Sizes

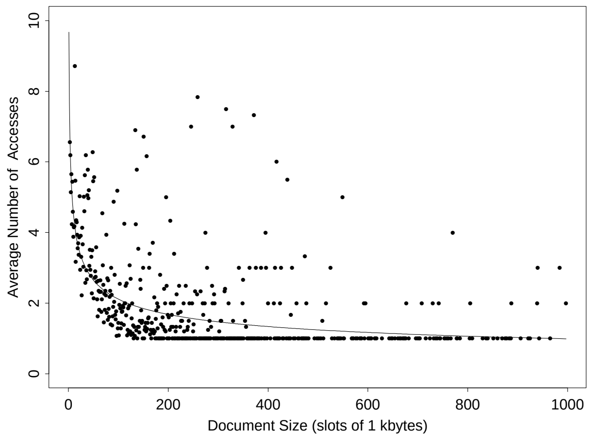


Figure 6: Distribution of Average Number of Requests by File Size.

To explore the influence of user choice on the distribution of documents actually transferred through the Internet, we measured the relationship between the number of times a document is accessed and the size of the document. Figure 6 shows a plot of the average number of times documents of a given size (in 1K bins) were referenced. The data shows that there is an inverse correlation between file size and file popularity. The line shown is a least squares fit to a log-log transform of the data ($y \sim x^{-0.33}$). This fit is statistically significant at a 99.9% level, but only explains a small part of the total variation ($R^2 = 0.32$).

Thus, in addition to the tendency for WWW documents as created to be small, users additionally *prefer* small documents. The combined effect of these two trends is shown in Figure 7. This figure shows the distribution of document requests made by users, by request size. This figure shows that actual document traffic generated by user requests also follows a hyperbolic distribution. The least squares fit line ($R^2 = 0.89$) is $y \sim x^{-1.66}$. Note that the exponent in the distribution of request sizes, -1.66 (indicating a strong preference for transferring small files) is very nearly the sum of the exponent describing user preference, -0.33 (a mild user preference for small files) and the exponent in the document size distribution, -1.35 (a moderate producer

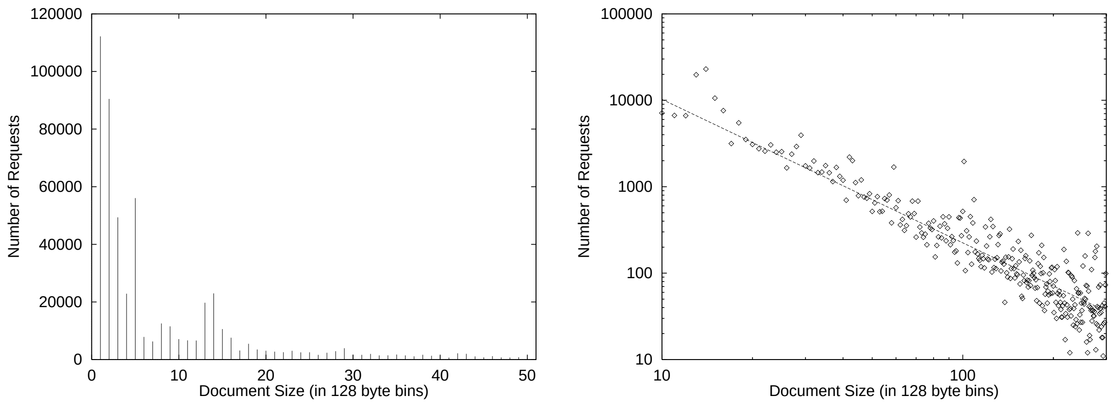


Figure 7: Distribution of Requests for Documents by Size; Left: Linear scale of sizes up to 6400 Bytes; Right: Log-Log scale of sizes 1280 bytes or more.

preference for small files). Thus at the present time, the distribution of transfer sizes is more strongly determined by document sizes than by user preferences, although both contribute.

The final instance of hyperbolic distributions in our data occurs as an instance of Zipf's law [15, discussed in [11]]. Zipf's law was originally applied to the relationship between a word's popularity in terms of rank and its frequency of use. It states that if one ranks the popularity of words used in a given text (denoted by ρ) by their frequency of use (denoted by P) then

$$P \sim 1/\rho.$$

Note that this distribution is parameterless, *i.e.,* ρ is raised to exactly -1, so that the nth most popular document is exactly twice as popular as the $2n$th most popular document. Zipf's law has subsequently been applied to other examples of popularity in the social sciences.

Our data shows that Zipf's law applies quite strongly to documents on the WWW. This is demonstrated in Figure 8 for all 46,830 documents referenced in our logs. The figure shows a log-log plot of references to each document as a function of the document's rank in overall popularity. The tightness of the fit to a straight line is remarkable ($R^2 = 1.00$), as is the slope of the line: -0.986. Thus the exponent relating popularity to rank for WWW documents is very nearly -1, as predicted by Zipf's law.

One interpretation for our file size distribution data is based on file type. It may be that (smaller) HTML files are preferred by users over (larger) images, leading to user preference for small files. To investigate this, we classified WWW objects into 9 categories, based mainly on file extension, as described in Table 7. The distribution of document requests by type is shown in Table 8. This table shows that user preference for small documents is not due to a preference for HTML files over images; in fact, the reverse is true — image files are by far the most commonly requested object, and they tend to be significantly larger than HTML files.

5 Application To Caching Strategies

This section shows how our data can be used in WWW caching strategies. In previous work, we used the raw data from our traces to drive extensive event-driven simulations of client-based caching protocols [2]. However, it is also possible to use the distributional results discussed in the last section to gain considerable insight in designing caching strategies. In this section we show a case study of how our high-level distributions can be used to guide caching policies for WWW documents.

Traditional studies of caching policies such as page replacement policies or cache-line replacement policies have been concerned with fixed-size transfer units. In contrast, caching WWW documents involves variable-sized units; as a result it is interesting to investigate what leverage

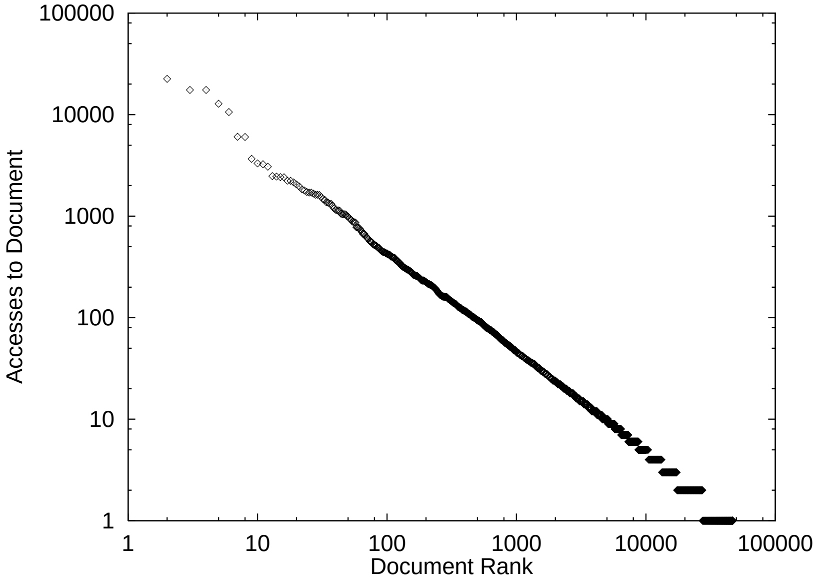


Figure 8: Zipf's Law Applied To WWW Documents

Category	File Extensions
HTML	html, HTML, htm, HTM, shtml, cs
Image	gif, xbm, GIF, jpg, jpeg, gif89, tif, tiff
Sound	au, wav, snd, lha, MOV
Video	mpg, mp2
Text	text, TXT, README, toc, p, ocr, abstract, bitmap
Formatted Document	ps, dvi, ps.gz, ps.Z
Dynamic	pl, cgi, count, also objects containing "?" or "cgi-bin" as part of the path
Archive	hqx, zip, gz
Other	anything that does not match the above listed objects

Table 7: Object Categories.

	HTML	Image	Sound	Video	Text	Formatted Document	Dynamic[4]	Archive	Other
Remote									
Number of Refs.	55351	394666	320	153	438	141	90	85	1657
Pct. of Refs.	9.61	68.55	0.06	0.03	0.08	0.02	0.02	0.01	0.29
Avg. Size (kB)	6.4	13.9	551	792	32.3	358	3.48	4404	191
Local									
No. of Refs.	37053	83340	21	23	271	1929	11	2	221
Pct. of Refs.	6.44	14.47	0.00	0.00	0.05	0.34	0.00	0.00	0.04
Avg. Size (kB)	2.89	8.39	2778	381	1.79	50.6	4.80	907	67.6

Table 8: Document Type Distribution.

can be gained through the use of document size information in caching policies. The distributional data presented in the previous section is well suited to helping answer this question.

Our goal in developing new caching strategies is to reduce the latency of WWW document accesses; thus we need an understanding of the relationship between document size and transfer cost. A complete study of the transfer time as a function of document size is beyond our scope, but we can develop a rough model for the relationship using our trace data. We will assume that document transfer time is the sum of a fixed overhead plus a per-byte cost:

$$T(s) = O + Bs.$$

where s is the size of a document in bytes. For both O and B we will assume the average value that these terms take in our data.[5] These parameters can be inferred from Figure 9, which shows the average transfer time of documents in our traces as a function of their size. Note that as s grows the influence of O is reduced and T/s approaches the constant B.

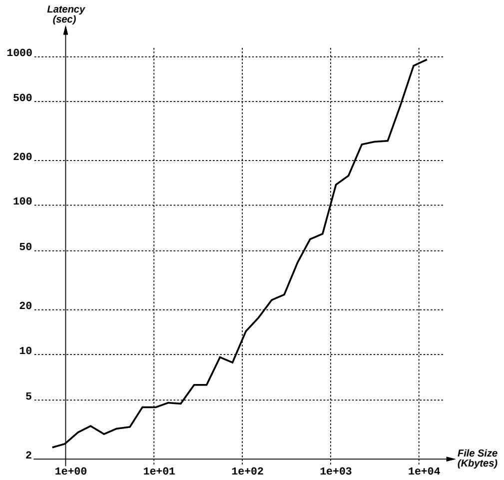


Figure 9: Transfer Time of Documents as a Function of Size

To include size information in a caching strategy, we would like to know the expected total improvement in latency for each byte that is cached, as a function of file size. Using that information we can decide whether to cache a file based on its size, and we can determine how much improvement in total latency can be gained from a size-based caching policy.

The data presented in the previous section provides the tools to answer this question. We consider a proxy-type cache, in which all requests in our traces are forwarded through a local server. Thus we treat all our traces in the aggregate as a single reference stream. Then we can calculate the total improvement in latency for each byte that is cached, L, as:

$$L(s) = \frac{(N(s) - 1)T(s)}{s}$$

where N(s) is the expected number of references to a document of size s, as shown in Figure 6. This equation yields the total document transfer latency that will be avoided by caching a document of size s, per byte of cache space used.

Note that we have assumed that a document, once cached, has zero retrieval cost in the future. Since we can also derive O and B estimates for local documents, we could include the cost of local transfers for future accesses to the cached documents. However, doing so does not significantly change the results that we present.

[5]A rough estimate, but sufficient for our purposes; to improve the accuracy of this approximation we consider only remote documents in this section.

A plot of L as a function of s is shown in Figure 10. The figure is plotted on log-log axes, and for file sizes greater than 1K bytes, is roughly linear. Thus we see that latency savings per byte is asymptotically hyperbolic, as we would expect from the equation for L, since T/s is asymptotically linear, and N is hyperbolic (as discussed in the previous section).

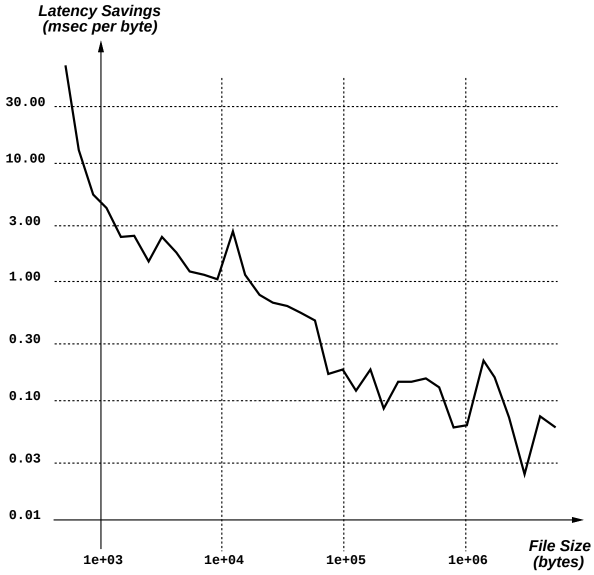


Figure 10: Latency Savings per Byte as a Function of Document Size

The most important aspect of Figure 10 is that it is monotonically decreasing, which means that WWW cache space should be preferentially allocated to smaller files over larger files. The reasons why cache space should be allocated to small files preferentially derive directly from the user preferences shown in Figure 6 and from the relatively large latency savings gained by avoiding the fixed cost overhead O in transferring small files.

However, these curves do not reflect the cost of allocating resources for file caching. In particular, we would like to know how much space will be required, and what the expected benefit will be, if we adopt a policy of caching all files less than a certain size.

To answer this question, we must first find the amount of cache space required to cache all files less than a certain size. This can be done by simply performing a running sum over the distribution shown in Figure 4, times its x axis. The result is shown in Figure 11; in the figure, the x axis is logarithmic while the y axis is linear. The figure shows that (for file sizes greater than 10K) the amount of cache space required increases logarithmically as a function of the largest file size cached. This is reasonable, since if the distribution of remote files is hyperbolic with exponent close to -2, then multiplying by x adds 1 to the exponent and integrating yields the log function. Although we found the distribution of all files to be approximately $y \sim x^-1.35$, the effect of selecting only remote files is to decrease the likelihood of large files, thus moving the exponent in the underlying distribution closer to -2.

Using the resulting distribution of cache sizes as a function of file sizes cached, we can predict the effect of caching using various cache sizes. We replot the data in Figure 10 as follows. Instead of plotting the latency savings per file size, we plot the latency savings against the cache size corresponding to the proper file size. The result is shown in Figure 12. On the left is the function on linear axes; on the right it is plotted on log-log axes.

Finally, we can determine the percent of total possible savings that are gained by caching as a function of cache size allocated. This is simply the running summation of the distribution shown in Figure 12. The resulting summation is shown in Figure 13.

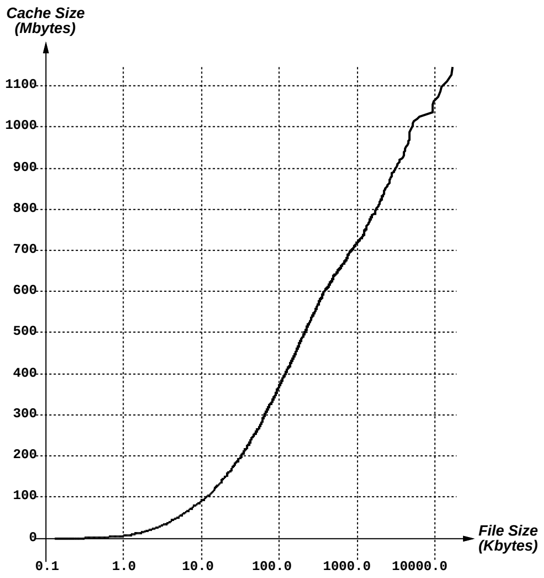


Figure 11: Cache Space Required as a Function of File Sizes Cached

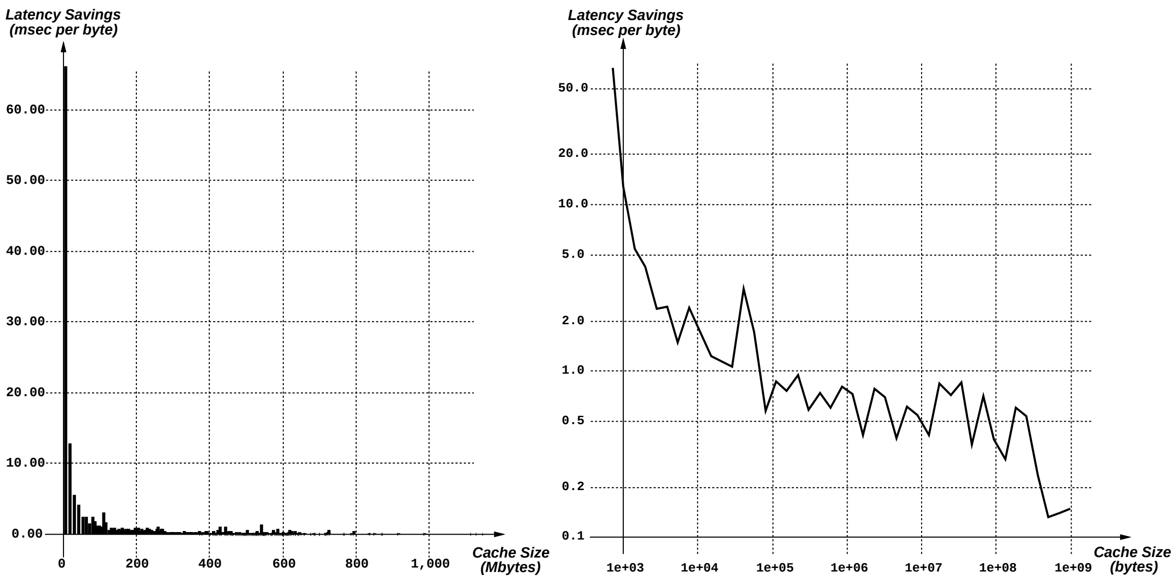


Figure 12: Latency Savings as a Function of Cache Size

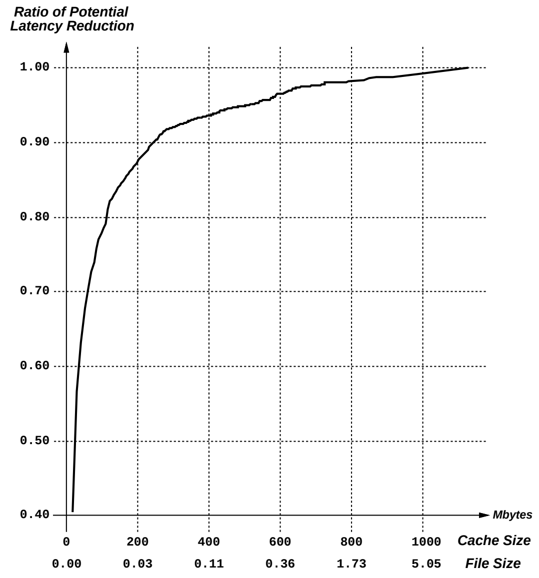


Figure 13: Fraction of Total Latency Savings as a Function of Cache Size

This figure shows the fraction of total latency in our traces that can be saved using a caching policy that caches all documents less than a certain size. The amount of cache space needed is shown on the x axis, and below that, the cutoff size of the documents to be cached for that size cache. The figure shows that with only 400MB of cache space, approximately 95% of maximum possible performance gains can be achieved. At this level, the cutoff size for documents would be approximately 110KB.

More importantly, the figure shows how sharp the improvement is in performance as cache size is increased from 0 to 100MB. In this region the improvement is approximately linear, which means that cache space can be distributed among the various file sizes without great changes in performance. This flexibility means that strategies that choose to cache a few large files (to improve worst-case latency) or many small files (to improve the most common case) are equally attractive.

6 Conclusion

It is clear that the World Wide Web is presenting a challenge to network managers and, moreover, to computer scientists. The growth pattern of the WWW suggests that careful use of the network, CPU, and disk resources that support the Web will become increasingly important over time.

To support our future research on WWW resource management, we have collected and described a large amount of data reflecting actual user accesses to the Web. Our data is unique as it has been collected by instrumenting clients rather than servers, and because it provides insight into demands that Web users make on disk and CPU as well as network resources.

In summarizing our data, we have identified a number of trends and reference patterns in WWW use. In particular, we have shown that many characteristics of WWW use can be modeling using power-law distributions, including the distribution of document sizes, the popularity of documents as a function of size, the distribution of user requests for documents, and the number of references to documents as a function of their overall rank in popularity. In addition, we have shown that even high-level, distributional information about WWW use can be helpful in designing caching strategies for WWW documents.

References

[1] T. Berners-Lee, L. Masinter, and M.McCahill. Uniform resource locators. RFC 1738, `http://www.ics.uci.edu/pub/ietf/uri/rfc1738.txt`, December 1994.

[2] Azer Bestavros, Robert L. Carter, Mark E. Crovella, Carlos R. Cunha, Abdelsalam Heddaya, and Sulaiman A. Mirdad. Application-level document caching in the Internet. In *Proceedings of the Second International Workshop on Services in Distributed and Networked Environments (SDNE'95)*, June 1995.

[3] Lara D. Catledge and James E. Pitkow. Characterizing browsing strategies in the world-wide web. In *Proceedings of the Third WWW Conference*, 1994.

[4] Anawat Chankhunthod, Michael F. Schwartz, Peter B. Danzig, Kurt J. Worrell, and Chuck Neerdaels. A hierarchical internet object cache. Technical Report CU-CS-766-95, Department of Computer Science, University of Colorado - Boulder, March 1995.

[5] Netscape Communications Corp. Netscape navigator software. Available from `http://www.netscape.com`.

[6] Carlos R. Cunha, Azer Bestavros, and Mark E. Crovella. Characteristics of www client-based traces. Technical Report BU-CS-95-010, Boston University Computer Science Department, 1995.

[7] Richard A. Floyd. Short-term file reference patterns in a unix environment. Technical Report 177, Computer Science Dept., U. Rochester, 1986.

[8] Steven Glassman. A Caching Relay for the World Wide Web. In *First International Conference on the World-Wide Web*, CERN, Geneva (Switzerland), May 1994. Elsevier Science.

[9] Merit Network Inc. Nsf network statistics. Available at `ftp://nis.nsf.net/statistics/nsfnet/`, Decemeber 1994.

[10] Gordon Irlam. Unix file size survey — 1993. Available at `http://www.base.com/gordoni/ufs93.html`, September 1994.

[11] Benoit B. Mandelbrot. *The Fractal Geometry of Nature.* W. H. Freedman and Co., New York, 1983.

[12] John K. Ousterhout, Herve Da Costa, David Harrison, John A. Kunze, Michael Kupfer, and James G. Thompson. A trace-driven analysis of the UNIX 4.2BSD file system. Technical Report CSD-85-230, Dept. of Computer Science, University of California at Berkeley, 1985.

[13] James E. Pitkow and Margaret M. Recker. A Simple Yet Robust Caching Algorithm Based on Dynamic Access Patterns. In *Electronic Proc. of the 2nd WWW Conference*, 1994.

[14] Jeff Sedayao. "Mosaic Will Kill My Network!" – Studying Network Traffic Patterns of Mosaic Use. In *Electronic Proceedings of the Second World Wide Web Conference '94: Mosaic and the Web*, Chicago, Illinois, October 1994.

[15] G. K. Zipf. *Human Behavior and the Principle of Least-Effort.* Addison-Wesley, Cambridge, MA, 1949.