

PERFORMANCE ANALYSIS OF A WWW SERVER

Virgílio Almeida *

Jussara de Almeida

Cristina Murta

{virgilio, jussara,
cristina}@dcc.ufmg.br
Departamento de Ciência da
Computação
Universidade Federal de Minas Gerais
30161-970 Belo Horizonte
Brazil

This paper addresses the problem of analyzing performance of WWW servers. The web has experienced a phenomenal growth and has become the most popular Internet application. As a consequence of its large popularity, the Internet has suffered from various performance problems, such as network congestion and overloaded servers. These days, it is not uncommon to find servers refusing connections because they are overloaded. Performance has always been a key issue in the design and operation of on-line systems. With regard to Internet, performance is also critical, because users want fast and easy access to all objects (i.e., documents, pictures, audio, and video) available on the net. Thus, it is important to understand WWW performance issues. This paper focuses on the performance analysis of a Web server. Using a synthetic benchmark (WebStone), we analyze three different Web server software running on top of a Windows NT platform and performing some typical WWW tasks

1. INTRODUCTION

The World Wide Web (WWW or Web) is a client-server architecture that integrates various types of information on the global Internet and on corporate IP (Internet Protocol) networks. The WWW allows users to retrieve text and multimedia objects from servers located throughout the world, with objects connected by hypermedia links.

* On sabbatical at the Computer Science Department, Boston University. E-mail: virgilio@cs.bu.edu

In the past two years, the Web has experienced a phenomenal growth and has become the most popular Internet application. In addition to that, many corporations and information systems (IS) departments found the internal Internet, also called Intranet, as an effective means of distributing information within the organization. As a consequence of its large popularity, the Internet has suffered from various performance problems, such as network congestion and overloaded servers. These days, it is not uncommon to find servers refusing connections because they are overloaded [10]. Performance has always been a key issue in the design and operation of on-line systems. With regard to Internet, performance is also critical, because users want fast and easy access to all objects (i.e., documents, pictures, audio, and video) available on the net. From the information provider viewpoint, performance is also a key issue, because the value of a Web server is associated with the number of people that visit the site in a given period of time, i.e., the number of hits recorded on the server. A long delay to service a request is a factor that discourages people to visit a site. Thus, it is important to understand WWW performance issues. This paper focuses on the performance analysis of a Web server.

There are a number of factors which are of interest in the analysis and evaluation of a Web server functionality and design. They include issues such as management tools, security, ease of use, authoring features and performance. Rather than attempting to analyze all aspects of a WWW server, our paper concentrates on issues that are relevant to server performance. Our goal is to analyze different Web server software on the same PC hardware, performing some WWW typical tasks. Our tests do not intend to rank server software in terms of its performance. The relative behavior of the servers on identical tasks is more important to us than absolute best performance that could be achieved for any individual system. For comparison purposes and because we do not have access to source code of the servers, our analysis is based on the "black box approach". We usually attempt to explain curious results through external testing, rather than examining of the code.

There are few papers on performance analysis of WWW servers. References [5,9,11,12] study workload characterization for WWW servers and clients. Basically, they show the types of objects available at WWW servers and analyze relevant aspects such as file size distribution, inter-reference time distribution and file popularity. Reference [12] examines characteristics of a large number of HTML (HyperText Markup Language) documents, collected by some Web crawlers (e.g., Inktomi, Lycos, etc). Techniques to characterize and model performance of client/server systems in general are presented in details in [1]. This paper focuses on performance analysis of WWW servers and is organized as follows. Section two discusses the WWW architecture and its main components, such as the HTTP protocol, the server software structure and some relevant performance measures. In section three, we describe a simple methodology of a performance analysis for Web servers. Section four describes the hardware, operating system (i.e., Windows NT), server software (i.e., Emwac, Purveyor, and Website) and workload (i.e., WebStone) used in our experimental environment. The results obtained are analyzed and interpreted in section five. Finally, concluding remarks appear in section six.

2. WORLD WIDE WEB PERFORMANCE

The World-Wide Web is a client-server framework that integrates various types of information on the global Internet [4]. It is a combination of Web Browsers and Web Servers that communicate using the HyperText Transfer Protocol (HTTP), which is layered on top of TCP/IP protocol. Browsers, such as Netscape, Mosaic, and Microsoft Explorer, provide an easy-to-use graphical interface for viewing pages and documents in the Internet. A Web server (also called HTTP Server) sends information (pages, images, etc) back to clients in response to their HTTP requests. It also allows the integration of various sources of information through Common Gateway Interface (CGI) programs that perform general computation in response to client requests. HTTP is a lightweight, stateless protocol. Figure 1 depicts a client-server interaction for the HTTP protocol. The client establishes the connection to the server and interacts with it using different methods defined in the protocol [4]. Basically, it sends a request for an object (e.g., document, image, database search, etc). The server takes a while to process the request and returns the object or results of the request. Objects are addressed by their Uniform Resource Locator (URL).

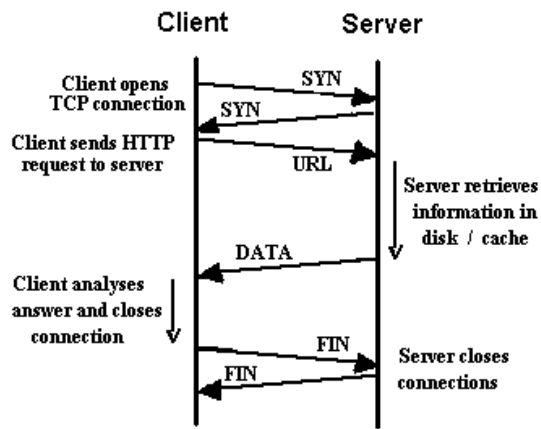


Figure 1: A simple client-server interaction

A Web server program generally has several client requests in progress at the same time, like a multiprogrammed operating system. Some servers implement that kind of multiprogrammed execution by forking a process or thread for each connection as it arrives. Depending on the implementation, the cost of forking a new process may be high and may cause unacceptable overhead per connection. Other servers attempt to minimize the overhead problem by implementing a mechanism known as a “pool of processes”, where a number of processes are created during initialization. The time to process an HTTP request depends on the server speed and on the request complexity. The former is a function of the hardware configuration (processor clock, memory architecture and I/O subsystem) and the latter depends on the size of the document and the amount of computation required by a CGI program. Time and rate are the basic measures for performance of a server. The rate at which HTTP requests are serviced represent the throughput (also called connection rate) and the time required to process a request is the response time (also called latency). Because the size of the objects vary significantly, throughput is usually measured in terms of bytes/sec.

3. PERFORMANCE ANALYSIS METHODOLOGY

The benefits of performance analysis of WWW servers are multiple. The first one is to assess the service level provided by a server, in terms of response time, error rate, and throughput. Other benefits are the following: to assess the server resource usage in order to identify bottlenecks, to anticipate performance problems, and to understand the influence of system’s features (e.g., maximum number of connections, file placement, processor scheduling policies, etc) on server performance. The latter can help system administrators to tune up the software and hardware parameters. The ultimate benefit of carrying out a performance analysis effort is to determine the capacity of a Web server, which can be defined as the largest throughput (in terms of hits/sec) at which the HTTP request execution time remains acceptable (e.g., 98% of requests serviced in less than 0.1 sec).

Web server performance evaluation is complex and depends on the server hardware, operating system, HTTP software, network speed, and workload. There exist various well-known methodologies for performance assessment of computer systems, as pointed out in [1]. However, there are significant differences between conventional computing environments and the WWW. First of all, the number of potential Web clients is in the tens of millions [3]. The Web is also characterized by a large diversity in terms of its components: different browsers and servers running on a variety of hardware platform, connected to the Internet at several different speeds. The workload consists of requests for different types of information such as text, graphics, video, and audio. Thus, it is important to adapt existing techniques [1,8] to the Web environment. In this section, we discuss a series of major steps to carry out a performance evaluation of a WWW server, namely:

- understanding the server environment

- monitoring the server operation
- characterizing the HTTP workload
- analyzing the server performance and capacity

The first step in any performance analysis project is to obtain a big picture of the environment. The main question in this step is to determine the purpose of the server, which implicitly defines the service level to be provided to users and the required performance. For instance, we need to answer questions such as: what is the potential number of clients for the information content available in the site? Is the site going to “sell” or offer information for free? Answers to these questions help to establish the site service level. Then, we need to identify the key components of a Web server, which altogether create the environment for processing HTTP requests. The hardware configuration includes processors, memory, disk and control units, and the network interface unit. The software consists of the operating system (and the TCP/IP protocol implementation) and the Web server. Another key element is the capacity (in Mbits/sec.) of the network link that connects the server to the Internet or the corporate network.

The main source of information for performance studies is the data collected from the observation of the server’s operation. Server behavior can be monitored by operating system’s tools, such as Unix/sar or the NT/Performance Monitor [1]. Those monitors provide resource usage information. In some cases, depending on the level of integration between the server and the operating system, information such as throughput and latency is also provided by monitor tools. Web servers can be configured to record information about all client requests. The access logs have one line of information per request processed by a server. Basically, each line contains the name of the host making the request, the timestamp the request was made, the filename of the requested object and its size in bytes. Logs are useful for workload characterization. The three most important parameters to characterize Web workloads, from the server standpoint, are the types of objects, their popularity and the file sizes [5, 9, 11]. A number of recent papers have characterized the workload of some WWW servers. Reference 5 shows WWW workload characteristics for six large servers. The document types accessed in the servers fall into six categories: HTML, images (e.g., gif and jpeg), sound (e.g., au and wav), video (e.g., mpeg and avi), dynamic (e.g., cgi and perl), and formatted (e.g., ps, dvi, and doc). They also show that HTML and IMAGE files account for over 90% of the total requests to the servers. It has been also shown [5,9,11] that the distribution of filesizes is heavy-tailed, i.e., the asymptotic shape of the distribution curve is hyperbolic. Reference [12] analyzes a large number of HTML documents and found that average size was 4.4 Kbytes and the maximum value of 1.6 Mbytes. In the next sections, we use the framework presented here to analyze performance of different servers.

4. EXPERIMENTAL ENVIRONMENT

In this section, we describe the environment where we carried out the Web server performance tests. Our server platform is an Intel Pentium (75 and 100 MHz) system with 16 MBytes of main memory, 256 KBytes cache memory and two 1-gigabyte disks. It has a standard 10-Megabit/second Ethernet connection card. Unix, Windows NT, Netware, and MacOS have all been used as Internet server operating systems. Each of them has some advantages and disadvantages concerning relevant features such as high-end scalability, multimedia tools, fast file systems, management tools, and security schemes. We chose Windows NT because it is becoming increasingly important in the operating system arena and because of its standard performance monitor tool, that can provide the data needed for our analysis. Performance Monitor [2] is a graphical tool for tracking computer performance, that comes with the NT operating system. It allows a user to study the behavior of objects such as processors, memory, logical disks, processes and threads. Performance Monitor provides their measurements through counters, which can be expressed as rates, such as Page Faults/sec, File Data Operations/sec or as timers, such as processor and disk utilization.

The client processes run on a SparcStation with 256 MBytes of main memory and operating system SunOS 5.4. The web server software used in the experiments are Emwac freeware HTTPS version 0.99, that originates from the European Microsoft Windows NT Academic Centre, Purveyor WebServer, version 1.2, from Process Software Corporation and WebSite from O'Reilly & Associates, version 1.1. All the servers implement the HTTP/1.0 protocol and run as a "service" in the Windows NT operating system. WebSite allows one to specify the maximum number of simultaneous connections to the server. It was set to 500, which is the maximum value.

4.1 WEBSTONE

Benchmarking has been regarded as a useful approach for analyzing and predicting performance of computer systems. Several benchmarks have been proposed for measuring hardware and software speed, including compilers and operating systems. Instead of developing a benchmark suite to represent a specific Web workload, we decided to use a standard benchmark. The workload of a WWW server consists basically of HTTP requests. WebStone is a configurable client-server benchmark for HTTP servers [6], that uses workload parameters and client processes to generate HTTP traffic that allows a server to be stressed in a number of different ways. It makes a number of HTTP 1.0 GET requests for specific pages on a web server and measures the performance of the server software and hardware platform. It is a distributed, multi-process benchmark. The master process (Webmaster), local or remotely, spawns a predefined number of client processes,. Each one of them generates HTTP traffic to the web server and collects statistics. After all clients finish running, Webmaster gathers the data collected by the clients and a generates a performance report. WebStone is designed to run for a specified period of time. The number of iterations can be also specified by the user. The client processes and the Webmaster may or not run on the same machine. The number of client processes per machine is limited only by the machine memory. In our experiments, Webmaster and client processes ran in the same Unix machine. The test time was set to 5 minutes, and the results presented are the average values of three experiments performed for each configuration and workload. The WebStone main results are throughput and latency. The former is measured in Bytes/second and the latter represents the average response time to complete a request, from the client standpoint. Other important measures are connection rate and Little's Load Factor, derived from Little's Law [1]. This factor reflects the degree of concurrency in the request execution. It is the average number of requests a server handles at a time. The Little's Load Factor is calculated by:

$$LLF = \frac{total_cumulative_time}{test_time} \quad (1)$$

where *total_cumulative_time* is the sum of the latency measured for all connections and *test_time* is the duration of the benchmark execution. Ideally, Little's Load Factor should be equal to the number of client processes. A lower value indicates that the server is overloaded and some clients are not been serviced before they time out.

The process of load generation in WebStone is performed by successively requesting pages and files from the server as fast as it can answer the requests. A new request is sent out to the server just after a client receives the answer of the preceding request. Actually, this is a problem we noted with the use of WebStone. It is difficult to mimic the behavior of real WWW users. They present cycles of idle time (user think time) followed by some Web activity, such as the network transmission and server processing. WebStone does not model user think times. The workload is defined by the number of client processes and by the configuration file, which specifies the number and the type of pages. Each page is a set of HTML files of different sizes. The type of a page is mainly determined by its size (number of files) and its access probability. A request for a page represents a request for each one of its files. The experiments were done using two different workloads (A and B), whose main characteristics are shown in table I. For workload A, the size of 94% of the accessed files is under 50 Kbytes. For workload B, the size of the files varies between 1 and 200 Kbytes, with equal access probability.

Characteristics	Workload A	Workload B
Number of files	18	180
Total size of files	298 Kbytes	5 Mbytes
Average file size	15 Kbytes	27 Kbytes

Table I: Workload Characteristics

5. RESULTS

In our experimental environment, we monitored the system behavior from two different standpoints: client and server. At the server side, the Performance Monitor tool recorded the resource usage and the behavior of threads and processes. In this work, we did not use information from the server logs. However, some server software exhibit statistics collected from the server log. Website, for instance, provides counters (average response time, throughput, etc) that are displayed by the Performance Monitor tool. At the client side, WebStone measured the latency of HTTP requests as well as the efficiency of the system. The results obtained are shown and discussed in this section.

5.1 THROUGHPUT AND CPU UTILIZATION

Figure 1 displays throughput (Kbytes/s), connection rate (HTTPOps/s), and processor utilization for workload A. Resource usage statistics were collected at two levels. At the operating system level, we obtained the total CPU, disk and memory utilization. At the lower level, we collected resource usage per process. Thus, we are able to see the amount of resources demanded by the operating system (e.g., system calls, overhead, etc) as a whole and by the server process. The processor utilization and throughput curves for the three servers are similar. We note that as we increase the number of concurrent clients, the throughput and connection rate also increase up to a certain point. The number of TCP/IP packets arriving at the server also increases, as does the number interruptions that the processor has to handle. As a consequence, the curve of processor utilization by the server process increases more slowly than the total processor utilization curve. When the total processor utilization reaches almost 100% the CPU becomes saturated and the throughput starts decreasing. Although the total processor utilization remains constant, we observe that processor utilization due to the server process decreases, because of the high number of interruptions that the server has to handle (i.e., the system overhead increases).

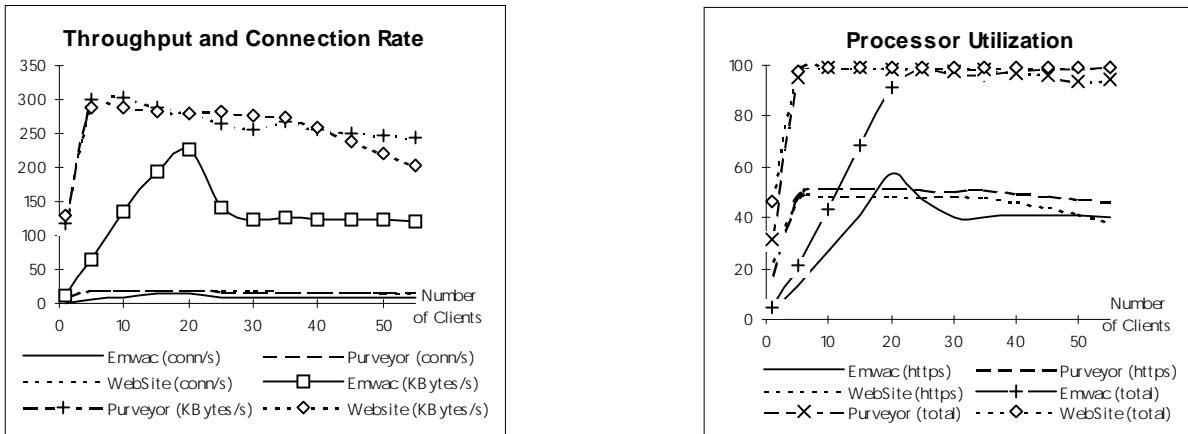


Figure 1: Throughput and Processor Utilization for workload A

Purveyor and Website reach about the same level of throughput and connection rate. Emwac has a lower throughput for its processor utilization is also lower than the other ones. The saturation point occurs at 10

clients for WebSite and Purveyor and at 20 clients for Emwac. The next section shows that latency is higher for Emwac than for the other servers. Although the network connecting clients and server was not dedicated, the experiments were done under similar conditions of network traffic. Thus, the high latency of Emwac indicates that a pending request spends more time in the Emwac server than in Purveyor and WebSite servers. The processor utilization by Emwac is lower than the utilization by Purveyor or WebSite. Our explanation is that the requests may be wasting time in the *listen()* queue of the server. The policy of starting new threads to handle pending requests and other details of implementation may be the cause for the different results observed in the experiments with the three servers. Although we do not present the results for workload B, we can say that they are similar to those obtained by workload A. As expected, the throughput for workload B is higher than the throughput of A. However, the connection rate decreases, for the server has to handle requests for larger files. The processor utilization by the server process is lower for workload B. The reason stems from the fact that the large files of workload B cause a high number of interruptions, due to the fragmentation of TCP/IP packets. We also note from the curves that CPU is the primary bottleneck in our experiments. It is the first resource to become saturated and limits the performance of the rest of the system.

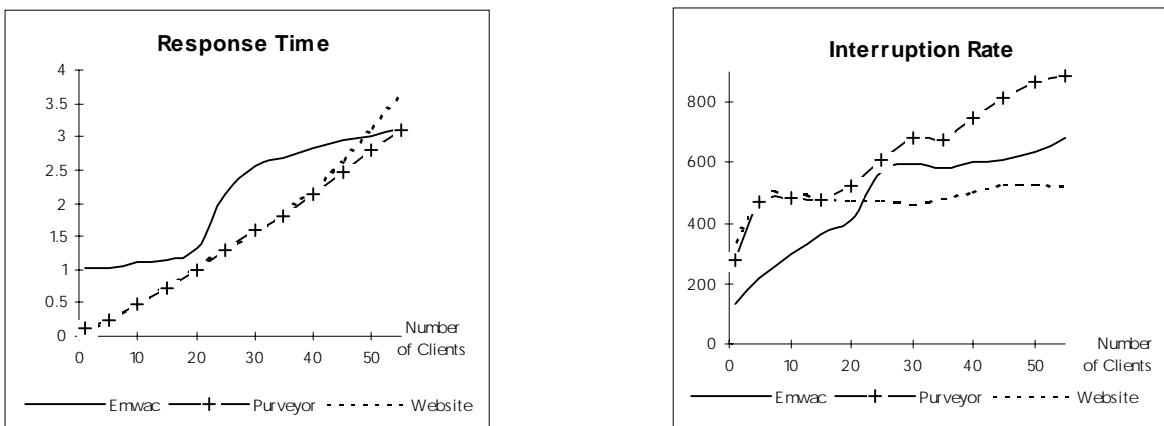


Figure 2: Response time and interruption rate for workload A

5. 2 RESPONSE TIME AND INTERRUPTION RATE

Figure 2 shows response time and interruption rate of workload A as a function of the number of clients. As expected, the response time curve increases with the number of clients as also does the number of interruptions. Response time is higher in Emwac than in Purveyor or WebSite for small numbers of clients. As the number of client increases, the error rate of Emwac increases (see next section). A higher error rate normally indicates that the server’s *listen()* queue is saturated, and “**Connection Refused**” messages are returned to clients. As the number of error messages increases, we can notice that response time increases slowly, for the *listen()* queue length decreases (due to the number of connections refused). This observation can be noted in the Emwac response time curve, that crosses the WebSite curve at 50 clients. The error rate (figure 3) of WebSite also causes a higher latency, specially when the number of concurrent clients increases. Purveyor has the lowest error rate and response time. For a small number of clients, the interruption rate of Emwac is lower than the ones of the other servers. That stems from the high latency that diminishes the total number of requests. After the saturation point, the number of errors increases as does the number of interruptions. That is due to the behavior of the WebStone benchmark. When an error occurs, the client process receives an answer faster than that expected in the normal case, when the request is accepted and treated by the server. As the client receives the answer, it sends a new request to the server, causing a new interruption. As the number of requests increases, the error rate also increases, and, as a consequence, the number of interruptions. The interruption rate of Purveyor gets higher as we increase the number of clients. That is explained by the increase in disk activity, as described

in section 5.5. Response time and interruption rate are higher for workload B than for workload A. That is due to the large average file size and the fragmentation of the TCP/IP packets.

5.3 LITTLE’S FACTOR AND ERROR RATE

Figure 3 displays the Little’s Load Factor and error rate as a function of the number of clients. For Emwac, we can notice that the Little’s Load Factor coincides with the identity curve until the saturation point. After that, the behavior of the curve changes and starts increasing slowly, suggesting that the system is spending more time in overhead activities, such as interruption and error handling. That is confirmed by the error rate curves, that exhibit a steep increase after 20 clients. As a consequence, processor utilization by the Emwac server process decreases after the saturation point, as can be seen in figure 1. The Little’s Load Factor for Purveyor and WebSite stay close to the identity curve, even for a high number of concurrent clients. The error rate for WebSite becomes significant only when the number of clients becomes greater than 40. After that point, its Little’s Load Factor starts increasing slowly but still keeps close to the identity curve. The Little’s Load Factor of Purveyor also starts increasing slowly for higher loads. That is not due to the error rate that is very low, but it is due to the higher interrupt rate. The fact that Little’s Load Factor for WebSite and Purveyor keeps close to the identity curve for higher loads indicates that the servers bear a high degree of internal concurrency in the request execution.

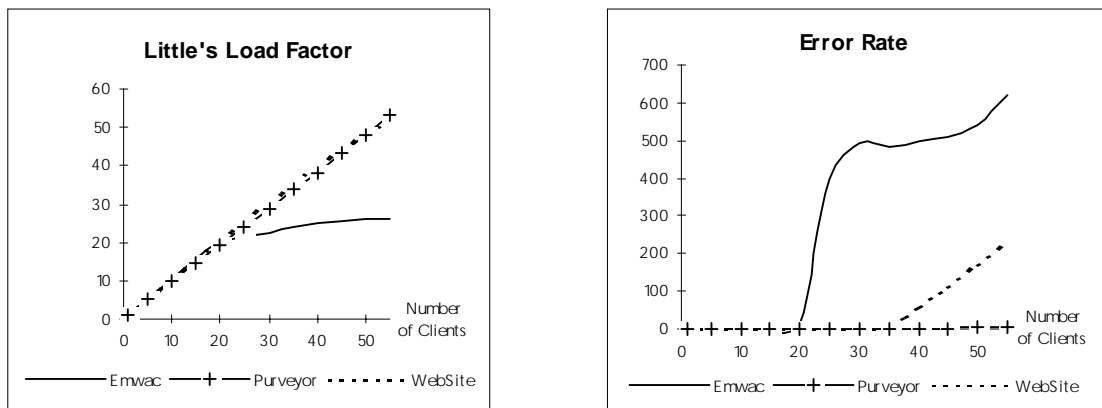


Figure 3: Little’s Load Factor and Error rate for workload A

5.4 PROCESSOR SPEED

A series of experiments were carried out using the same Emwac HTTP server in two different hardware platforms. The difference between them is the clock rate: the first one is a 75 MHz Pentium, and the second is a 100 MHz Pentium. Figure 4 shows throughput (Kbytes/s) and connection rate as a function of the number of clients for workload A. Both curves are similar in terms of shape, but the average throughput achieved by the 100 MHz-processor is 36% higher than that obtained by the slower processor. The differences in the two curves becomes apparent after 20 clients, which is the point where the CPU gets saturated, and the throughput reaches its maximum value. Those results indicate that CPU is the primary bottleneck of the WWW server executing our workload.

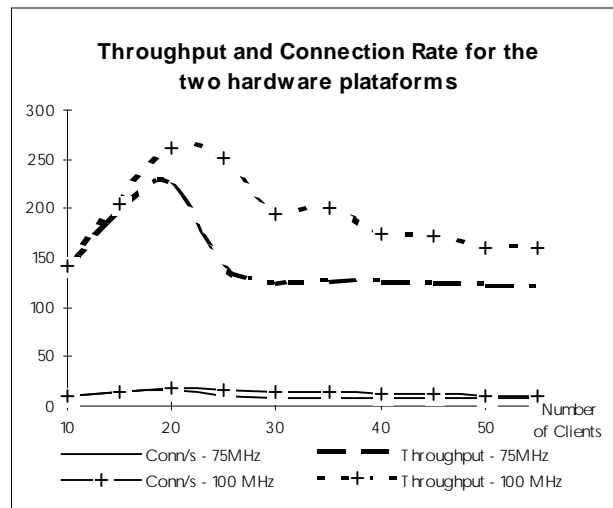


Figure 4: Throughput for two different processor speed

5.5 DISK AND MEMORY USAGE

Disk and memory usage for the two workloads are presented in figures 5 and 6. By analyzing each server separately, we notice that disk utilization and memory activity (measured in pages/sec) curves are similar. The differences for the two workloads are due to their file characteristics (Table I). The low disk utilization shown in the graphs of figure 5 is explained by the small size of the files of workload A (i.e., 18 different files with total size equal to 298 Kbytes). Due to an efficient caching mechanism, several disk accesses were avoided. The total size of the files requested by the workload B is 5 Mbytes, much larger than those of the workload A. Looking at the results for the three servers, WebSite and Emwac exhibit similar behavior. Purveyor has a much more intensive disk activity. Furthermore, we notice that Purveyor increases disk and memory activities as we increase the number of concurrent clients. For 55 concurrent clients, the disk was busy 78% of the observation time. That high utilization does not happen for WebSite or Emwac, Neither memory nor disk are bottlenecks for workload A. For workload B, Website and Purveyor have a significant disk utilization, when the number of clients becomes large. For 55 clients, the disk utilization by Purveyor, WebSite, and Emwacs are 92%, 64%, and 7%, respectively. This difference is explained by the throughput of WebSite and Purveyor that is much higher than the one achieved by Emwacs. The higher the throughput, the higher the disk and memory utilization.

Details of implementation, such as the use of internal cache, may be explanation for the differences in disk usage by the servers. If we increase the average file size, we will notice that Emwac will behave similar to the two other servers. In order to understand the way the Emwac uses disks, we carried out a set of experiments with different workload and memory configuration. In the first experiment, we increased the average file size to 300 Kbytes and used a PC with 16 Mbytes. In the second experiment, we used an average file size equal to 27 Kbytes and 8 Mbytes of RAM. For both experiments we observed high levels of disk utilization for the Emwac server. For those special workloads, the disk utilization increased with the number of clients, as we noted for the other servers with workload A and B.

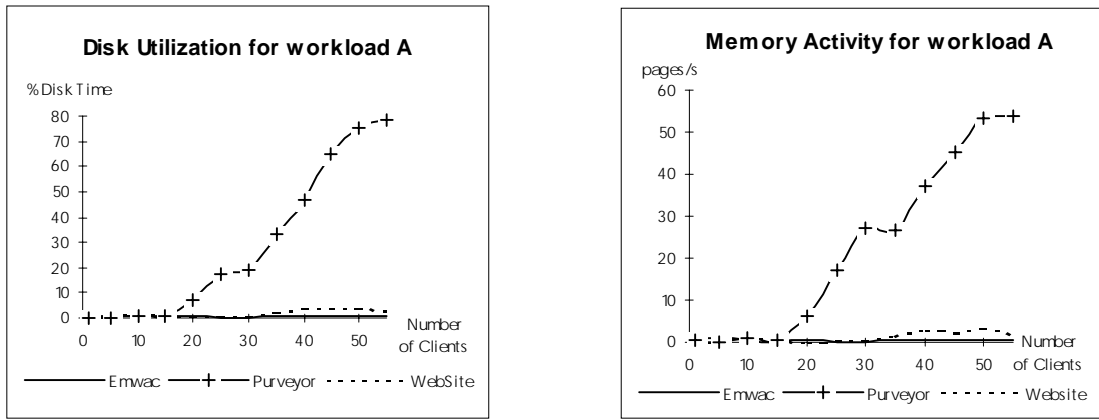


Figure 5: Disk and memory usage for workload A

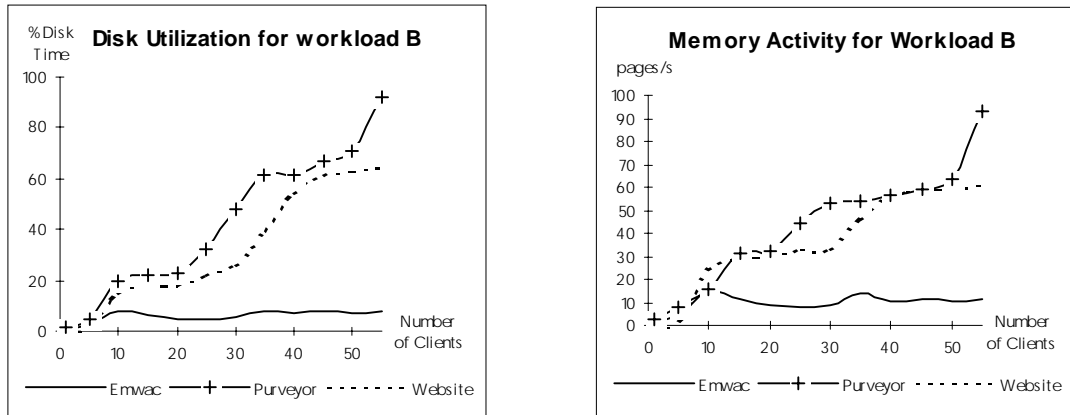


Figure 6: Disk and memory usage for workload B

6. CONCLUDING REMARKS

This paper presents a performance analysis of a WWW server. We first discussed the main steps to carry out a WWW performance analysis effort. We have examined the performance of three different servers (Emwac, Purveyor, and Website) running on top of a Windows NT platform. Using standard performance tools provided by the NT operating system and by the WebStone benchmark, we carried out a series of experiments to monitor the behavior of Web servers. No one server dominates our benchmarks. Its performance varies with the nature of the workload, i.e., file sizes, total number of files and number of clients. The Webstone benchmark is useful to generate HTTP workloads for performance comparison purpose. However, we noted that Webstone is not adequate to model the behavior of real WWW users. It does not represent user think times. Using the “black box approach”, we tried to explain the relationships between performance and characteristics of each server. A problem with that approach is that it is not able to explain all results observed in the experiments. In our benchmark, we identified CPU as the primary bottleneck to server performance in our experiments. We also observed a close relationship between error rate and throughput of the servers. As a future work, we plan to develop an analytical model to represent and predict performance of WWW servers.

REFERENCES

1. *Capacity Planning and Performance Modeling: from mainframes to client/server systems*, Daniel Menascé, Virgilio Almeida and Larry Dowdy, PTR Prentice Hall, Englewood Cliffs, 1994
2. *Optimizing Windows NT*, Russ Blake, Microsoft Press, 1993
3. World Wide Web: Whence, Whiter, What next?, Henning Schulzrinne, IEEE Network, March 1996.
4. The World Wide Web, T. Bernes-Lee, R. Cailiau, A. Luotoneu, H. Nielsen, and A. Secret, Communications of the ACM, Vol. 37, No. 8, August 1984.
5. Web Server Workload Characterization: the search for invariants, M. Arlitt and C. Williamson, Proceedings of Sigmetrics 96, ACM, May 1996.
6. WebStone: the first generation in HTTP server benchmarking, G. Trent and M. Sake, MTS Silicon Graphics, February 1995.
7. Windows NT as a Personal or intranet Server, L. Press, Communications of the ACM, Vol. 39, No. 5, May 1996
8. Performance Analysis and Modeling of a Windows NT Server, V. Almeida and G. Fialho, CMG Conference Proceedings, Nashville, December 1995.
9. Characteristics of WWW Client-Based Traces, C. Cunha, A. Bestavros, and M. Crovella, Tech. Rep., BU-CS-95-010, Boston University, July 1995.
10. Software for Reliable Networks, K. Birman and R. van Renesse, Scientific American, May 1996.
11. Self-similarity in World Wide Web traffic: Evidence and possible causes, Proceedings of the 1996 ACM SIGMETRICS, May 1996.
12. An Investigation of Documents from the World Wide Web, P. Aoki, A. Woodruf, E. Brewer, P. Gauthier, L. Rowe, Proceedings of the Fifth International Conference on WWW, May 1996.