Deconstructing SPECweb99

Erich M. Nahum
IBM T.J. Watson Research Center
Yorktown Heights, NY, 10598
nahum@watson.ibm.com

Abstract

SPECweb99 has become the de-facto standard workload generator used to evaluate Web server performance in the computer industry. In this paper, we examine how well SPECweb99 captures the server performance characteristics that have been identified by the research community, such as URL popularity and the distributions of request methods, response codes, and transfer sizes. We compare these characteristics generated by SPECweb99 both with those reported in the literature and against a set of sample Web server logs, and find that the workload generator has a varying record depending on the characteristic. In particular, SPECweb99 suffers from failing to capture conditional GET requests, which significantly affects both the response code statistics and the transfer size distributions. We conclude with some recommendations to improve the accuracy of the benchmark in the future.

1 Introduction

A central component of the response time seen by Web users is the performance of the origin server that provides the content. There is great interest, therefore, in quantifying the performance of Web servers: How quickly can they respond to requests? How well do they scale with load? Are they capable of operating under overload, i.e., can they maintain some level of service even when the requested load far outstrips the capacity of the server?

Obtaining the answers to these questions can be an involved process. While a server must satisfy certain feature requirements (e.g., supporting Java or PHP), operators want to have some notion of how the server will perform, in order to do capacity planning. This in turn leads to purchasing decisions which are influenced by *benchmarks*. Many benchmarking programs are available for a wide variety of applications, such as databases and scientific programs, and Web servers are no exception.

Web server benchmarks are typically known as *load generators*, since the way they measure performance is to masquerade as clients and generate a synthetic load of HTTP requests for the server. Figure 1 shows a typical experimental setup, where clients running a workload generator program submit requests to a Web server. The idea is that, by measuring how the server performs in a controlled environment, operators will hopefully have a reasonably good notion of how that server will perform in the real world. In turn, by comparing the numbers for competing Web servers, operators can determine which will perform better. Of course, a Web server is more than simply the HTTP server software; a workload generator can be used to compare the performance of the same software running on two different pieces of hardware, or to compare two different operating systems running on the same hardware.

Many Web server benchmarks have appeared over the years, including WebStone [34], SPECweb96 [15], S-Client [7], SURGE [8], httperf [29], SPECweb99 [15], TPC-W [33], WAGON [26], and WaspClient [30]. As people's understanding of Web server behavior has expanded over time, so have the server benchmarks evolved and been improved. In addition to providing broad performance statistics, such as throughput in HTTP operations/second, many benchmarks attempt to ad-

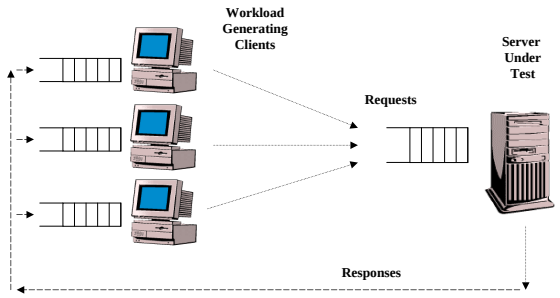

Figure 1: Workload Generation

dress characteristics that earlier ones ignored or were not aware of. For better or worse, however, people and organizations tend to settle on a single benchmark for comparative purposes. In the case of Web server benchmarks, particularly for hardware vendors, this benchmark is SPECweb99.

In this paper, we quantify how well SPECweb99 captures server workload characteristics such as request methods, response codes, URL popularity, and transfer size distributions. We find that SPECweb99 has a varying record depending on the characteristic, as we describe in detail below.

The remainder of this paper is organized as follows. In Section 2 we describe the SPECweb99 workload generator. Section 3 we present our methodology for comparisons. Section 4 presents our results in detail. Section 5 overviews related work. In Section 6 we summarize our findings and offer recommendations for ways for SPECweb99 to improve its accuracy.

2 SPECweb99

SPECweb99 has become the de-facto standard workload generator used in the computer industry for performance evaluation, robustness testing, and evaluating server optimizations. The popularity of SPECweb99 as a benchmark can be seen by the volume of results published at the SPEC Web site. At the time of this writing, 141 results have been published in under 3 years, which include hardware vendors such as Compaq, Dell, Fujitsu, HP, IBM, and Sun; operating systems such as AIX, HPUX, Linux, Solaris, and Windows NT; and Web server software such as Apache, IIS, Netscape Enterprise, and Tux. In contrast, for example, TPC-W has only 14 submissions from 3 vendors.

SPECweb99 is produced by the Systems Performance Evaluation Consortium (SPEC), an industry-backed non-profit corporation whose role is to provide independent benchmarks for comparative purposes. SPEC provides a number of benchmarks, such as SPECSFS97 for file systems, SPECCPU2000 for processor performance, and SPECweb99 for Web servers. SPEC is supported by its member corporations, such as HP, IBM, and Sun, and its benchmarks are decided on by a consensus process of the members. This process, while open to all members who make a financial contribution to SPEC, is closed to the general public. In addition, the source code to the benchmark is only available to member institutions or through a fee, which is typically discounted for non-profit organizations such as universities. In order for vendors to advertise the results of their SPECweb99 runs, they are required to submit their numbers to the SPEC Web site where the results are published.

Instead of a conventional performance metric such as server throughput or response time, the primary metric used by SPECweb99 is "number of simultaneous conforming connections." According to the web site:

> "SPECweb99 measures the maximum number of simultaneous connections requesting the predefined benchmark workload that a web server is able to support while still meeting specific throughput and error rate requirements. The connections are made and sustained at a specified maximum bit rate with a maximum segment size intended to more realistically model conditions that will be seen on the Internet during the lifetime of this benchmark." [15]

To ensure that connections are conforming, SPECweb99 monitors the bandwidth available to

Name:	Chess 1997	Olympics 1998	IBM 1998	World Cup 1998	Dept. Store 2000	IBM 2001	SPEC web99
Desc.:	Kasparov-Deep Blue Event Site	Sporting Event Site	Corporate Presence	Sporting Event Site	Online Shopping	Corporate Presence	Workload Generator
Period:	2 weeks in May 1997	1 day in Feb. 1998	1 day in June 1998	31 days in June 1998	12 days in June 2000	1 day in Feb. 2001	1 hour in June 2002
Hits:	1,586,667	11,485,600	5,800,000	1,111,970,278	13,169,361	12,445,739	10,481,768
KBytes:	14,171,711	54,697,108	10,515,507	3,952,832,722	54,697,108	28,804,852	151,495,102
Clients:	256,382	86,021	80,921	2,240,639	86,021	319,698	16
URLs:	2,293	15,788	30,465	89,997	15,788	42,874	55,054

Table 1: Logs used in examples

each connection and enforces both an upper and lower bound on that bandwidth, in order to emulate modem-connected clients. The lower bound is 320000 bits/sec or roughly 39 KB/sec; if a connection achieves lower than that bandwidth the run is discarded as "non-conforming." The upper bound is 400000 bits/sec or roughly 48 KB/sec; if the connection gets greater bandwidth the connection calls sleep() for the remainder of time that the connection "should" have taken.

SPEC has taken a number of steps to prevent "gaming" the benchmarking process and enforce a certain degree of realism. For example, the link-layer maximum transmission unit (MTU) must not be larger than 1500 bytes, presumably since over the wide-area Internet, it is virtually impossible to find a path MTU of greater than the Ethernet frame size. Enforcing this as the maximum prevents distorting results by using media that allow larger MTUs such as ATM or Gigabit Ethernet with Jumbo Frames. Similarly, SPEC enforces a maximum segment lifetime (MSL) of 60 seconds, which is used to manage the duration of TCP's TIME_WAIT state. Shrinking this value can artificially inflate server performance numbers [6]. Although RFC 793 [32] suggests an MSL value of 2 minutes, SPECweb99 uses a value of 60 seconds since that is what is used in "most BSD derived UNIX implementations."

While settling on a single Web server benchmark such as SPECweb99 allows relatively "apple-to-apple" comparisons, it has the disadvantage that a single particular benchmark may not capture the performance characteristics that are relevant to a particular Web site. While "relevant" may be a relative term, over time a certain number of characteristics have been discovered to be shared across a wide variety of Web sites. A straightforward question thus arises: how well does SPECweb99 capture the performance attributes that have been observed across a large number of sites?

3 Methodology

Our approach is to take various characteristics or performance metrics identified by the literature and compare them to those produced by SPECweb99. Characteristics are derived from the server logs generated from a SPECweb99 run. Then, for each characteristic, we conclude how well it is captured by SPECweb99.

The server used to produce the SPECweb99 log is an 8-way 900 MHz Pentium III symmetric multiprocessor (SMP), running Apache 2.0.36 and the Linux kernel version 2.4.17. The server has 32GB RAM and 4 Gigabit Ethernet adapters. Sixteen 866 MHz Pentium III PCs were used as clients to generate a load of 2500 concurrent connections.

To help illustrate the comparisons, we also present examples of the same characteristics derived from several logs. Table 1 gives an overview of the logs used in the examples, several of which are taken from high-volume web sites that were managed by IBM. One log, taken from an online department store, is taken from a site hosted by but not de-

Request Method	Chess 1997	Olymp. 1998	IBM 1998	W. Cup 1998	Dept. 2000	IBM 2001	SPEC web99
GET	92.18	99.37	99.91	99.75	99.42	97.54	95.06
HEAD	03.18	00.08	00.07	00.23	00.45	02.09	00.00
POST	00.00	00.02	00.01	00.01	00.11	00.22	04.93

Table 2: HTTP Request Methods (percent)

signed or managed by IBM. We also include most of the 1998 World Cup logs [4], which are publically available at the Internet Traffic Archive. Due to the size of these logs, we limit our analysis to the busiest 4 weeks of the trace, June 10th through July 10th (days 46 through 76 on the web site).

Since our analysis is based on web logs, certain interesting characteristics cannot be examined. For example, persistent connections, pipelining, network round-trip times and packet loss all have significant affects on both server performance and client-perceived response time. These characteristics are not captured in Apache Common Log format and typically require more detailed packet-level measurements using a tool such as tcpdump. These sorts of network-level measurements are difficult to obtain due to privacy and confidentiality requirements.

An important caveat worth reiterating is that any one Web site may not be representative of a particular application or workload. For example, the behavior of a very dynamic Web site such as eBay, which hosts a great deal of rapidly changing content, is most likely very different from an online trading site like Schwab, which conducts most of its business encrypted using the Secure Sockets Layer (SSL). Several example Web sites given here were all run by IBM, and thus may share certain traits not observed by previous researchers in the literature. As we will see, however, the characteristics from the IBM sites are consistent with those described in the literature.

Dynamic content [3, 12, 13, 23] is becoming a central component of modern transaction-oriented Web sites. SPECweb99 is unusual compared to most other workload generators in that 30 percent of the requests it produces are for dynamically generated content. While dynamic content generation is clearly a very important issue, there is currently no consensus as to what constitutes a "representative" dynamic workload, and so we do not evaluate SPECweb99 in this respect.

4 Results

In this Section we present our results in detail.

4.1 Request Methods

The first trait we examine is how well *request methods* are captured by SPECweb99. Several methods were defined in the HTTP 1.0 standard [10] (e.g., HEAD, POST, DELETE), and multiple others were added in the 1.1 specification [20, 21] (e.g., OPTIONS, TRACE, CONNECT). While a server should, of course, implement all methods required to support a particular HTTP standard, a workload generator should seek to recreate the request methods that are seen most frequently at a Web site.

Table 2 shows the percentage of request methods seen in the various logs. Here, only those methods which appear a non-trivial fraction are shown, defined in this case as greater than one hundredth of a percent. GET requests are the primary method by which documents are retrieved; the method "means retrieve whatever information ... is identified by the Request-URI" [10]. The HEAD method is similar to the GET method except that only meta-information about the URI is returned. The POST method is a request for the server to accept information from the client, and are typically use for for filling out forms and invoking dynamic content generation mechanisms. While different logs have slightly varying breakdowns, the vast majority of

Response Code	Chess 1997	Olymp. 1998	IBM 1998	W.Cup 1998	Dept. 2000	IBM 2001	SPEC web99
200 OK	85.32	76.02	75.28	79.46	86.80	67.73	100.00
206 Partial Content	00.00	00.00	00.00	00.06	00.00	00.00	00.00
302 Found	00.05	00.05	01.18	00.56	00.56	15.11	00.00
304 Not Modified	13.73	23.25	22.84	19.75	12.40	16.26	00.00
403 Forbidden	00.01	00.02	00.01	00.00	00.02	00.01	00.00
404 Not Found	00.55	00.64	00.65	00.70	00.18	00.79	00.00

Table 3: Server Response Codes (percent)

methods are GET requests, with a smaller but noticeable percentage being HEAD or POST methods. This is consistent with findings in the literature [25].

Table 2 also shows the percentage of methods used by SPECweb99. The majority of requests generated by SPECweb99 are GET requests, although the ratio, 95 percent, is slightly lower than those from 3 of the 4 real logs, which range from 97 to 99 percent. While the logs have a small percentage of HEAD methods, SPECweb99 does not generate any HEAD requests at all. Finally, the number of POST requests, while relatively small at 5 percent, is 2 orders of magnitude greater than the fraction seen in the real sites, which have POST methods on the order of hundredths of a percent. SPECweb99 uses POSTs heavily for its dynamic content request generation. However, the mismatch here implies either that SPECweb99 is over-estimating the amount of dynamic content on a Web site, or that sites use other methods for that purpose, e.g., a GET request for a URL with a cgi-bin prefix.

Since SPECweb99 generates a first-order approximation of request methods, we conclude that it does a passable job in terms of capturing this metric. However, some refinement, particularly in terms of generating HEAD requests, seems necessary.

4.2 Response Codes

The next characteristic we study are the *response codes* generated by the server. Again, the HTTP specifications define a large number of responses, the generation of which depends on multiple factors such as whether or not a client is allowed access to a URL, whether or not the request is properly formed, etc. However, certain responses are much more frequent than others.

Table 3 shows those responses seen in the logs that occur with a non-trivial frequency, again defined as greater than one hundredth of a percent. We see that the majority of the responses are successful transfers, i.e., the 200 OK response code.

Perhaps the most interesting aspect of this data is, however, the relatively large fraction of 304 Not Modified responses. This code is typically generated in response to a client generating a GET request with the If-Modified-Since option, which provides the client's notion of the URL's last-modified time. This request is essentially a cache-validation option and asks the server to respond with the full document if the client's copy is out of date. Otherwise, the server should respond with the 304 code if the copy is OK. As can be seen, between 12 and 23 percent of responses are 304 codes, indicating that clients re-validating up-to-date content is a relatively frequent occurrence, albeit in different proportions at different Web sites.

Other responses, such as 403 Forbidden or 404 Not Found, are not very frequent, on the order of a tenth of a percent, but appear occasionally. The IBM 2001 log is unusual in that roughly 15 percent of the responses use the 302 Found code, which is typically used as a temporary redirection facility. While we are unaware of the cause of this behavior, we believe it is a rudimentary content distribution mechanism.

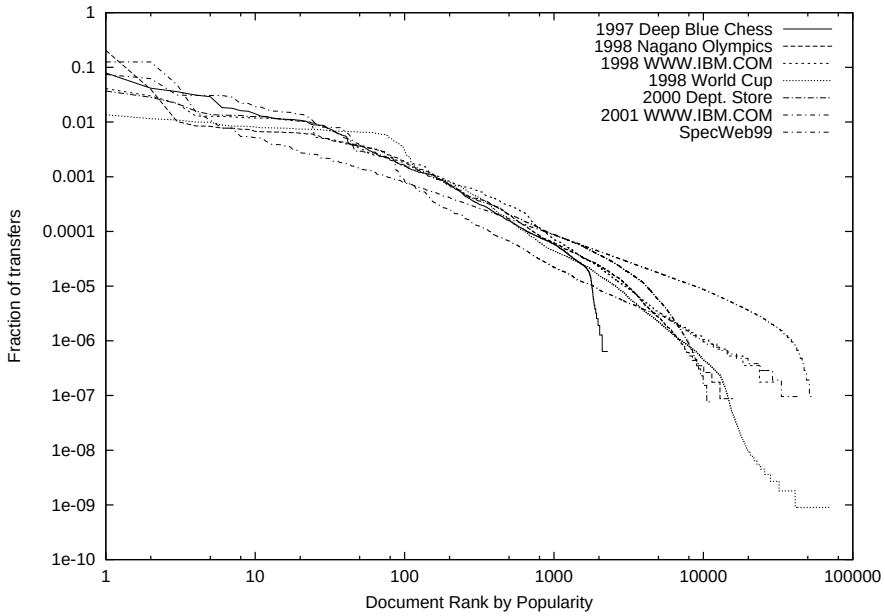

Figure 2: Document Popularity

Table 3 also shows the response codes generated during the SPECweb99 run. Here, we see that all responses are 200 OK codes. While it might be argued that certain error codes such as Forbidden or Not Found should not be included in a benchmark, the relative frequency of 304 responses indicates that they are normal part of Web server behavior and should be captured by a workload generator. Thus we conclude that SPECweb99 is not representative in this respect.

4.3 Object Popularity

Numerous researchers [2, 5, 16, 31] have observed that, in origin Web servers, the relative probability with which a web page is accessed follows a Zipf-like distribution. That is,

$$p(r) \approx C/r^{\alpha}$$

where $p(r)$ is the probability of a request for a document with rank r, and C is a constant (depending on α and the number of documents) that ensures that the sum of the probabilities is one. Rank is defined by popularity; the most popular document has rank 1, the second-most popular has rank 2, etc. When α equals 1, the distribution is a true Zipf; when α is another value the distribution is considered "Zipf-like." Server logs tend to have α values of one or greater; proxy server logs have lower values ranging from 0.64 to 0.83 [11].

Figure 2 shows the fraction of references based on document rank generated from the sample Web logs and the log produced by SPECweb99. Note that both the X and Y axes use a log scale. As can be seen, all the curves follow a Zipf-like distribution fairly closely, except towards the upper left of the graph and the lower right of the graph.

This Zipf property of document popularity is significant because it shows the effectiveness of document caching. For example, one can see that by simply caching the 100 most popular documents, assuming these documents are all cacheable, the vast majority of requests will find the document in the cache, avoiding an expensive disk I/O operation.

Since the curve for SPECweb99 follows the Zipf-like distribution as closely as the other curves, we say the SPECweb99 benchmark captures this characteristic effectively.

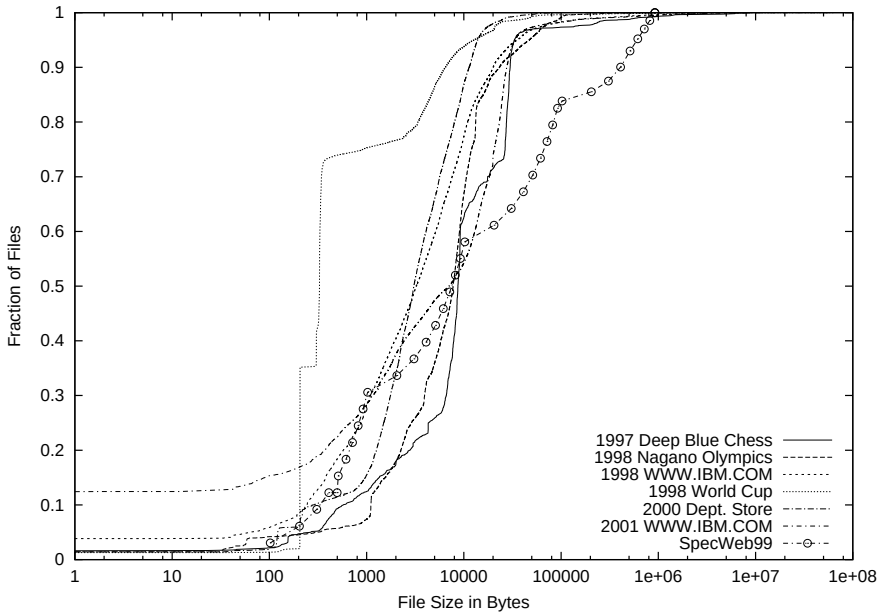

Figure 3: Document Size (CDF)

4.4 File Sizes

The next characteristic we examine is the range of sizes of the URLs stored on a Web server. File sizes give a picture of how much storage is required on a server, as well as how much RAM might be needed to fully cache the data in memory. Which distribution best captures file size characteristics has been a topic of some controversy. There is consistent agreement that sizes range over multiple orders of magnitude and that the body of the distribution (i.e., that excluding the tail) is Log-Normal. However, the shape of the tail of the distribution has been debated, with claims that it is Pareto [16], Log-Normal [19], and even that the amount of data available is insufficient to statistically distinguish the the two [22].

Figure 3 shows the CDF of file sizes seen in the logs, as well as those from SPECweb99. Note that the X axis is in log scale. As can be seen, sizes range from 1 byte to 10 megabytes, varying across 7 orders of magnitude. In addition, the distributions show the rough 'S' shape of the Log-Normal distribution.

Figure 3 also shows the CDF of file sizes from SPECweb99. While the distribution is roughly Log-Normal, it does not vary as widely as the curves from the logs. This is because the smallest file requested in SPECweb99 contains 102 bytes, and the largest 921600 bytes. Perhaps more significantly, the curve is significantly less "smooth" than the curves from the logs. The SPECweb99 curve has interpolation lines present, whereas the curves from the real logs consist only of actual data points. This is because the SPECweb99 file set generated on the server has only 36 distinct file sizes, whereas files on real Web sites exhibit much greater variety.

As mentioned earlier, a metric of frequent interest in the research community is the "tail" of the distribution. While the vast majority of files are small, the majority of bytes transferred are found in large files. This is sometimes known as the "Elephants and Mice" phenomenon. To see how well SPECweb99 captures this property, we graphed the complement of the cumulative distribution function, or CCDF, of the logs. These are shown in Figure 4. The Y values for this graph are essentially the complement of the corresponding Y values from Figure 3. Unlike Figure 3, however, note

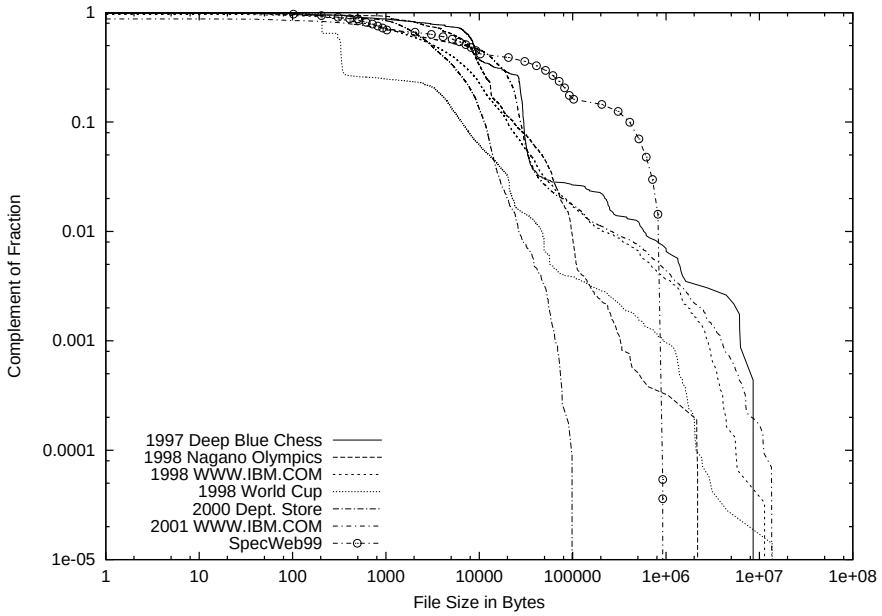

Figure 4: Document Size (CCDF)

here that the Y-axis uses a log scale to better illustrate the tail. We observe that all the logs have maximum files in the range of 1 to 10 MB, with the exception of the Department Store log, which has no file size greater than 99990 bytes.

Figure 4 also shows the CCDF from SPECweb99. As can be seen, the tail exhibited by SPECweb99 does not reach as far as the tails from most of the other logs, except for the Department Store log. Again, this is because SPECweb99 does not request any files greater than 921600 bytes.

To summarize, SPECweb99 does capture the broad characteristics of file size distributions, by varying across several orders of magnitude and exhibiting a Log-Normal distribution. However, it fails to capture the deeper aspects of the distributions, such as smoothness and length in the tail. We thus conclude that, in this case, SPECweb99 does a passable job, albeit one that could be improved.

4.5 Transfer Sizes

A metric related to Web file sizes is Web *transfer sizes*, or the size of the objects sent "over the wire." Transfer sizes are significant since they connote how much bandwidth is used by the Web server to respond to clients. In Apache Common Log Format, transfer size is based on the amount of *content* sent, and does not include the size from any HTTP headers or lower-layer bytes such as TCP or IP headers. Thus, here transfer size is based on the size of the content transmitted. The distribution of transfer sizes is thus influenced by the popularity of documents requested, as well as by the proportion of unusual responses such as 304 Not Modified and 404 Not Found.

Figure 5 shows the CDF of the object transfers from the logs. As can be seen, transfers tend to be small; for example, the median transfer size from the IBM 2001 log is roughly 230 bytes! An important trend is to note that a large fraction of transfers are for *zero* bytes, as much as 28 percent in the 1998 IBM log. The vast majority of these zero-byte transfers are the 304 Not Modified responses noted above in Section 4.2. When a conditional GET request with the If-Modified-Since option is successful, a 304 response is generated and no content is transferred. Other return codes, such as 403 Forbidden and 404 Not Found, also result in zero-byte transfers, but

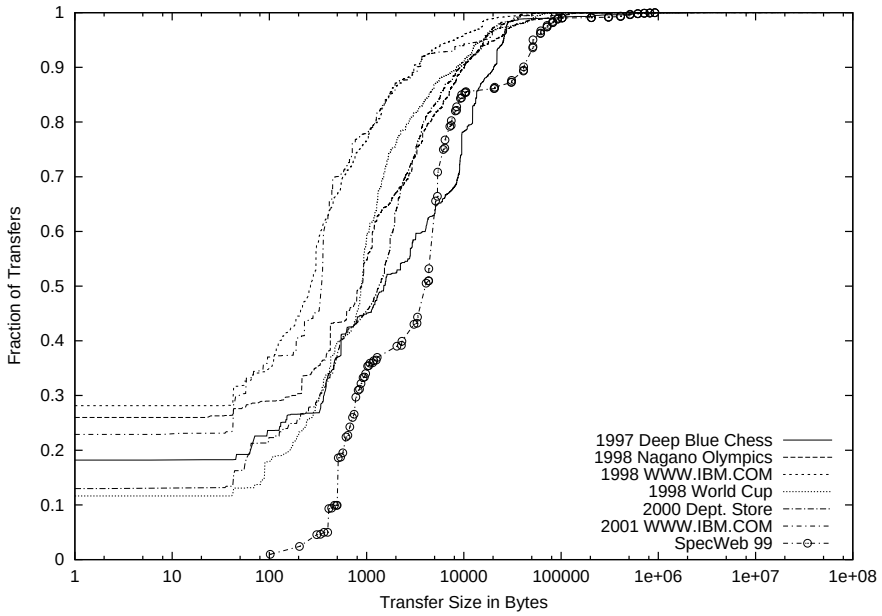

Figure 5: Transfer Size (CDF)

they are significantly less common. The exception is the IBM 2001 log, where roughly 15 percent of the 302 Found responses contribute to the fraction of zero-byte transfers.

Figure 5 also includes the transfer size distribution from SPECweb99. In this case, the curve does not match well the other curves from the logs. In addition to the smoothness discrepancy noted in Section 4.4, the sizes tend to be larger than observed from the logs. For example, the median transfer size is 5120 bytes, compared to the 230 bytes in the IBM 2001 log. Perhaps most significantly, since SPECweb99 does not produce any conditional GET requests with the If-Modified-Since header, no 304 responses are generated, and thus zero-byte transfers do not occur. This has a significant impact on server performance, since in the case where a 304 is generated, the server will have less work to do, as the file will not be sent to the client.

Figure 6 shows the transfer size distributions when 304 responses are removed from the logs. The original SPECweb99 transfer distribution is also included. As can be seen, removing the 304 responses reduces the number of zero-byte transfers significantly. From this perspective, SPECweb99 appears more representative, although the transfer sizes still tend to be higher than those from the logs, at least in the body of the distribution.

Figure 7 shows the CCDF of the transfer sizes, in order to illustrate the "tail" of the distributions. Note again that the Y axis uses a log scale. The graph looks similar to Figure 4, perhaps since these transfers are so uncommon that weighting them by frequency does not change the shape of the graph, as it does with the bulk of the distribution in Figure 5. As in Figure 4, the tail from SPECweb99 drops off earlier than the tails from the other logs, with the exception again of the Department Store log.

Since the transfer sizes produced by SPECweb99 are so disparate from those seen in the logs, we conclude that the workload generator is not representative in terms of this metric.

4.6 HTTP Version

Another question we are interested in is what sort of HTTP protocol support is available in SPECweb99. While HTTP 1.1 was first standardized in 1997

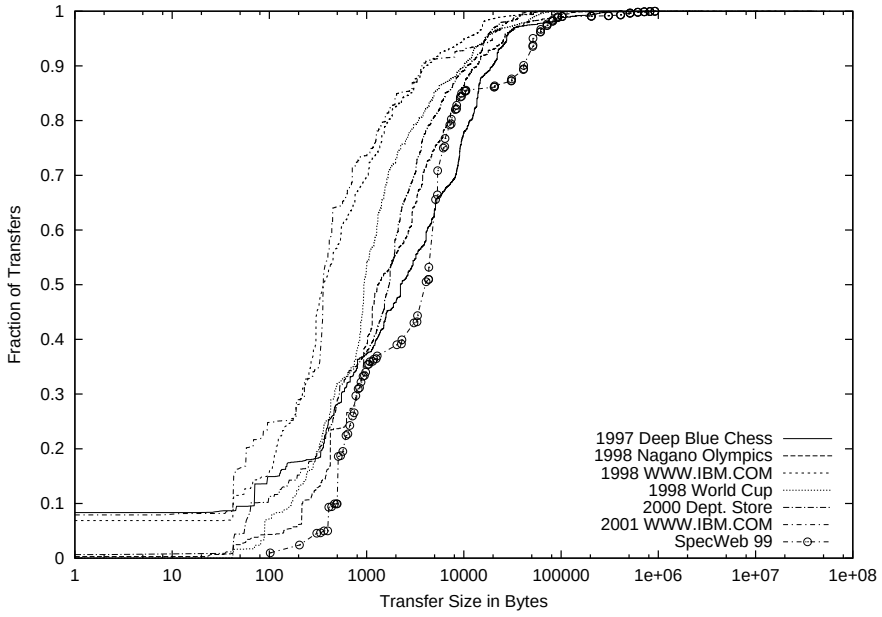

Figure 6: Transfer Size excluding 304s (CDF)

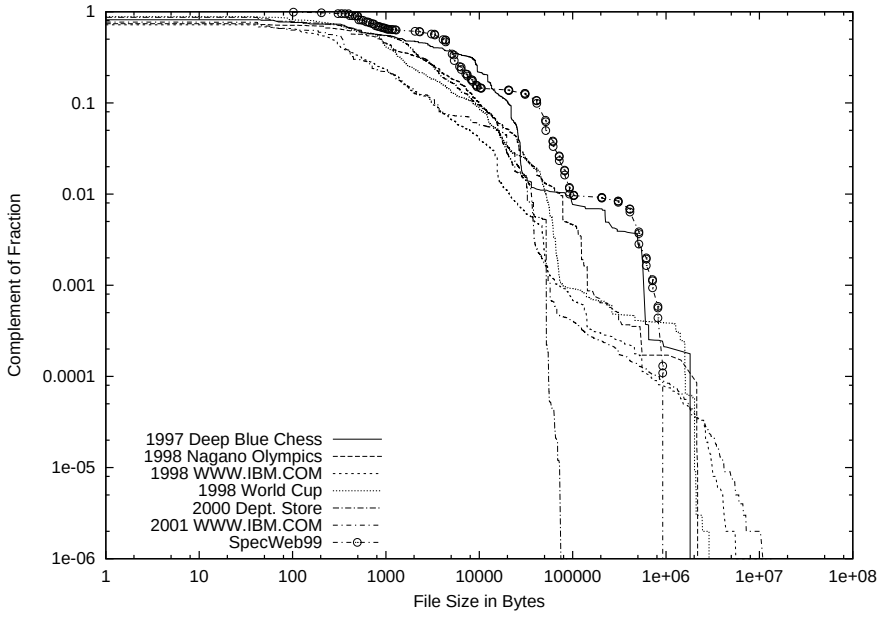

Figure 7: Transfer Size (CCDF)

Protocol Version	Chess 1997	Olymp. 1998	IBM 1998	W. Cup 1998	Dept. 2000	IBM 2001	SPEC web99
HTTP 1.0	95.30	78.56	77.22	78.62	51.13	51.08	30.00
HTTP 1.1	00.00	20.92	18.43	21.35	48.82	48.30	70.00
Unclear	04.70	00.05	04.34	00.02	00.05	00.06	00.00

Table 4: HTTP Protocol Versions (percent)

[20], the protocol has undergone some updating [21, 25] and in some ways is still being clarified [24, 28]. The transition from 1.0 to 1.1 is a complex one, requiring support from browsers, servers, and any proxy intermediaries as well.

Table 4 shows the HTTP protocol version that the server used in responding to requests. A clear trend is that over time, more requests are being serviced using 1.1. However, SPECweb99 99 appears to utilize 1.1 more heavily than the real sites do. In the most recent logs, from 2000 and 2001, HTTP 1.1 is used in just under half the responses, whereas SPECweb99 uses HTTP 1.1 for 70 percent of the responses. This is because the SPECweb99 code utilizes HTTP 1.1-style persistent connections for 70 percent of the connections, and submits between 5 and 15 requests on each connection before closing it. SPECweb99 does not currently support more advanced HTTP 1.1 features such as pipelining, range requests, or chunked encoding. While real sites will most likely reach and pass the 70 percent mark, SPECweb99 was perhaps overly optimistic about the prospects for deploying 1.1, especially for the time it was released, in August of 1999.

Given the depth and complexity of the HTTP 1.1 protocol, the numbers above only scratch the surface of how servers utilize HTTP 1.1. Many features have been added in 1.1, including new mechanisms, headers, methods, and response codes. How these features are used in practice is still an open issue, and as mentioned earlier, server logs are insufficient to fully understand HTTP 1.1 behavior. Thus, we make no claims about "representativeness" here, but include the numbers for interest.

4.7 Request Inter-Arrival Times

The final metric we examine is the inter-arrival time between requests. Previous research [5, 4, 25] has tended to look at the inter-arrival times between requests from the same users, where users are typically identified by the remote IP address. Since the SPECweb99 log has only 16 IP addresses, we do not look at inter-arrival times in that way here. Instead, we look at inter-arrival times between successive requests *across* all users. This gives an idea of the load or intensity of the traffic.

One problem with the Apache Common Log format is that entries have a time granularity of only one second. Thus, it is impossible to distinguish the inter-arrival times of multiple requests that arrive within the same second. To approximate the inter-arrival times in these cases, we assume that arrivals are evenly spread within one second. For example, if the log shows 5 requests arrive in the same second, we treat that as 5 data points for 200 millisecond inter-arrival times. While somewhat arbitrary, we feel it does give an indication of the intensity of the traffic.

Figure 8 shows the CDF of the inter-arrival times generated from the logs. Note that the X axis is in log scale. Inter-arrival times appear to follow a log-normal distribution, with different logs exhibiting different average times. The busiest traces, the World Cup and Olympic logs, for example, have median inter-arrival times of 1.9 and 1.6 milliseconds respectively, whereas the 2000 Department Store log median time is 55 milliseconds.

Figure 8 also shows the CDF of the inter-arrival times generated from the SPECweb99 logs. Here the median inter-arrival time is 0.14 milliseconds,

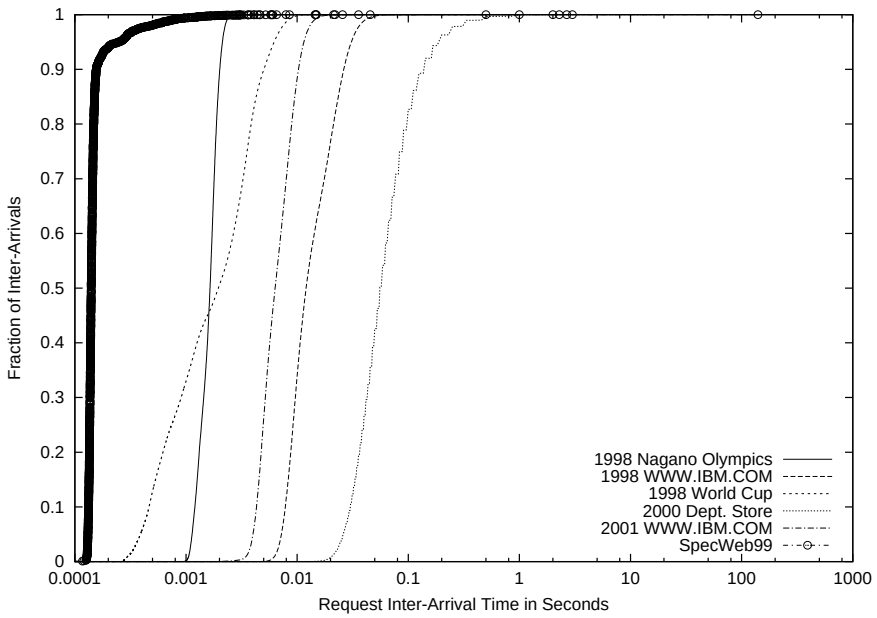

Figure 8: Inter-Arrival Times

an order of magnitude smaller than the busiest sites in the traces.

SPECweb99 is useful in this context in that it can generate a request rate that is much higher than is observed in even the busiest web sites. However, it may not capture the effects of long-lived client sessions. More importantly, connection and session duration are most likely underestimated, since network characteristics such as loss and delay are unaccounted for.

Since the research literature uses a different notion of inter-arrival times, we again make no claims about the accuracy of SPECweb99 in this context.

5 Related Work

Web server workload generation has been a topic of interest for several years. In this section we provide a brief overview of the area.

The first well-known workload generator was WebStone [34], which was released in 1995. It did only simple request generation for 5 unique files, ranging from 500 bytes to 5 MB.

In 1996, SPEC released their first-generation workload generator, titled SPECweb96 [14]. SPECweb96 introduced the notion of scaling with load and made an attempt to be more representative, claiming the workload was based on logs from several active sites. No validation was presented, however, and SPECweb96 had a relatively small set of file sizes in 4 categories.

S-Client [7] introduced the notion of *open-loop* workload generation. Most workload generators are *closed-loop* in that they are tightly coupled to the server in that they are measuring, by waiting to generate a new request only after a response has been received. By generating requests at a rate orthogonal to the capacity of the server, S-Client can generate loads that overload the server.

SURGE [8] was the first workload generator that paid special attention to reproducing server workload characteristics such as document popularity, embedded references, and transfer size. SURGE originally only captured HTTP 1.0 traffic, but was subsequently extended to implement persistent connections and pipelining [9].

httperf [29] is a workload generator which has the main goal of extensibility and broad correctness testing. httperf allows a wide variety of options and configurations to probe a range of scenarios on the server. httperf also includes the open-loop model used by S-Client.

WaspClient [30] is a hybrid workload generator built by using the workload model from SURGE with the request generation engine from S-Client. WaspClient was used to evaluate server performance in wide-area conditions, rather than over a LAN.

TPC-W [33] is a specification for generating a workload for a full Web site, as opposed to just a single Web server. It seeks to model an online bookstore similar to Amazon.com, and thus stresses components such as the Java-based application server and the back-end database. TPC-W is only a specification, however; no source code is provided. While TPC-W is likely a growing influence on Web server performance evaluation, it does not yet have the critical mass from hardware vendors that SPECweb99 does. For example, at the time of this writing, the TPC-W web site lists only 14 submissions from 3 vendors.

6 Conclusions and Future Work

We have presented a range of Web server workload characteristics and examined how well they are captured by the industry-standard benchmark SPECweb99. The accuracy of SPECweb99 varies depending on the characteristic; for example, document popularity is captured well, whereas transfer size is modeled relatively poorly. In order to improve the accuracy of the benchmark, two main issues need to be addressed.

First, conditional GET requests with the If-Modified-Since header should be included. This will both allow more representative response codes and capture small transfers more effectively. This will require augmenting the workload generator to keep track of last modified times returned by the server and include those times in subsequent conditional GET requests. While this can be done using a simple statistical model, a more accurate method will require a model of object cachability and modification times.

Second, the file and transfer size distributions need to be improved. The range of file sizes should be increased, in order to "fill out" the distribution to make it more smooth. At the same time, average file sizes should shrink to better match the smaller transfer sizes exhibited by real sites. Finally, the file size distribution should be extended to better capture the "tail" of the distribution and very long transfers. The largest file currently used in SPECweb99 is less than 1 MB; it should be at least 10 MB.

Several possibilities exist for future work; we briefly outline them here.

SURGE [8] was designed to reproduce many of the characteristics described in this paper. While we have focused on SPECweb99 due to its popularity in industry, a clear next step would be to evaluate SURGE using the same metrics.

HTTP 1.1 introduces a number of new mechanisms to the Web, such as pipelining, chunked encoding, and range requests. As HTTP 1.1 deployment widens and its use on servers increases, a well-grounded and quantified understanding of how these features are used in practice will be needed. In turn, workload generators such as SPECweb99 will need to accurately capture the use of these mechanisms if they are to remain representative.

Similarly, Web sites that use dynamic content remains a relatively unexplored area. Workload characterization studies are required in order to better understand the behavior of these sites. Unfortunately, this will not be easy, as commercial Web site owners wish to protect both their users' privacy and their business application software. They will be reluctant to share information that they view as a competitive trade secret, such as application servlets and back-end databases.

Acknowledgments

Several people provided valuable feedback that improved the presentation of this work, including the anonymous referees, Fred Douglis, Tal Malkin, Katherine Nahum, and Vivek Pai. Thanks to Troy Wilson of the IBM Linux Technology Center for providing a large-scale SPECweb99 log.

References

[1] Jussara M. Almeida, Virgilio Almeida, and David J.Yates. Measuring the behavior of a World-Wide Web server. In *Seventh IFIP Conference on High Performance Networking (HPN)*, White Plains, NY, April 1997.

[2] Virgilio Almeida, Azer Bestavros, Mark Crovella, and Adriana de Oliveira. Characterizing reference locality in the WWW. In *Proceedings of PDIS'96: The IEEE Conference on Parallel and Distributed Information Systems*, Miami Beach, Florida, December 1996.

[3] Cristiana Amza, Emmanuel Cecchet, Anupam Chanda, Sameh Elnikety, Alan Cox, Romer Gil, Julie Marguerite, Karthick Rajamani, and Willy Zwaenepoel. Bottleneck characterization of dynamic Web site benchmarks. Technical Report TR02-388, Rice University Computer Science Department, February 2002.

[4] Martin F. Arlitt and Tai Jin. Workload characterization of the 1998 world cup Web site. *IEEE Network*, 14(3):30–37, May/June 2000.

[5] Martin F. Arlitt and Carey L. Williamson. Internet Web servers: Workload characterization and performance implications. *IEEE/ACM Transactions on Networking*, 5(5):631–646, Oct 1997.

[6] Mohit Aron and Peter Druschel. TCP implementation enhancements for improving Webserver performance. Technical Report TR99-335, Rice University Computer Science Dept., July 1999.

[7] Gaurav Banga and Peter Druschel. Measuring the capacity of a Web server. In *Proceedings of the USENIX Symposium on Internet Technologies and Systems (USITS)*, Monterey, CA, Dec 1997.

[8] Paul Barford and Mark Crovella. Generating representative Web workloads for network and server performance evaluation. In *Proceedings of the ACM Sigmetrics Conference on Measurement and Modeling of Computer Systems*, Madison, WI, June 1998.

[9] Paul Barford and Mark Crovella. A performance evaluation of hyper text transfer protocols. In *Proceedings of the ACM Sigmetrics Conference on Measurement and Modeling of Computer Systems*, Atlanta, GA, May 1999.

[10] Tim Berners-Lee, Roy Fielding, and Henrik Frystyk. Hypertext transfer protocol – HTTP/1.0. In *IETF RFC 1945*, May 1996.

[11] Lee Breslau, Pei Cao, Li Fan, Graham Phillips, and Scott Shenker. Web caching and Zipf-like distributions: Evidence and implications. In *Proceedings of the Conference on Computer Communications (IEEE Infocom)*, New York, NY, March 1999.

[12] Emmanuel Cecchet, Anupam Chanda, Sameh Elnikety, Julie Marguerite, and Willy Zwaenepoel. A comparison of software architectures for e-business applications. Technical Report TR02-389, Rice University Computer Science Department, February 2002.

[13] Jim Challenger, Arun Iyengar, Karen Witting, Cameron Ferstat, and Paul Reed. A publishing system for efficiently creating dynamic Web content. In *Proceedings of the Conference on Computer Communications (IEEE Infocom)*, Tel-Aviv, Israel, March 2000.

[14] The Standard Performance Evaluation Corporation. SPECWeb96. `http://www.spec.org/osg/web96`, 1996.

[15] The Standard Performance Evaluation Corporation. SPECWeb99. `http://www.spec.org/osg/web99`, 1999.

[16] Mark Crovella and Azer Bestavros. Self-similarity in World Wide Web traffic: Evidence and possible causes. *IEEE/ACM Transactions on Networking*, 5(6):835–846, Nov 1997.

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

[18] Jacques Derrida. *Of Grammatology*. Johns Hopkins University Press (1st American Ed.), 1977.

[19] Allen Downey. The structural cause of file size distributions. In *Proceedings of the Ninth International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS)*, Cincinnati, OH, Aug 2001.

[20] Roy Fielding, Jim Gettys, Jeffrey Mogul, Henrik Frystyk, and Tim Berners-Lee. Hypertext transfer protocol – HTTP/1.1. In *IETF RFC 2068*, January 1997.

[21] Roy Fielding, Jim Gettys, Jeffrey Mogul, Henrik Frystyk, Larry Masinter, Paul Leach, and Tim Berners-Lee. Hypertext transfer protocol – HTTP/1.1. In *IETF RFC 2616*, June 1999.

[22] Weibo Gong, Yong Liu, Vishal Misra, and Don Towsley. On the tails of Web file size distributions. In *Proceedings of the 39th Allerton Conference on Communication, Control, and Computing*, Monticello, Illinois, Oct 2001.

[23] Arun Iyengar and Jim Challenger. Improving Web server performance by caching dynamic data. In *Proceedings of the USENIX Symposium on Internet Technologies and Systems*, Monterey, CA, December 1997.

[24] Balachander Krishnamurthy, Jeffrey C. Mogul, and David M. Kristol. Key differences between HTTP/1.0 and HTTP/1.1. In *Proceedings of WWW-8 Conference*, Toronto, Canada, May 1999.

[25] Balachander Krishnamurthy and Jennifer Rexford. *Web Protocols and Practice*. Addison Wesley, 2001.

[26] Zhen Liu, Nicolas Niclausse, and Cesar Jalpa-Villanueva. Traffic model and performance evaluation of Web servers. *Performance Evaluation*, 46(2-3):77–100, 2001.

[27] Bruce Mah. An empirical model of HTTP network traffic. In *Proceedings of the Conference on Computer Communications (IEEE Infocom)*, Kobe, Japan, Apr 1997.

[28] Jeffrey C. Mogul. Clarifying the fundamentals of HTTP. In *Proceedings of WWW 2002 Conference*, Honolulu, HA, May 2002.

[29] David Mosberger and Tai Jin. httperf – a tool for measuring Web server performance. In *Proceedings 1998 Workshop on Internet Server Performance (WISP)*, Madison, WI, June 1998.

[30] Erich M. Nahum, Marcel Rosu, Srinivasan Seshan, and Jussara Almeida. The effects of wide-area conditions on WWW server performance. In *Proceedings of the ACM Sigmetrics Conference on Measurement and Modeling of Computer Systems*, Cambridge, MA, June 2001.

[31] Venkata N. Padmanabhan and Lili Qui. The content and access dynamics of a busy Web site: Findings and implications. In *ACM SIGCOMM Symposium on Communications Architectures and Protocols*, pages 111–123, 2000.

[32] Jon Postel. Transmission control protocol. *IETF RFC 793*, pages 1–85, September 1981.

[33] The Transaction Processing Council (TPC). TPC-W. `http://www.tpc.org/tpcw`.

[34] Gene Trent and Mark Sake. WebStone: The first generation in HTTP server benchmarking. `http://www.sgi.com/Products/WebFORCE/WebStone`, 1995.