Web Server Workload Characterization

John Dilley
Hewlett-Packard Laboratories

Abstract

Use of the Internet and the World Wide Web have increased rapidly over the past few years. HP customers deploying Web servers want to understand how their servers are being used by Internet users, how those patterns of use are changing over time, and what steps they should take to ensure adequate server response to the incoming requests today and in the future. This requires an evaluation of the requests offered to the Web server and the characteristics of the server's response to those requests over a suitably long time interval.

In this report we present the results of a study of one customer's Internet Web server system over a two month period. During that time the traffic to the site increased significantly in terms of incoming requests and outgoing bytes. We examine the request and response types, and characterize the traffic distribution on the basis of request size, response time, and other factors. We conclude with system performance recommendations and identify future directions for our Web research.

1.0 INTRODUCTION

This report presents the workload characteristics of a busy customer Internet World Wide Web Server. The purpose of this study is to obtain a better understanding of today's WWW traffic patterns and to set the stage for analysis of system resource utilization as a function of Web Server workload. By workload we mean both the request stream presented by clients (the work) as well as the server response to the requests (the load). Characterization involves determining and describing the fundamental character of the workload as presented over time. This report describes in detail the requests made by clients to the WWW server and the characteristics of the system's response, including the distribution of response sizes and server response times. We conclude with a set of system performance recommendations for this server.

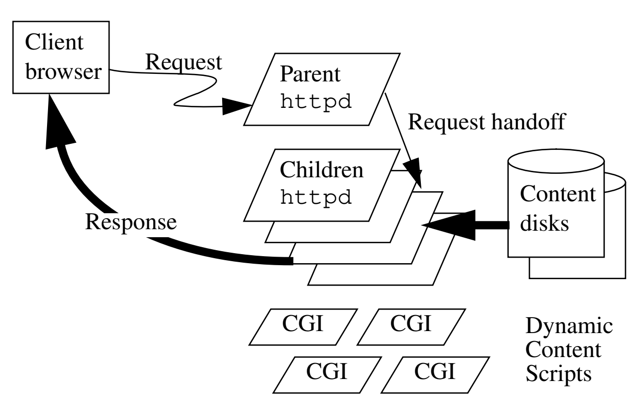
With an understanding of the workload as it evolves over time performance engineers can analyze the demands users of a service are placing on the system and see trends in user behavior over time. One of the benefits of workload characterization is that it allows construction of analytical models of the workload and simulators that emulate client behavior in order to study the performance of similar systems in a controlled lab test environment. In this controlled environment it is possible to measure with greater accuracy the effects of the various types of user requests and thereby construct accurate models of server system resource utilization as a result of a given workload.

A model of system resource utilization supports capacity planning for future deployments where the workload is known and helps developers assess trade-offs in application development based upon application and system workload data from the field. Developers and system managers can optimize system architecture and design in such environments to achieve optimal performance.

1.1 System Under Study

The Web server system we studied consisted of a single HP 9000 Model 735 workstation with 144 MB of system RAM, two 2 GB disks connected via fast wide SCSI, on a 10BaseT Ethernet network with T3 Internet access. The Web server implementation was NCSA 1.5 [1] with a local modification that allowed us to collect server response time information (see "Server response time" on page 8 for a description). The NCSA implementation uses one server parent process to receive all incoming requests and creates a site-determined number of pre-forked HTTP daemon (`httpd`) processes. The server we studied used 40 persistent children processes.

Figure 1 Web Server Architecture



The architecture of the Web server system is shown in Figure 1. In the figure the Client browser application creates a TCP connection to the server system and makes an HTTP request to the Parent `httpd` process. The Parent `httpd` gives the request to one of the existing (“pre-forked”) Children `httpd` processes for service. (If there are not enough Children processes a new process will be created dynamically to handle the request and will exit when the request is complete.) To satisfy the HTTP request the Child `httpd` process:

1. parses the incoming request to determines what content the client desires
2. retrieves the content (in the case of HTML or Image requests) or computes the result (for dynamic CGI requests).

   CGI (Common Gateway Interface) requests invoke applications to return computed content each time they are selected. They are invoked as the result of the posting of a *form*, clicking on an active *image map* or a hypertext link that refers to a *CGI script*.
3. writes the resulting content to the TCP connection and closes the socket.

The Client browser receives the result of the request and displays it to the user.

The study period consisted of two months during which we received daily logs from the HTTP daemon processes (`httpd` logs request completions to `access_log`) and from the MeasureWare Agent system performance instrumentation. There were no significant failures or major server changes so the data is of very high quality.

1.2 HTTP Daemon Log File Format

To characterize the workload offered to a WWW server system we examined the client request completions logged by the HTTP daemon process (`httpd`) to its `access_log` file. Each WWW request causes a line to be added to this file containing the following information:

- a timestamp indicating the arrival time of the request
- the address of the system making the request: either the client system’s IP address or the IP address of an intermediate firewall or proxy server supporting the client
- the type of request (GET an HTML file, POST a response to a form, etc.)
- the URL target of the request (which may contain parameters)
- the HTTP status after the processing of the request (OK, error, redirect)
- the number of bytes returned, and
- from the OpenMarket server or our enhanced NCSA 1.5 `httpd` the server response time of the HTTP request.

  Server response time is the difference in wall clock time between when the request arrived at the Child process and when the last byte of content was written to the network socket by the Child `httpd`.

Using this information we analyze the offered workload in terms of the request type and interarrival time; the content size on the site; the response size and server response time; the clients requesting service; and so on.

Note that this study focuses only on the server system and in particular on the behavior of the `httpd` processes on the server. This study does not capture the client behavior nor the behavior of the intervening (Internet) network(s) between the Client browser and the `httpd` processes. In particular this study does not examine the effects of client caching, WWW proxy cache or firewall systems, or end-to-end latency of data returned to the client; these are all noted as important aspects of WWW application performance.

Note also that the `httpd` server response time metric is an optimistic view of the end-to-end latency and throughput of each request (see "Server response time" on page 8). Despite this fact, `httpd` server response time provides us with an important attribute of server system behavior, namely the duration of service for a request by the Child `httpd` process. This information is used to construct an analytic model of Web server performance in [2]. That model predicted client response times based upon the server response time.

1.3 Performance Analysis Toolkit

The volume of data in the `httpd` log files is quite large—usually 20-30MB per day (uncompressed) for a busy Web site. We needed a way to analyze this data quickly and efficiently so we developed the Performance Analysis Toolkit [3]. Using the Toolkit, the performance analyst converts the ASCII logs to a record-oriented binary format, and then uses tools that scan the binary records to capture the metrics under study. By performing analysis on the binary data our tools achieve a roughly 50x speedup relative to previous tools that ran on the raw log files.

The analysis process consists of three distinct phases.

1. Data Conversion. Conversion of the ASCII `httpd` logs to the corresponding binary “reduced log format”.

This is done once per log file. A converted log file is around 20% of the size of the raw ASCII log file.

2. Data Reduction. Computation on the binary logs resulting in tabular ASCII output. The reduction tools are separate, modular applications using the library Framework to access data in the binary logs.

3. Visualization. Presentation of the results of the reduction tools. The visualization tools are mostly `perl` scripts that use the `gnuplot` utility to create the graphs presented in this document for our analysis.

Workload Visualization and Analysis

This paper examines requests at a Web server by unique file size, response size, server response time, and throughput using a variety of statistical analyses including the mean, median, and sample distribution of the data. These distributions are presented using a combined plot with a histogram and the Cumulative Distribution Function (CDF) for the data set.

For the histogram and CDF the data were segmented into logarithmic (base 2) sized buckets; e.g., the 128 byte bucket includes all responses between 64 and 127 bytes. The frequency value corresponding to each of the buckets in the histogram represents the proportion of occurrences of a data value in that range. The CDF line represents the cumulative frequency of the current and all previous buckets.

The reason for doing using a logarithmic distribution is that the data we see (such as file size or response time) has a great variance. If using a linear scaled distribution, most of the data would be crowded by the Y axis, with a large number of empty buckets at the high end. With a logarithmic distribution we can more clearly see the detail at the low end, while also capturing the range at the high end.

The point at which a CDF graph crosses the 50% frequency indicates the median value for a distribution. The tallest histogram box corresponds to the mode (most frequently occurring value) of a distribution. We typically plot both the arithmetic mean and the geometric mean on the distribution for reference. The arithmetic mean is the ordinary sum of the values divided by the number of values. The geometric mean is the N^{th} root of the product of the values. The arithmetic mean is coincident with the median in a balanced distribution, such as a normal or uniform distribution. The geometric mean is coincident with the median in a log-normal distribution (the geometric mean can also be computed as the arithmetic mean of the log of the sample values).

1.4 Document Outline

The remainder of this document presents our analysis of the workload on the Web Server under study. The workload study is presented in sections as follows.

- Section 2.0 examines the traffic offered to `httpd`—in daily, weekly, and hourly periods
- Section 3.0 examines the user requests by type and the distribution of file sizes for content on the site
- Section 4.0 examines `httpd` responses by type, size, server response time, and throughput.
- Section 5.0 examines the distribution of user requests in terms of the client network.
- Section 6.0 summarizes the work and presents some directions for future research in the area.
- Appendix A contains supplemental images referenced throughout the text.

The paper concludes with a discussion of the next phase of study, speculation about the likely evolution path for Web workloads, and the impact of those changes on system performance.

2.0 TRAFFIC PATTERN ANALYSIS

This section presents a view of the aggregate traffic at the `httpd` processes on the server system over the analysis period of April and May 1996.

Figure 2 Daily Traffic

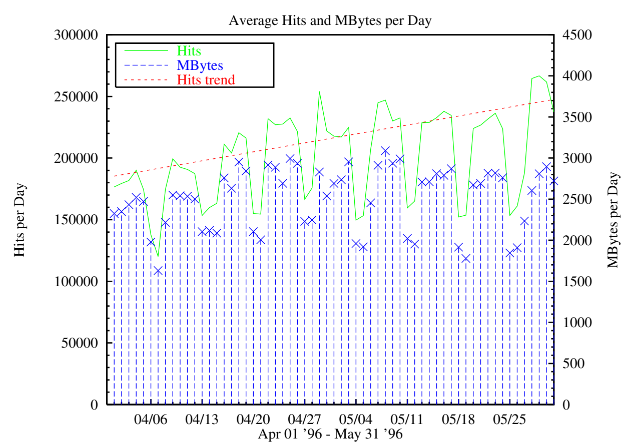


Figure 2 shows the total traffic seen by `httpd` in terms of HTTP client requests (commonly referred to as hits) and megabytes transferred from `httpd` as a result of those requests. In this figure hits are presented using lines connecting the values from day to day using the left-hand Y axis; megabytes are represented by vertical bars read on the right-hand Y axis.

From our analysis we observe the following.

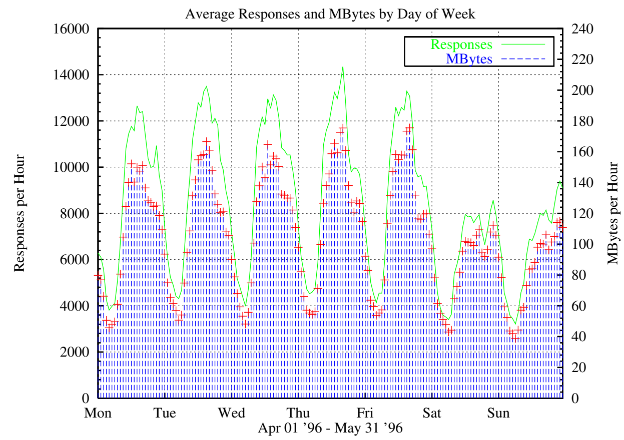
- The `httpd` throughput for the two month period averaged 2.5 GB/day for all transfers or 30KB/sec aggregate continuous traffic. This is the total data volume out of all `httpd` processes for all concurrent connections.
- The system received 12.3 million connection requests over the period averaging over 200,000 hits/day (nearly 8,400 hits/hour). During periods of activity there were on average 24 concurrent connections[1]; the peak was 90 concurrent connections at one point.
- The average throughput per HTTP connection was 1.5KB/sec. Connection throughput is calculated by dividing the transfer size of a request by server response time.
- The traffic at `httpd` is increasing both in terms of hits per day and megabytes transferred per day.
- A linear regression of the traffic on weekdays for these two months showed the traffic to be increasing at roughly 7,000 hits per week and 37MB per week. Some caveats to be aware of with this estimation:
    - ❏ Growth may not be linear: Internet usage is growing at an exponential rate [4].
    - ❏ System capacity will eventually limit growth, reducing the apparent growth rate.
    - ❏ Advertising of the site (explicit and implicit, such as by advertising a product discussed on the site) can have a significant effect upon traffic.
    - ❏ The correlation coefficient for the weekday regression fit was r^2=0.58 indicating there is still a substantial amount of variance not covered by our estimate (when considering all days the fit was very poor: r^2=0.19).

Weekly Traffic

The traffic in Figure 2 shows clear periodicity with the day of the week. A further analysis shows the expected periodicity with hour of day as well. Figure 3 plots the aggregate HTTP traffic by hour for each day of the week in the period studied.

1. A *concurrent connection* is a request that starts prior to the end time of one or more earlier requests. End time is defined as start time plus server response time.

Figure 3 Weekly Traffic



From Figure 3 we observe:

- Monday through Friday have a similar traffic pattern; Saturday and Sunday are slower days and show different patterns from each other and from the other days.
- The slow time in the above chart is between 04:00 and 07:00 each day. The busy time is 16:00 on weekdays; 22:00 on weekends.
- The number of hits at 22:00 on Sunday is greater than the number of hits at 22:00 on any other day of the week.

Hourly Traffic

The weekly traffic is also periodic within a day (as evident from Figure 3). The peak utilization is during the work week so we examined traffic for the weekdays (Monday-Friday) by hour.

Figure 4 Weekday Hourly Traffic

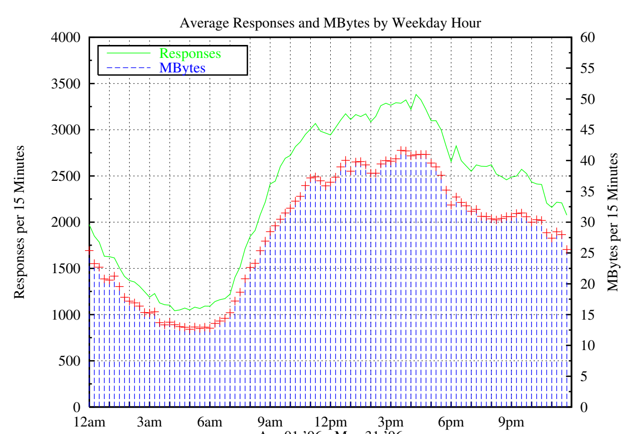


Figure 4 shows that the byte traffic tracks closely with the request rate during weekdays and indicates the peak traffic hours are 15:00-16:00 (Eastern time). Further analysis of system resource utilization and bottleneck analysis will focus on this peak time.

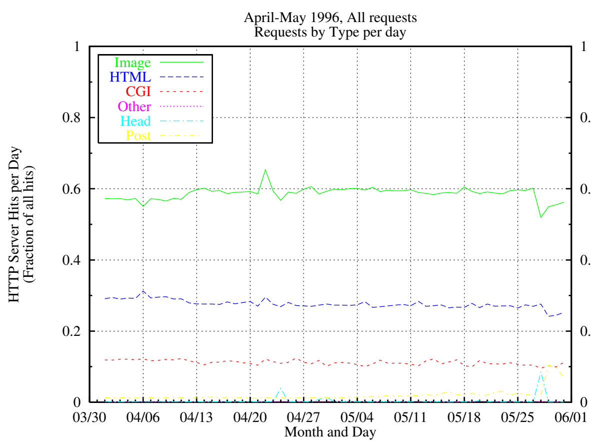
3.0 User Request Characterization

Aggregate Requests by Type

This section characterizes the user requests by looking at the type of requests made by users, the request interarrival time, and the size distribution of files requested from the site.

Figure 5 presents the traffic by HTTP request type. The request type is determined by looking at the URL (path and extension) requested by the client. At this site the dominant type per request is images; this is consistent with what we observe at other sites with professionally developed content. Looking at the numbers in terms of bytes transferred, images are by far the dominant type with nearly 90% of the byte traffic (see Figure 12 in Appendix A). This is typical given the size distribution of images (see Figure 15). Note that there is no video content on the site and audio content represents only a small portion of the requests.

Figure 5 Aggregate Requests by Type



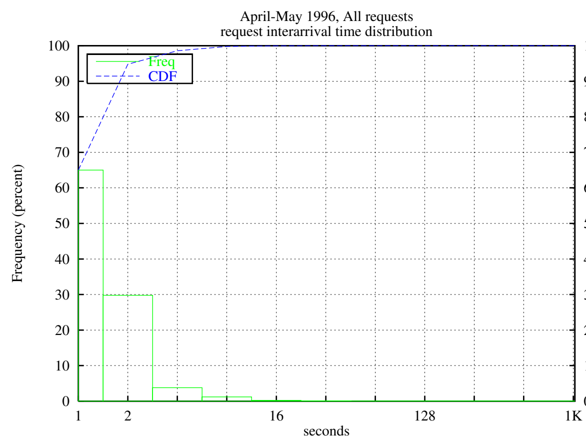
Request Interarrival Time

The interarrival time for hits at the server indicates the rate of user requests as well as the think time for both browsers and human users. A user request is a collection of hits starting with an HTML request and including that HTML and any subsequent inlined Image requests. Browser think time is the interval between retrieval of an HTML file and the subsequent request for inlined images within the same user request. User think time is the time between user requests by the same user.

Unfortunately at this site a majority of the requests come through firewalls and cache servers. These intervening servers coalesce many users into one client IP address (which is how users are identified), so the content server sees only one apparent user. Therefore our ability to detect individual user requests is greatly diminished, and we cannot perform an analysis of user think time at this site. Our measurement of user requests provides only a lower bound on the actual number of user requests at the site.

Figure 6 shows the distribution of the interval in seconds between subsequent requests for any content on the site. This `httpd` implementation logs requests with only second granularity so a finer breakdown of the interarrival times within one second is not available.

Figure 6 Request Interarrival Time



This distribution is consistent with the high concurrent usage rate on the server (on average 24 concurrent requests): the interval of time between requests should be very small with so many concurrent requests (the mean request interarrival time at this site was 0.4 sec). When we examine the interarrival time just for HTML documents the peak moves to the [1-2) second bucket but still drops off rapidly as above.

Looking at the interarrival times strictly for the site's home page we see that the median arrival time between requests for the site's home page is 8-15 seconds. (See Figure 13 in Appendix A). We interpret this to mean that a new user is selecting the site that frequently, either by clicking a link that leads to the site or selecting the site from their bookmarks or favorites. Existing users navigating within the site should not generate a hit on the home page since it will likely be in their browser's cache.

File Size Distribution

First we examine the distribution of the unique files on the server system (i.e., ignoring multiple requests for the same file).

Since this analysis used only the `httpd` logs we cannot analyze the distribution of files that were never requested, only those that were requested one or more times. We only considered those transfers which had non-zero size and were successfully retrieved (i.e., had a HTTP server response code of "200 OK"). Aborted transfers were ignored (they are detected as "short transfers" as described below).

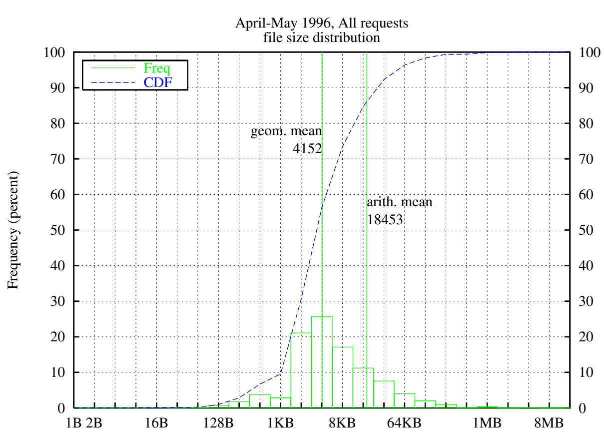
From Figure 7 the most common file size (the mode) is between 2-4KB with 25% of the available files in that (2KB wide) range. The CDF curve crosses the 50% mark at 4KB as well, which indicates the median and the mode are coincident. Further, 95% of all files are less than 64KB in size.

The average (arithmetic mean) file size is 18KB; over 80% of the files are smaller than the mean file size. This indicates that relatively few very large files are responsible for much of the upper tail of this distribution, and supports our use of a logarithmic scale for examining file sizes. By contrast, the geometric mean of this distribution is 4KB, which is coincident with the median.

We also looked at the rate of change of content files and observed that the retrieval size for certain URLs changed very frequently. The types of content changing in size fall into the following categories.

- CGI (dynamic) content. The most frequently changing content are CGI scripts, which return variable sized content depending upon the user query. Size changes are expected to be frequent here (on the order of number of retrievals per URL).

Figure 7 File Size Distribution



- Dynamic (.shtml) pages. Many of the pages on the site return a dynamic date and time stamp; the formatting of the date has a "feature" in which a leading 0 for the minute or second value is omitted thereby changing the size of the page by one or two characters.
- Large images. Size "changes" for these files often correspond to short transfers—where the user clicks a link on the page containing the image or instructs the browser to quit loading the image (possibly by navigating through their history stack or bookmarks, clicking with an image map that is partially downloaded, or clicking the stop button), causing `httpd` to receive a "broken pipe" indication on the socket. (The server still records it with the status "200 OK".)

The short transfers are of interest as they can indicate both user frustration (in the case where transfers are too slow) and unnecessary system work (transferring data that is ultimately discarded). The six files with the most short transfers for April and May were GIF images; most of them were image maps, including the site's home page. Over a quarter of all transfers for these images were terminated before they were complete. Their size ranged from 37KB to 67KB.

Performance Hint

To reduce the number of short transfers, frequently retrieved images should be kept to a minimum size if possible, by reducing the image size, its color depth, or by encoding it more efficiently (e.g., using the JPEG instead of GIF format).

HTML and Image File Size Distribution

For HTML pages, the mode of the distribution is 2-4KB; HTML files in this range account for most of the files in the overall distribution for this range. The distribution of HTML files tails off very quickly over 8KB. (See Figure 14).

The response size distribution illustrates the relative popularity for files in each bucket. See Figure 16 on page 15 for that discussion.

For images the file size distribution has a much heavier tail than HTML files. The mode of this distribution is 16-32KB, corresponding to moderate size and resolution GIF or JPEG images. There are also quite a few images in the range 1-2KB, corresponding to thumbnail or other small or low resolution images. (See Figure 15.)

Content Popularity

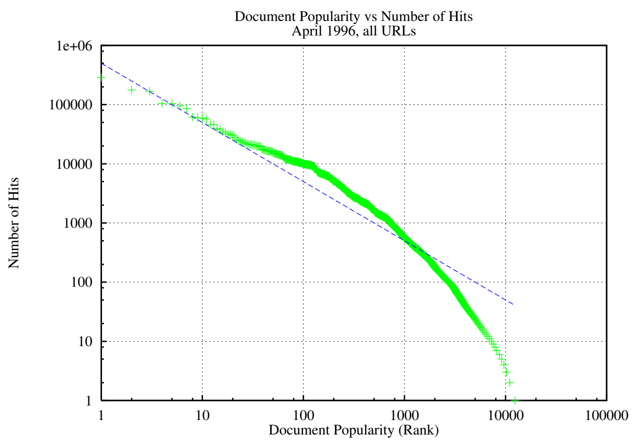
Previous studies of Web workloads [8][9] have shown a high degree of locality of reference among request streams. The popularity for the content at a site tends to follow a power law distribution; some studies suggest that content popularity specifically follows a *Zipf* distribution. In general a power law distribution is one of the form:

$$Y = x^a \qquad \text{(EQ 1)}$$

For some constant a. In the case of the Zipf distribution the constant a is exactly *-1*. Put another way, Zipf's law predicts that the number of hits of a document (H) is related to the document's popularity rank (r) via the formula $H = 1/r$.

When we examine the popularity rank versus request count at this site we see a similar pattern. We visualize this distribution using a log-log plot in order to be able to see detail at the low end of both the rank and hits axes; we also use the log transformed data for curve fitting (linear regression).

Figure 8 Document Popularity vs. Number of Requests



For the first 1,000 URLs in popularity the slope of the best fit curve (on the log-log plot) is -1.05 (with a correlation coefficient of 0.986). We include a line of slope -1.00 for comparison. However, for the first 100 URLs the slope is only -0.71. The slope for the next two thousand URLs is 1.77 (with a correlation of 0.999), and beyond that the slope was consistently below -2 with a correlation coefficient of 0.99 or better. This distribution confirms that popular content on this site is very popular; in addition we observe that unpopular content rapidly becomes increasingly unpopular. (Note that there were over 10,000 distinct URLs available in April.)

These results were confirmed when looking at individual days (to help reduce the effect of transient content).

4.0 SERVER RESPONSE

This section characterizes the response of the server to the user request stream by HTTP response code (success, redirect, failure), server response time, and response throughput (the response size divided by the server response time).

On this site 88.9% of the requests are satisfied directly and without error (the "200 OK" response code). Another 10.7% of requests are "successful" in that they returned useful information to the client browser. Less than half a percent of the total responses were errors at the site. The other "successful" request types are:

- 302 Found: The "Found" response type indicates that a document has a different temporary location. This status code is usually returned as a result of clicking on an image map. The response contains a URL that the browser should access instead—which typically the client browser does immediately (causing a subsequent hit on a different URL).
- 304 Not Modified: Some browsers and proxy cache servers use HTTP CONDITIONAL GET requests to check whether an HTML file or image in their cache is up-to-date. The CONDITIONAL GET request carries the last known modification time for that content; the server determines if the content has been modified since then. If the server has newer content for that URL the newer content is returned to the client (typically resulting in a "200 OK" status code). If the document has not been modified the server responds only with the "304 Not Modified" status code saving a document transfer.
- 301 Moved: This response type indicates the document has moved permanently to a different location. This is

Table 1 Server Responses by Code

Response code	Hits	% of Total	KBytes	% of Total
200 OK	21863046	88.9%	3130003	99.9%
302 Found	1695286	6.89%	151406	0.05%
304 Not Modified	946528	3.85%	0.00	0.00%
404 Not found	74828	0.30%	30214	0.01%
500 Internal Error	5672	0.02%	2888	0.00%
301 Moved	4784	0.02%	0.00	0.00%
401 Unauthorized	1418	0.01%	0.00	0.00%
000 Unknown	104	0.00%	28	0.00%

returned in most cases when a client browser requests a URL that corresponds to a directory without including the trailing "/" character. The response also includes the URL the client browser should contact for the document (with the trailing "/")—the browser will immediately request that URL.

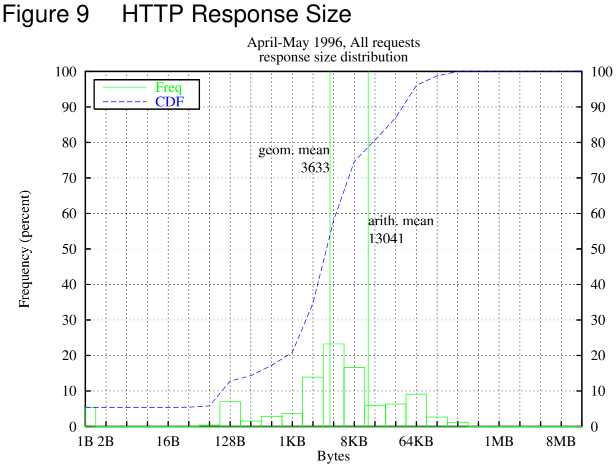
Performance Hint

Links to directories on a site should either end with "/" or fully specify the HTML file ".../index.html" to prevent the additional round-trip from the client.

Response Size

The `httpd` response size analysis is similar to the file size distribution analysis except it examines each transfer, not just each unique file. As mentioned earlier this response size value is the size of the HTTP data and does not take into account protocol overhead. Overhead varies with connection type, HTTP content, and network condition (which determines retransmission requests and TCP window size).

Figure 9 HTTP Response Size



At this site the arithmetic mean of the response size was 13KB; the median response size was 2-4KB. The mode coincided with the median but the distribution of response sizes has a minor mode at 128B and another at 64KB. These minor modes correspond to the modes of the distribution of CGI and Image requests respectively. These can be seen clearly from the HTML, Image, and CGI response size graphs (see Appendix A). The 64KB bucket includes transfers of the site's home page which is a 49KB image map.

From this CDF 50% of all responses are for 4KB of data (the median) or less and 95% of all requests are for files of 64KB or less. The mean in this distribution is much larger than the median and mode indicating that some very large transfers are responsible for a significant amount of the traffic.

Comparing this distribution with the distribution of file sizes it is apparent that the CGI content is responsible for a moderate number of accesses (the CGI file size distribution cannot be calculated and therefore is not presented). The file size distribution also falls off more consistently from its mode than the response size distribution, indicating that large files (especially images in the 32-64KB range) are relatively more popular on a hits-per-file basis. This could be caused by client caching effects (such as large image files flushing other files out of the browser's cache) or user preference.

Figure 16 presents the size distribution for HTML files only. The figure shows that the 2-4KB HTML files are very popular, accounting for nearly 50% of all HTML transfers. In the HTML file size distribution (Figure 14) there were a number of HTML files in the 1-2KB range which are proportionately less popular than 2-4KB HTML files; and 256-512B files (5% of all HTML files) are rarely requested (less than 1%).

For images (Figure 17) the most common transfer size on our logarithmic scale is for images in the range from 32-64KB (a 32KB wide range) with 16% of the transfers. However the scale is deceptive—the number of transfers of images in the 7KB range [1KB-8KB) (represented on the chart by the 2KB, 4KB, and 8KB histogram buckets) is nearly three times greater accounting for about 45% of all image transfers. Furthermore 77% of the transfers were smaller than 32KB.

Comparing this figure to the file size distribution of images (Figure 15), there is a preference for requests of the larger (32-64KB) images on the site as compared with the 8-16KB and 16-32KB images, of which there are actually many more files on the site. The site's home page explains some of this effect: it is 49KB and accounts for 3% of all transfers.

The CGI content returned (Figure 18) shows a pronounced spike at 64-127 bytes corresponding to a few lines of text or URLs. There is another peak at 2-4KB corresponding to larger content such as documentation or image thumbnails.

Server response time

The server response time metric at this site was collected using a special instrumented version of the NCSA 1.5 `httpd`. This daemon notes when a request arrives at an `httpd` process and computes the total elapsed time in milliseconds until the last byte has been written to the network protocol stack by `httpd`. This measurement does not capture client or network delay—in particular it does not take into account the time for:

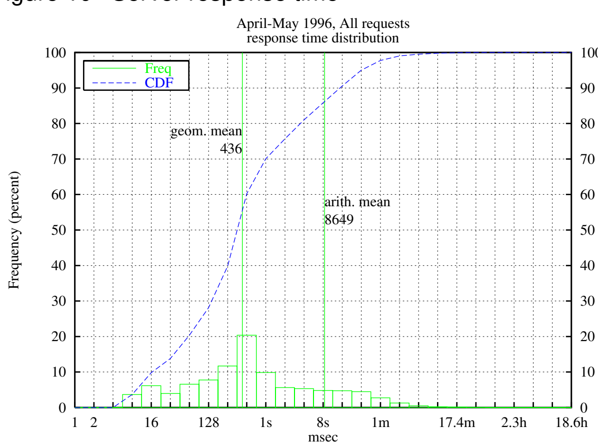
- client connection setup (one round trip to the server system) prior to the request arriving at the Web server

- request queueing and protocol processing delay on the server system prior to arriving at the `httpd` process
- server transmission of the last window of TCP data from server system's network buffers to the client system
- TCP connection teardown on the server system
- Any processing time or I/O latency on the client system.

The last TCP window of data is typically not more than 8KB of data and can be even less (the TCP window cannot open to its full size if there is little data to transmit). The server response time metric *does* include the round trip time of all network transfers and the corresponding client acknowledgments except for the last window of data. Client acknowledgment of each data packet is required by TCP before the window size is increased or subsequent data is sent by the server.

We have studied the correlation between `httpd` server response time and client response time in LAN and FDDI environments and have found the discrepancy of reporting to vary considerably with the response size. The response time is within 10% of the server response time for transfers over 128KB; but for smaller transfers the metrics can differ by more than a factor of 10. We have not evaluated the discrepancy across the global Internet. Such a study must take into account the variety of clients and service providers; the discrepancy here would also be substantial.

Figure 10 Server response time



Note that while the server response time metric does not give an accurate measure of client response time for most (98% of) transfers, it is still a valuable measurement in the sense that it reflects the time `httpd` took to satisfy each request. This is potentially important for server resource utilization and system capacity planning.

The server response time distribution shows that the most common server response time was 256-511 msec. The median server response time was under 512 msec (60% of clients were served within 512 msec of `httpd` residence).

However this distribution has a very heavy tail. The `httpd` arithmetic mean server response time was nearly nine (9) seconds and 10% of all requests took 15 seconds or longer to satisfy. The distribution of image and CGI requests accounted for much of the heavy tail as can be seen from their graphs in the following sections; in particular the contribution of images accounts for 85% of this tail of requests taking 15 seconds or longer to satisfy.

For HTML files (see Figure 19) the server response time has a strong peak and falls off consistently on the upper and lower ends paralleling the HTML response size distribution. Interestingly, the mode and median of the HTML response time distribution are 256-511 msec, and there are very few (less than 4%) responses less than 128 msec. However, we see that 28% of all responses (and nearly 40% of Images) are served in under 128 msec. This anomaly is caused by the use of *parsed HTML*, in which the `httpd` examines the HTML content and expands variables in the content (e.g., the LAST_MODIFIED tag is replaced by the modification date of the HTML file). However, since the content sizes are relatively small (compared to Images) the network latency is limited; and since they are consistent in size (see "HTML File Size Distribution" on page 15), the processing time is also limited on the upper end.

Performance Hint

In order to reduce CPU consumption on this site the use of parsed HTML should be limited. Nearly all of the HTML content on this site was parsed by the `httpd`, requiring additional CPU time. The main use of parsed HTML on this site was to expand the content modification time. This technique can be replaced by the use of static information in the content to reduce CPU utilization.

Images present an entirely different distribution from HTML content (see Figure 20). The CDF for this distribution is nearly a straight line on a logarithmic plot, indicating an exceedingly heavy tail. Further analysis reveals:

- The median server response time for image requests is still under one half second, but the arithmetic mean is over 13 seconds! A few very long requests are skewing the mean significantly from the distribution's median value.
- Only 18% of all requests take longer than the mean server response time; 5% of all requests have longer than a minute server response time.

- 3 requests took over an hour of server response time to satisfy. These requests have a significant effect on the mean, making it less useful as a predictor of system behavior.

The heavy-tailed distribution is likely caused by large image files going over slow network connections. The tendency for a significant number of requests to take more than a second combined with the relatively high request rate at this site indicates a need for processing requests in parallel. (The average concurrency level of 24 requests indicates that response parallelism is being exploited at this site.)

The CGI response time distribution (for dynamic content) also shows a heavy tail, although not as prominent as for Images. The response time is quite long relative to the response size distribution. This is because each CGI request requires CPU processing (typically the running of a `perl` script) in addition to disk data transfer and network protocol overhead. The CPU processing time increases the server response time prior to the data being returned. The CPU time can be significant depending upon the nature of the CGI application invoked.

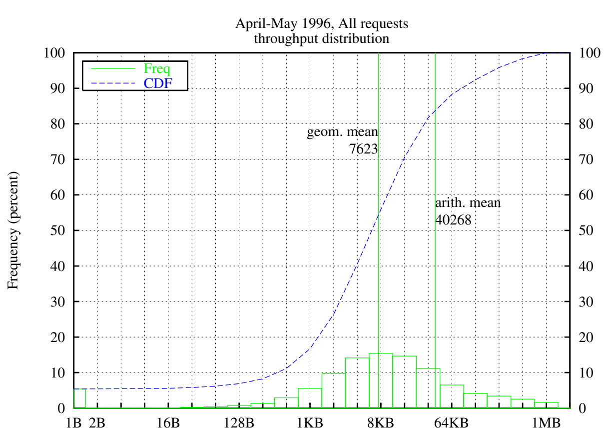
Throughput (Response Size by Server Response Time)

Given the response size and server response time we compute the `httpd` throughput on a per request basis.

At this site the server was on a 10BaseT Ethernet segment connected to the Internet by a T3 link. The actual maximum network bandwidth of the Ethernet segments and routers between the server and the site's Internet Service Provider was not measured, but is assumed to be on the order of 5-8 Mb/sec.

The server throughput histogram and CDF show that the median throughput is nearly 64Kb/sec (8KB/sec) and there is a strong concentration of the rates around this value. 90% of the requests result in 8Kb/sec or better transfer from the server. Recall that server response time is a poor predictor of client response time, so this throughput measure only describes response throughput at the server.

Figure 11 Server Throughput



We further examine the throughput for the HTML. Image, and CGI types. We can see that the HTML throughput (see Figure 22) is more consistent than for the other types. This is expected given the response time and response size distributions.

For the Image type (Figure 23) the throughput distribution is significantly broader than the HTML distribution—having faster as well as slower transfer rates. The faster rates are due to the small image content being transferred without parsing. The slower rates of course are due to large image content being sent across the Internet, possibly encountering network congestion or slow connections. Interestingly, the median throughput is the same for both HTML and image content: 64Kb/sec.

The CGI throughput (Figure 24) is influenced by the added latency due to extra CPU processing required to process these requests.

5.0 Client Domain Analysis

In this section we examine the domain origin of the user requests. In the `httpd` logs only the IP address is recorded (conversion to a DNS domain name is disabled on this site). To compute the client network name we stripped off the host portion of the requesting client's IP address yielding just the network number and converted this to a network name using the InterNIC `networks.txt` database.

The 12.3 million requests to the site during the analysis period, were from 368,000 distinct client IP addresses on 48,571 distinct IP networks. Of those:

- 17 were class A networks, 4,395 were class B networks, and the rest were class C.
- The distribution of requests by client network followed a power law distribution ($y=x^a$) in terms of both hits and bytes transferred. The correlation coefficient for this fit was r^2=0.98 for hits (r^2=0.61 for bytes).
- The greatest number of requests came from AOL-BNET with 2.81% of the requests (1.27% of the bytes, also the most).
- The UUNETCUSTB36 + UUNETCUSTB37 networks together accounted for 1.80% of hits and 1.70% of bytes (more than AOL-BNET).

- User access from Internet Service Providers (ISPs including AOL, UUnet, PSI, Prodigy, CompuServe), Universities, industries, and governments seem to be the main sources of requests to the Web Server.
  - The most frequent University users are at UCB (University of California Berkeley) and RIT (Rochester Institute of Technology)
  - The most frequent industrial users are Xerox, HP, and IBM, in that order.
- About 22% of accesses were from networks whose network number was not registered in the InterNIC `networks.txt` file as of May 30, 1996. This illustrates the difficulty of keeping up with network changes.

Table 2 Top 25 Client Networks

Client Network	Hits	%	MB	%
aol-bnet	345409	2.81	1988362	1.27
uunetcustb36	155062	1.26	1834390	1.17
psineta	69153	0.56	922992	0.59
websterxerox	67946	0.55	299624	0.19
ucb-ether	67197	0.55	206383	0.13
uunetcustb37	66320	0.54	819372	0.52
194.25.2.0	62592	0.51	617513	0.39
hp-internet	39272	0.32	620935	0.40
swipnet	39262	0.32	563956	0.36
204.148.103.0	32104	0.26	255688	0.16
198.83.19.0	31111	0.25	294468	0.19
mindspring	27800	0.23	333586	0.21
ans-bnet14	27763	0.23	423865	0.27
demon	27113	0.22	301735	0.19
hinet-b	25338	0.21	236085	0.15
jaring-nat	24608	0.20	299322	0.19
ans-bnet15	24401	0.20	326063	0.21
rit	22809	0.19	276080	0.18
ibm-hpcc14	21767	0.18	341732	0.22
203.241.132.0	20960	0.17	238972	0.15
198.83.18.0	19881	0.16	158079	0.10
commscicntr2	19061	0.16	377036	0.24
motorola	18253	0.15	261326	0.17
idt2	18139	0.15	225869	0.14
uiuc-campus-b	17621	0.14	229962	0.15

6.0 CONCLUSIONS AND FUTURE WORK

The Web Server under study is a busy Internet site receiving a quarter million hits per day and experiencing increasing traffic on a week-to-week basis. The traffic tends to be similar to that at other Web sites based upon our experience and a review of the literature. In particular, we can confirm several of the invariants proposed by Arlitt in [9]. We see a similar traffic growth to Kwan, McGrath and Reed [10], and confirm their weekly and hourly request patterns.

We also observe that the document popularity profile as noted by Almeida, Bestavros, Crovella, and de Oliveira [8]. We have not to date explored their other findings regarding content locality of reference, and note this as an area for future research.

During the workload characterization phase we have not examined system resource utilization in depth; but based upon the data we have it does not appear the server system was under extreme pressure for the duration of this study. We conclude this by examining the daily and weekly traffic graphs: systems under severe request pressure tend to have a pronounced “flat top”, a period during which the maximum number of hits or bytes per time period remains consistent. A “flat top” can indicate either that the request or response traffic exceeds the available network bandwidth or that the mean arrival rate at the server is greater than the mean service rate. In the latter case, a server system or network bottleneck is limiting the service rate. This simple analysis does not give any insights on how to improve the performance for the end user, nor does it indicate the point at which a resource bottleneck will limit system performance in the future.

6.1 System Performance Recommendations

During this study image content dominated the site’s traffic in terms of bytes transferred with HTML coming in second. This is consistent with traffic at other Web servers with many images (commercial sites with professional content tend to rely heavily on images). The primary task of the `httpd` when returning image and HTML content is the transfer of file system data from disk (or cache memory) to the network. In order to maximize system performance the key factors are cache size and overall I/O bandwidth.

- WWW content exhibits a great degree of server-side temporal locality of reference (see [5][8][9]), so LFU caching allows hot content to be served from memory instead of disk. Client side and proxy caching are also important to reduce server load.
- I/O bandwidth from disk to memory and bus bandwidth between memory and network are a premium resource in large, busy web servers.
- Network bandwidth between the server and the nearest Internet NAP is a key performance factor (this may be the limiting resource at the site we studied). The local network must have sufficient bandwidth for the number of concurrent connections.
- At busy Internet sites there will be a large number of concurrent requests. To serve these requests there should be a sufficient number of `httpd` processes; this may require tuning the maximum number of user processes. Operating system network buffer space and the maximum allowed number of open TCP connections must also be sufficient for the expected workload.

6.2 WWW Workload Evolution

Over the next few years we expect to see a change in Web workload as more content becomes available, as more of it is professionally produced and managed, more clients (end users) come on-line, network bandwidth increases, and available content expands to include new media types.

Currently Internet bandwidth is a limited resource, particularly at the network edges (and especially into homes with analog phone line modems). With increased deployment of Broadband Internet Delivery Systems (HP BIDS or its competitors) into consumer premises using high speed access networks like cable modems or high speed telephony (ADSL, etc.), consumer connection speeds will approach that of the enterprise desktop. This deployment is projected to take many years due to the high cost of installing infrastructure capable of providing this access to consumer premises. But as network bandwidth increases audio and video content will become possible to deploy in real-time and with adequate quality. Availability of these services will likely cause a significant change in the user request pattern and given the media size, will quickly dominate today's image and HTML content in terms of total bytes transferred.

However, it is not likely that significant amounts of audio and video content will be served by today's `httpd` with the current download-and-play model. For these content sizes, streaming is a more natural access method:

- Downloading content first takes a significant time during which the user must wait; with a streaming service the user begins playing the content as soon as a local buffer pool is sufficiently primed (to accommodate jitter, or network delay variance).
- Downloading content requires a large amount of disk cache on the client system. Using a browser's native download-and-play model this large content will flush many smaller pieces of content from the cache, lowering cache utilization. By contrast streaming applications are designed to buffer only enough content to provide the required quality of service for the stream, and thus do not involve the browser cache.
- However, streaming media will not be possible until there is enough available network bandwidth to accommodate the stream with acceptable delay jitter. If the network pipes are either too small, or subject to congestion and high delay variance, downloading the content is the only viable solution.

In addition to isochronous audio and video content, higher network bandwidth will allow the use of more and larger high resolution images. With the coming availability of high quality consumer print devices (the next generation of DeskJet technology), consumers will increasingly be retrieving and printing large image content. This content will not cause a qualitative change in the workload, but will cause more image content to be transferred per connection. Depending upon a site's file size distribution and response size distribution, alternative local and remote caching mechanisms may be in order to reduce either user requests (by caching many small files) or bytes transferred (by caching more large popular files). This trade-off is discussed to some extent in [9].

6.3 Future Work

The next phase of our study will focus on system resource utilization as a function of offered workload. The study of resource utilization will be based upon `httpd` logs and daily MeasureWare Agent (PCS) logs we collected during the study period. The primary contributions of the resource utilization phase will be to understand current system behavior, perform a primary bottleneck analysis to determine the resource most responsible for limiting current performance, and to create a model describing system resource consumption as a function of offered workload and a related capacity planning model to allow system sizing based upon expected workload. Using the resource model we will be able to estimate the point at which the current system will be saturated and the most effective solution to that condition.

At the time of this publication a study of the data from this site using the Layered Queueing Network Model analysis technique has been performed and the results prepared for publication in [2].

In addition to the exploration of Web server performance, other important areas of research include proxy and cache server performance. A proxy server provides clients the ability to cross security domains (firewalls) to retrieve content; a cache server provides a single content cache for many co-located client systems. Content caches are often located at proxy servers, since requests must travel through the proxy. This combination is referred to as a "proxy cache". Some research topics in this area include cache server utilization, bandwidth reduction on network links, and latency reduction for clients. All these are affected by cache server topology and must be studied in relation to various topology models.

Another area of study is exploration of the traffic for self-similarity. Several papers have characterized traffic at Web servers and observed the presence of long-range dependence in the traffic [6][8]. These studies have not analyzed traffic over long time frames as we have; it would be interesting to determine if a two month traffic trace shows the same levels of long-range dependence and locality of reference among requests as seen in these other research papers.

7.0 ACKNOWLEDGMENTS

This study is the result of a team effort. In particular I wish to thank our customer for making their data available and reliable; Rich Friedrich for his guidance and experience in the analysis of the data; Jim Salehi for his thorough review comments and suggestions; Gary Herman and Gita Gopal for their encouragement, suggestions, and review comments; Tai Jin for the instrumented `httpd` and his expert advice on `gnuplot`, `perl` and other tools that made this analysis possible; and Jerome Rolia for collaboration and insight on Web server performance modeling.

HP's study of this WWW server and our access to the data are covered under a Non-Disclosure Agreement. The identity of the customer has been masked to allow distribution of this report. HP employees interested in the identity of the customer should contact the author.

8.0 REFERENCES

[1] A Scalable HTTP Server: The NCSA Prototype. E. Katz, M. Butler, R. McGrath. Computer Networks and ISDN Systems, Volume 27, November, 1994.

[2] Measurement Tools and Modeling Techniques for Evaluating Web Server Performance. J. Dilley, R. Friedrich, T. Jin, J. Rolia. HPL-TR-96-161, December 1996. Submitted to Performance Tools '97.

[3] Web Server Performance Analysis Tools web page. `http://nexus.hpl.hp.com/bspt/tools/` (accessible only to browsers within HP).

[4] Internet Domain Name Survey by Network Wizards. Available at `http://www.nw.com/`.

[5] Characteristics of WWW Client-based Traces C. Cunha, A. Bestavros, M. Crovella. BU-TR-95-010.

[6] Explaining World Wide Web Traffic Self-Similarity. M. Crovella, A. Bestavros. BU-TR-95-015.

[7] Application-level Document Caching in the Internet. A. Bestavros, R. Carter, M. Crovella. Published in Proceedings of SDNE'95: The second International Workshop on Services in Distributed and Network Environments. Whistler, Canada, June 1995.

[8] Characterizing Reference Locality in the WWW. V. Almeida, A. Bestavros, M. Crovella, A. de Oliveira. To be published in Proceedings of Parallel and Distributed Information Systems (PDIS) '96.

[9] Web Server Workload Characterization: The Search for Invariants. Master's thesis, Martin Arlitt, University of Saskatchewan.

[10] User Access Patterns to NCSA's World Wide Web Server. T. Kwan, R. McGrath, D. Reed. `http://www-pablo.cs.uiuc.edu/Papers/WWW.ps.Z`

Other Web references:

- OCEANS Research Group at Boston University, `http://cs-www.bu.edu/groups/oceans/Home.html`
- DISCUS group at the University of Saskatchewan, `http://www.cs.usask.ca/projects/discus/discus_reports.html`
- Pablo and Mosaic research group at UIUC, `http://bugle.cs.uiuc.edu/`

APPENDIX A

This appendix contains 13 supplemental images referenced in the text.

Figure 12 Aggregate Response Bytes by Type

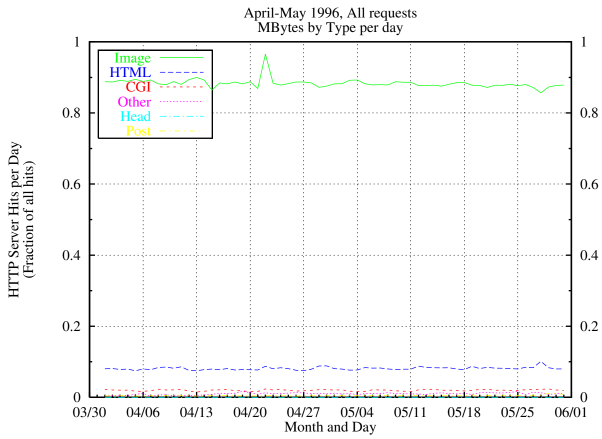


Figure 13 Home Page Interarrival Time

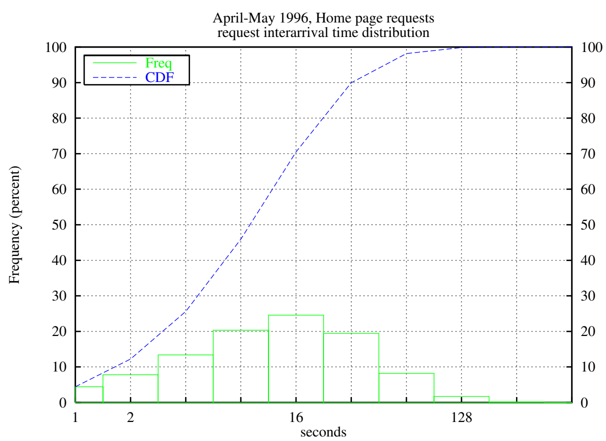


Figure 14 HTML File Size Distribution

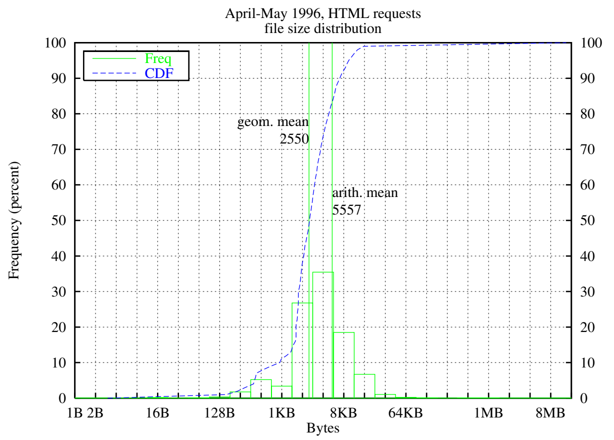


Figure 16 HTML Response Size Distribution

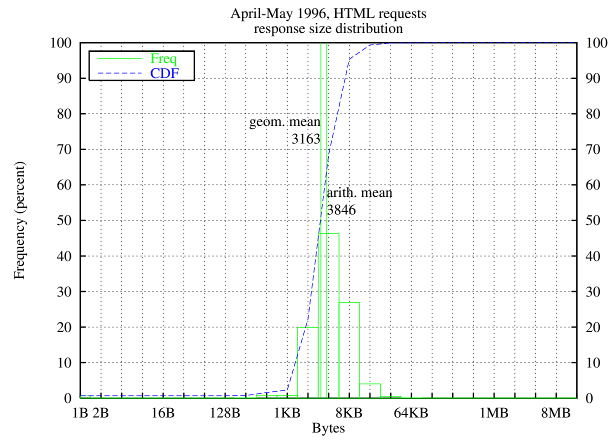


Figure 15 Image File Size Distribution

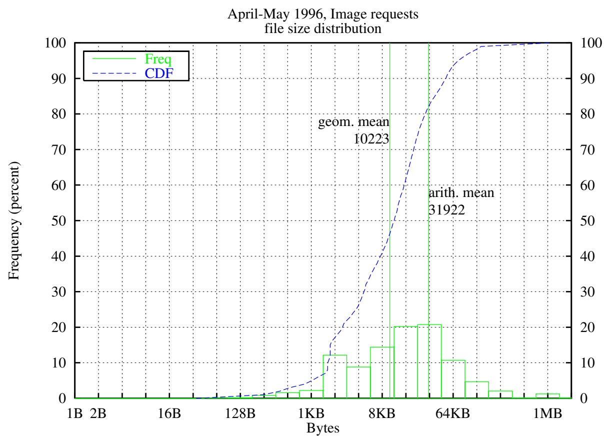


Figure 17 Image Response Size Distribution

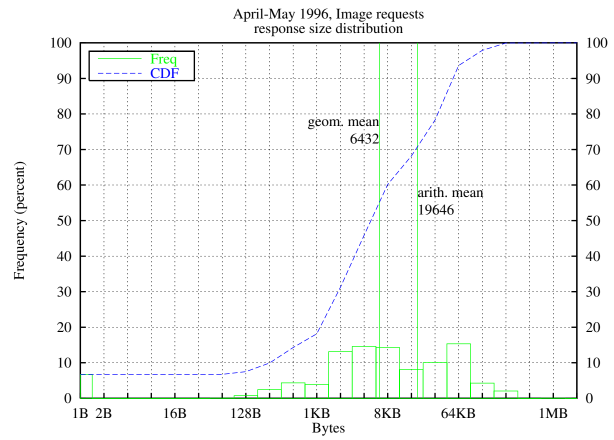


Figure 18 CGI Response Size Distribution

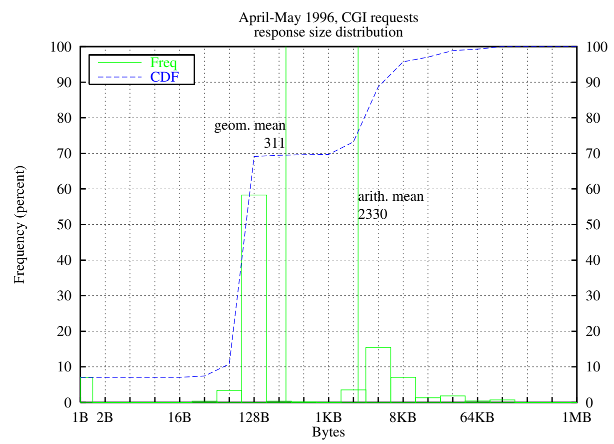


Figure 19 HTML Server Response Time

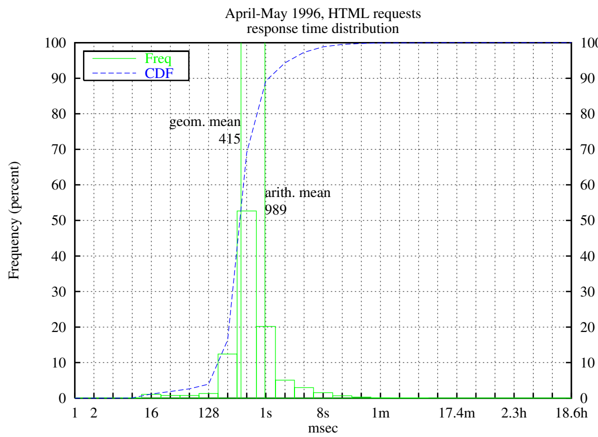


Figure 20 Image Server Response Time

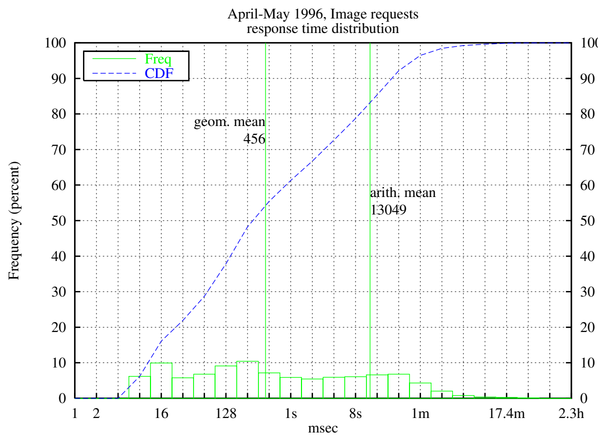


Figure 21 CGI Server Response Time

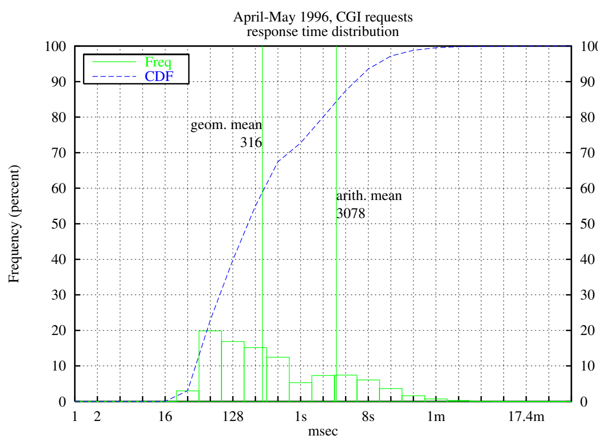


Figure 22 HTML Response Throughput

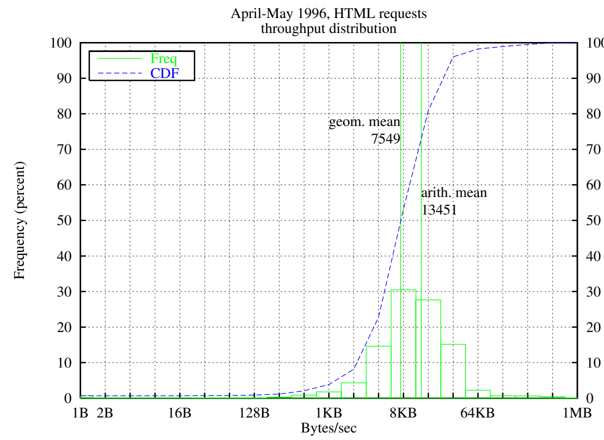


Figure 23 Image Throughput

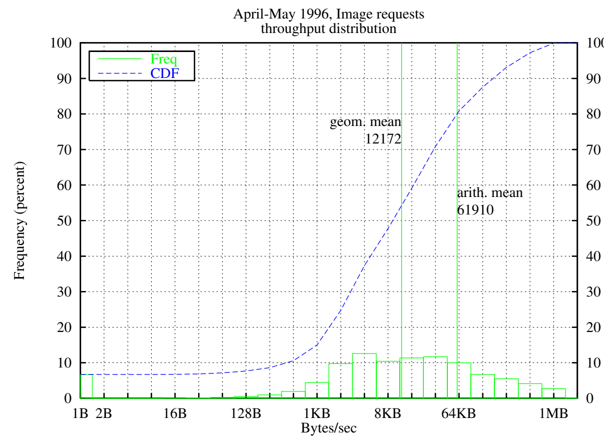


Figure 24 CGI Throughput