Performance Modeling and Evaluation of Web Server Systems with Proxy Caching

Yasuyuki Fujita

Department of Informatics and Mathematical Science

Graduate School of Engineering Science

Osaka University

Osaka, Japan

January 2000

PREFACE

Performance Modeling and Evaluation
of Web Server Systems with Proxy Caching

by
Yasuyuki Fujita

Dissertation Director: Prof. Hideo Miyahara

Recently, the Internet has been expanding intensively; the number of WWW (World Wide Web) users has been growing, and the network capacity provided by network carriers and Internet service providers has been increasing for enabling high quality services. The Internet users would pay the reasonable cost for a reasonably better service. For ISP (Internet Service Provider) to provide better QoS (Quality of Services) to users, it is important to design the entire network appropriately; that is, the bottleneck of the network should be identified adequately, and a well-balanced allocation of network resources to users should be performed. However, this is not an easy task: how the amount of network resources that satisfies user's QoS expectation can be determined? For typical WWW users, a response time (i.e., time elapsed until a requested document (i.e., a WWW page) has been successfully transferred) is one of most important performance measures. However, a response time of a requested document is difficult to expect since it is affected by both the network capacity and the processing power of the requested Web server. For predicting a response time of a Web server system, a relation between incoming traffic at the Internet access link and the processing delay of the Web server should be revealed. To design a Web server system, first, it is necessary to build the model of a Web

server, furthermore, a simple mathematical model is preferred since conventional computer simulation takes a lot of time for such a complicated network system.

The first objective of this thesis is to model a single Web server without network. We perform various benchmark tests for an existing Web server, and investigate its characteristics in a quantitative manner. In our benchmarking experiments, a high–speed ATM switch is used to eliminate a possibility for the network to be the bottleneck. Benchmarking results show that the performance of the Web server can be improved by preparing the helper process for the `http` daemon. We quantitatively show a work demand, which is defined as the processing time for a given document size on a Web server. We then propose the performance model of a Web server, which consists of a FIFO (First-In First-Out) queue as a dispatcher and a PS (Processor Sharing) queue as a single processor used by all helper processes. Accordingly, the performance model of a Web server is modeled as an M/G/1/PS queue with a limited number of jobs allowed in the server.

Using our model of the Web server, we then propose and examine the performance engineering problem of the Web server. Several numerical examples of our model show that the mean response time of small size documents becomes very large when the offered traffic load is high. It is also shown that the performance improvement by increasing the number of helper processes is minor for the a typical document size distribution. In a future Web server system handling various multimedia contents, it is expected that the average size of documents is quite large. Therefore, the number of helper processes is an important factor in the future Internet.

One of key technologies in a recent Web server system is a Proxy server and its caching mechanism, which is called as *Proxy caching*. By using the Proxy caching, the response time of a requested document becomes very small, if the requested document has already cached at the Proxy server. This mechanism dramatically improves the response time for documents located at a distant Web server. However, the Proxy caching requires an additional overhead to retrieve and save the requested document if it has not been cached. In this thesis, the performance characteristics of a Proxy server are investigated by performing two benchmark experiments for modeling and evaluating the Proxy server with caching. We quantitatively show a work demand in two cases: when a requested document has been cached, and when it has not. We then show that a PS queue is adequate to model the Proxy server.

The second objective of this thesis is to apply our analytic model to design a real Web server system. We focus on two Web server systems: (1) a public Web server in the Internet, and (2) a Proxy server provided by ISP. We demonstrate applicabilities of our Web server system modeling approach to evaluate performance of these two Web server systems. In the first case, the Web server is open and publicly accessible from the Internet users. In this case, the response time of a document retrieval consists of two factors: a delay experienced at the Web server, a transmission delay over the access line and the Internet backbone. Our simulation results derived from this model are mainly affected by the performance of the Web server and the bandwidth of the access line connected to the Internet. We show that the increased capacity of the access line improves the transmission delay particularly when the request arrival rate is small. And we also show the improvement of the performance of the Web server is important to improve the response time, but dramatic improvements cannot be achieved if the processing power of the Web server becomes large since the delay within the Internet backbone becomes dominant in our model. Hence, an improvement of the Internet backbone is of necessity for realizing very high–performance Web server system once both the access link and the Web server are adequately prepared.

Using the second case, we demonstrate the applicability of our approach to investigate the effect of the Proxy caching. When the cache hit ratio exceeds about 50%, the transmission delay within the access line is not so improved. We therefore point out that a more complicated and slightly improved caching algorithm does not help improving the transmission delays within the access line. In most cases, the transmission delay within the Internet is much larger than the processing delay at the Web server and the transmission delay in the access line. One would expect that the faster Internet backbone improves the total response time, but it does not always lead to the dramatic improvement. Using our analytic model, we can show the reason that the performance bottleneck moves from the Internet backbone to other location – the access line.

Finally, we investigate the applicability of our analytical method to solve a queueing network model by comparing with our simulation experiments. We use a MVA (Mean Value Analysis) method and model a Web server as an IS (Infinite Server) queue where a work demand at the IS queue is obtained from our previous analysis of the Web server. Numerical examples show a good agreement of the analytic method with simulation when either the Internet backbone or the access line is the bottleneck. Furthermore, when the Web server becomes the bottleneck, an analytic result and a simulation result show a reasonable agreement.

Through all of our research, we have discussed with the engineering problems of the resource allocation on the Web server system. Our modeling approach makes it possible to construct a high-performance Web server system by evaluating the performance characteristics quantitatively. It is also possible to design the system with considering not only a well-balanced allocation of the resources within the Web server system but also production costs. That is, our approach can identify the performance bottleneck of the Web server system and can be used for its performance planning.

Acknowledgements

I would like to express my sincere appreciation to continued support of my advisers, Professor Hideo Miyahara, through the trials and tribulations of this Ph.D. thesis. I would like to also thank his encouragement and invaluable comments in preparing this thesis. And I am deeply grateful to Professor Nobuki Tokura and Professor Katsuro Inoue for serving on my thesis committee and also for their kind advice and invaluable comments that helped me in improving this thesis.

I am most grateful to Professor Masayuki Murata for his enthusiasm in teaching and pursuing performance modeling of Web Server Systems with me. He has been actual adviser and been opening my eyes toward the Web System. His active interest and encouragement have been of great help in furthering my efforts in this area, and his standards of excellence will stay with me throughout my career.

My thanks go to members of Advanced Network Architecture group of Multimedia Information System Laboratory in the Department of Informatics and Mathematical Science for their valuable advice, continuing encouragement and friendship.

I would like to express my appreciation to The Kansai Electric Power Co., Inc. for giving me an opportunity to study in Osaka University. I am heartily thankful for the fellowship and underpinning — special thanks to Mr. Tadao Tanabe, Director of Kansai Electric, and Mr. Yoshitaka Kusano, General Manager of Information System Center, for their expertise as well as warm-hearted suggestions.

I am deeply grateful to my parents. They always give me endless love and assist me. Finally, I dedicate this thesis to my lovely wife Yukiko for her endless support, encouragement and understanding.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

In recent years, the Internet has experienced phenomenal growth. The number of hosts connected to the Internet was 29,670,000 in the world in January 1998, and it is increased up to 43,230,000 in January 1999, and 56,218,000 in July 1999 [1]. It is especially true when we expect the Internet with high quality since various kinds of media including text, image, audio, and motion video are being offered through the WWW (World Wide Web) ; the realized technology on the Internet. To generate the best possible performance for WWW services to the Internet users, an understanding of the relation among system resources such as the Internet backbone, the Internet access line and the Web server are of necessity.

Accordingly, studies on traffic characteristics of the Internet become important to deeply understand the traffic behavior of the Internet and to build the efficient Internet. The traffic characteristics of the Internet have already been investigated by several researchers; see, e.g., [2] and references therein. In [2], it was observed that document sizes and request interarrival times follow log-normal distributions, and the request interarrival times during the busiest two hours follow an exponential distribution. Based on the authors proposed, an M/G/1/PS model was applied for dimensioning the Internet access line. Furthermore, the effects of the traffic characteristics, affected by the document caching in the access line, have been studied, for example in [3]. However, their study does not consider the performance of the Web server. In actual, of course, the quality of service offered to users is affected by the request processing delay at the Web server as well as the transfer delay within the network. If the Web server simultaneously processes many requests, it is likely that the Web server easily becomes a bottleneck. That is, the quality of service to users is affected not only by the network bandwidth but also by the Web server processing speed of requests. So, the improving of the Web server performance is a critical issue for the Web sites which service a high volume of requests.

Studies concerning on the Web server performance are very few. In [4], the authors focused on a method of improving the Web server performance in the situation that the CPU processing power is a limiting resource. In [5], the purpose of the authors was to build a new Web server mechanism, and they offered several performance results showing that their proposed server is efficient and scalable. In these studies, the authors have investigated to improve the performance of the Web sever itself. On the other hand, several papers have examined performance characteristics of the Web server by benchmarking [6, 7] to compare the Web server hardware, and to analyze the influence on response times by different document sizes. And in these studies, the authors are also examined a way of starting http daemon. We also measure the performance of the Web server by utilizing the benchmarking tool, and a part of obtained results has already been known in the above literature. The study on the Web server modeling is presented in [8]. In that paper, the authors model the Web server and the Internet as an open queueing network model. However, validation of the modeling approach is not shown. Furthermore, numerical parameters used in performance evaluation are fictitious and not based on the actual systems.

However, the above studies focus on the characteristics of the Web server performance, especially. So, they do not consider the entire Web system. A recent Web system often uses the Proxy caching, which can be used to improve the document response times. The caching mechanism in the Proxy server returns the requested document if it has been cached by the Proxy server. On the other hand, when the requested document has not been cached, the Proxy server forwards the document request to the original Web server and retrieves the document, and forwards the document to the user. Concerning the Proxy server, there are a lot of studies on the caching algorithms and a distributed Proxy server based on the caching technologies [9-11]. On the contrary, our study is to investigate the performance characteristics of the Proxy server to be included in the performance model of the entire Web server system.

1.2 Benchmark tool

Recently, some benchmark tools are available such as WebStone [12], SPECweb96 [13] and WebBench [14]. Those benchmark tools run independently of the server platform or server software running on it, and the clients prepared by those tools generate requests to the server to examine the server's behavior and performance (e.g., the response time, server throughput and so on). The main purpose of benchmarking compares the performances of different Web server platforms using the same Web server software or the performances of different Web server softwares on the same platform. While benchmark results have already been publicly available on the Internet, those results are based on different experimental environments of benchmark softwares and server platforms, and we can not directly apply them as parameters to model the Web server. In our experiment, we used WebStone [12] as the benchmarking tool that is used the cases of the Web server and the Proxy server experiments.

In our experiments, we used WebStone since we can specify the distribution of requested document sizes, the frequency of access to the Web server, and the number of clients generating loads for the Web server [15]. Figure 1.1 illustrates the Web-

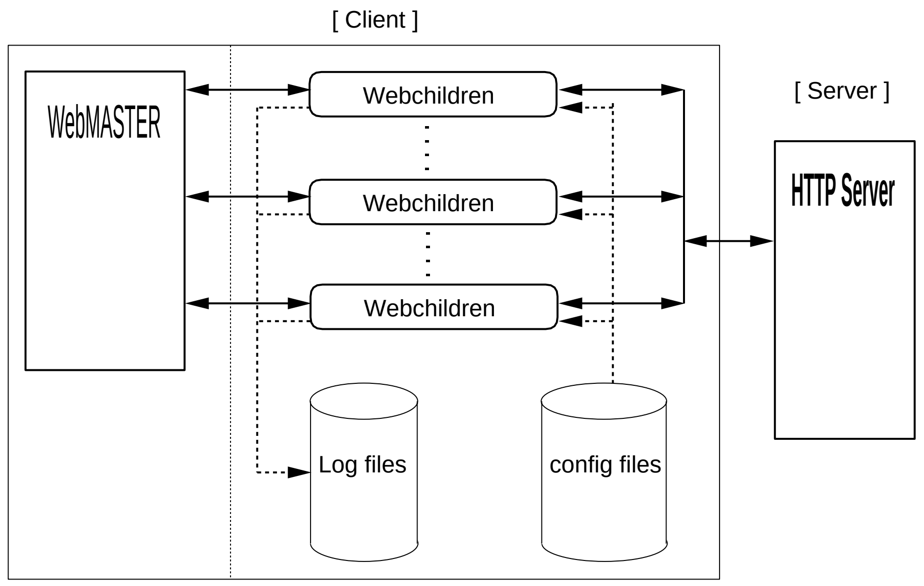

Figure 1.1: WebStone structure.

Stone structure. The “Webchilderen” are controlled by the “WebMASTER”, which remotely spawns the multiple Webclidrens on one or more client machines. Each Webchildren issues the request one after another to the Web server. More precisely, after the Webchildren establishes the HTTP connection with the Web server, it sends one request at a time. It immediately closes the connection after it receives the document. Then, the new connection is established to obtain the next document. Namely, it employs the HTTP version 1.0 [16]. In our study, it is not a main problem whether the version of HTTP is 1.0 or 1.1 because we assume the general environment such that the Web server accepts the document retrieval requests from anonymous users.

The following run rules are prescribed in using WebStone [15].

- File set: When the benchmark results are published, the files mix specifying ratios of the documents should follow “filelist.standard” included in the WebStone distribution. The content of `filelist.standard` is shown below. Each line consists of the name of the document and the percentage that it is referred. Furthermore, the file name indicates its file size (i.e., Web document

size). For example, the first line, “/file500.html”, means that the document with 500 bytes is accessed with probability of 350/1000. In [15], it is described that the `filel-
ist.standard` shows the representative file set for the real Web server and the frequency of the accesses.

```
/file500.html 350
/file5k.html 500
/file50k.html 140
/file500k.html 9
/file5m.html 1
```

- Benchmark Run Configuration: The runtime must be at least 10 minutes. It provides adequate time for the server and client configuration to reach a steady state, and it is long enough to cancel out high variations observed in the first few minutes of the run.

- The Number of Clients: The number of clients should vary from 20 to 100 in increments of 10 so that performance of the server under a wide variety of loads can be observed.

- Server Machine Configuration: It is necessary that the user should report the operating system, memory configuration and any special operating system modifications, and especially changes to the TCP/IP stack.

1.3 Outline of Dissertation

In this thesis, we investigate basic performance characteristics of the Web server system, which enable us to identify the system bottleneck and find the well–balanced allocation of the resources including the Internet backbone, the access line, the Web server and the Proxy server. In the rest of this Section, we summarize the objectives of this thesis.

1.3.1 Analysis and Modeling of Web Server Performance

[17-23]

In Chapter 2, we conduct several benchmark experiments to characterize the Web server performance. We used the ATM switch to interconnect the client and server to neglect the transmission delay within the network. And we used Web-Stone as a benchmark tool. In our experiments, we consider (1) the way to prepare the HTTP daemon, (2) the number of the helper processes, (3) the relation between document sizes and response times, (4) the document size distribution, and (5) network capacity. In modeling the Web server, we selected the effective and highly efficient way to prepare the HTTP daemon from our experimental results. We next investigate the relation between the document size and the required processing time on the Web server, and show how the Web server performance is affected by the document size and the document distribution. As a result, the quantitative parameters that are essential in modeling the Web server as a numerical parameter, can be represent. The parameters can be expressed as the work demand for given WWW document size determined. The work demand is obtained from the response time divided by the number of clients accessing at the same time. Hence, it is independent of the number of clients. Therefore, a processor sharing schedule discipline may be adequate to model the Web server. It would be applied when our concern is to obtain the response time averaged over all document sizes. However, we would like to find the response time for each document size. So, we next have experience to evaluate the delay at the dispatcher process. And we should consider the delay at the dispatcher. From our discussions, we propose the a queueing model of the Web server and confirm the its scheduling discipline.

Next, we examine the model of the Web server in detail. Using our approximate analytical method, we give how the Web server performance can be improved. And through numerical examples, we discuss the performance engineering problem of the Web server. We observe that the processing time on the Web server is influenced by the traffic load. In the future Web service, the WWW documents with large size documents are likely increased. Those include the motion video and audio data. So, we can propose how to keep the quality of service in terms of document retrieval time and give simulation examples with our analytical results.

1.3.2 Analysis and Modeling of Proxy Server Performance

[24-26]

Next, we investigate quantitative performance characteristics of the Proxy server by the benchmark tests. We show the experimental results for modeling the Proxy server in Chapter 3. In our experiments, we set the document hit ratio as an input parameter. We used the ATM switch to connect the client, the Web server, and the Proxy server. We take the following approach to investigate the performance of the Proxy server and use the WebStone as the benchmark tool. In our experiment, the clients of WebStone issue the requests to the Proxy server, and the Proxy server returns the document if it has the document. If the Proxy server has not kept the document, it forwards the request to Web server and returns the document to the user. Thus, the experimental results reported by the WebStone can be used to investigate the performance characteristics of the Proxy server by controlling the document hit ratio.

In this Chapter, we present two experiments, (1) the number of clients to access the Proxy server at the same time, and (2) the hit ratio of the cache. We investigate the work demands of cached and no–cached documents. The quantitative parameters that are essential in modeling the Web server as a numerical parameter, can be represent. The parameters can be expressed as the work demand for given WWW document size determined. The work demand is obtained from the response time divided by the number of clients accessing at the same time. Hence, it is independent of the number of clients. On the other hand, we focus the relations between the document size and the response time on the Proxy server. Its result shows that the tendencies of the response time of the cached document are changed over a certain document size. So, we produce the coefficient values for the cache hit ratio and the WWW document size. As described above, we can propose the queueing model of the Proxy server with caching and its scheduling discipline.

1.3.3 Performance Modeling and Examples of Web Server Systems

[24, 21, 22, 25, 26]

From discussions above, by modeling the Web server and the Proxy server, we are able to build the entire Web server system model including the Internet backbone, the Internet access line, the Web server, and the Proxy server. Hence, we propose the performance evaluation model which can examine throughput and response time, as a performance index, of the document transmission request from users.

In Chapter 4, a modeling approach of the Web server and the Proxy server models, (1) the Web site is publicly open to the Internet, and (2) users within a certain local network access the Internet via Proxy server. In our simulation, we also evaluate the performance of the Web server system in order to demonstrate the applicability of our approach. When we expect the faster response time, we cannot find out the good solution easily. Because it is indistinctness what is the bottleneck in the Web system. So, we will indicate quantitative results by demonstrating applicabilities of our Web system modeling approach to evaluate a performance of the Web system. Therefore, we will show the relations of the resources, in this thesis, the Web server, the Internet access line and the Internet backbone. Furthermore, we consider the effect of the Proxy caching, in other word, we need to take account of the effect of decreasing the network traffic load by the document caching on the Proxy server. Using such the performance evaluation model, we except the extreme bottleneck and are able to build the well–balanced systematic network. In this thesis, we show the examples of the performance evaluation based on our proposed model and discuss its meaning.

Finally, Chapter 5 includes concluding remarks and future research topics.

Chapter 2

Analysis and Modeling of Web Server Performance

2.1 Experimental Configuration of the Web Server

Our experimental system is illustrated in Figure 2.1. We used the ATM switch to interconnect the client and server to avoid the network being bottleneck. It was necessary to test the maximum throughput of the Web server. Noting that we used the ATM switch, we found that the overhead due to the network delay is negligible when compared with the time elapsed at the server. Therefore we will not discuss the network delay in this thesis except Subsection 2.2.5 where we will show the results using Ethernet.

The hardware and software used in our experiments are summarized in Table 2.1. We used Apache (version 1.2.4) [27] for the HTTP server. Recently we have many HTTP servers, which include Internet Information Server by Microsoft Co. [28], Enterprise Server by Netscape Communications Co. [29], NCSA [30], and CERN [31]. Among them, we have adopted Apache since it has about 55 percents market-share of the HTTP server [32]. A server configuration of our experiment is as follows; the logging option is set as default. Document files are stored in the

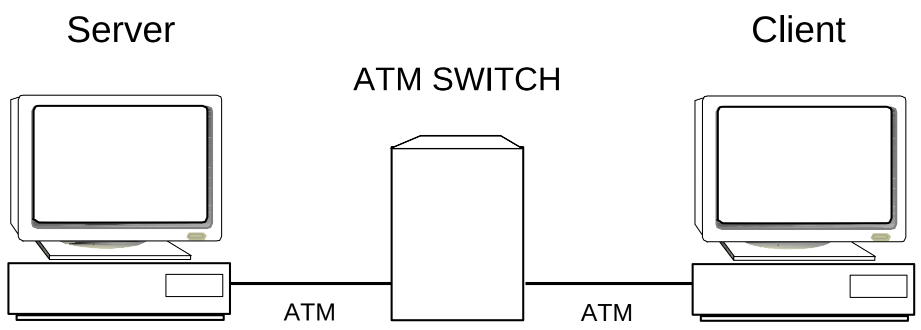

Figure 2.1: Experimental system configuration of the Web server.

local disks of the Web Server. On both of the server and client machines, we did not modify the operating system for tuning TCP/IP stack. That is, we set up the experimental system same to the commonly used Web servers.

Table 2.1: Experimental system of the Web server.

	Server	Client
Machine	Indiogo2(SGI)	Indigo2(SGI)
CPU	IP26 75 MHZ	IP22 250 MHZ
Main Memory	320 Mbytes	192 Mbytes
Web Server	Apache 1.2.4	—
Benchmark	—	WebStone 2.0.1

In our experiments, we used WebStone (Ver. 2.0.1) as the benchmark tool. We did not follow the run rules of WebStone in some experiments since we aim at collecting the quantitative data for Web server modeling. In such cases, we will explicitly present the configuration. In each simulation setting, we tested five experiments, each of which runs ten minutes, to obtain the reliable results. Then, the average of those five experiments is presented in the below. Last, we note that in our experiments, we only considered the document transfers, and did not include requests that require the processing at the Web server such as cgi.

2.2 Experimental Results of the Web Server

In this section, we present the results of our experiments as follows;

- Experiment 1: The way to prepare the HTTP daemon on the Web server performance (Subsection 2.2.1)
- Experiment 2: The effect of the number of helper processes on the Web server performance (Subsection 2.2.2)
- Experiment 3: The relation between document sizes and response times (Subsection 2.2.3)
- Experiment 4: The effect of the document size distribution on the Web server performance (Subsection 2.2.4)
- Experiment 5: The effect of network capacity (Subsection 2.2.5)

2.2.1 Experiment 1: The way to prepare the HTTP daemon

In this subsection, we evaluate the effect of the way to prepare the HTTP daemon on the Web server performance. The following five methods can be considered in preparing the HTTP daemon [7].

(1) A daemon process listens to the port of httpd. When the connection request arrives at the server, the daemon forks a new copy of itself, and the child process handles the request. (In what follows, we will refer it to as the *fork* case.)

(2) An `inetd` program waits at the inetd port. When the connection request arrives, `inetd` issues fork() and exec() for the HTTP daemon against each request. (*inetd* case)

(3) A single daemon starts a separate thread for each request. (*multithread* case)

(4) A dispatcher process receives each request and passes it to one of helper processes, which are in the preallocated pool of processes and are created once at startup time. Then, the selected helper process accepts the request. If the helper processes are not available, the request is enqueued. (*helper* case)

(5) A dispatcher process receives each request and passes it to one of the helper processes. The difference from the above (4) is that, if all of helper processes are handling requests, the dispatcher creates a new process by executing fork(). (*helper+fork* case)

In our experiments, we did not use a multithreaded machine for the Web server. Further, it is difficult and complicated to implement a multithreaded server, and the multithreaded server only works on the limited system. Therefore we did not consider the case of *multithread*.

Hereafter, we compare the results obtained by *fork*, *inetd*, *helper*, *helper+fork* as the way of starting the HTTP daemon process. The number of clients is changed from 10 to 110. A standard document set `filelist.standard` is used for the document distribution. In the case of *helper* experiments, the number of helper processes was fixed at 16. The results dependent on the number of clients are compared in Figures 2.2 and 2.3 where the average response times and server connection rates are shown, respectively. Here, the response time means the time from when the request is created at the client until the corresponding document is received by the client, and the server connection rate does the number of successfully processed connections (i.e., documents in the current case) per second.

When the inetd process creates the HTTP daemon, it is necessary at the server side to read all of configuration files and to set up options of the HTTP daemon. Those are reasons that its performance is lowest. On the other hand, the case of *fork* does not need to read configuration files, and therefore the HTTP daemon is started much faster than *inetd*. It results in that the *fork* case outperforms the *inetd* case.

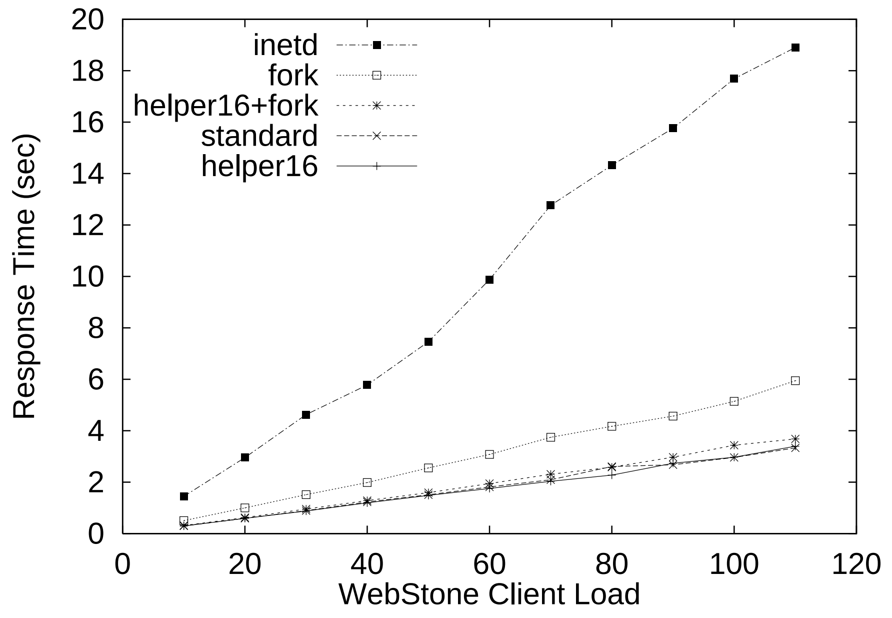

Figure 2.2: Effects of the way to prepare the HTTP daemon on the average response time.

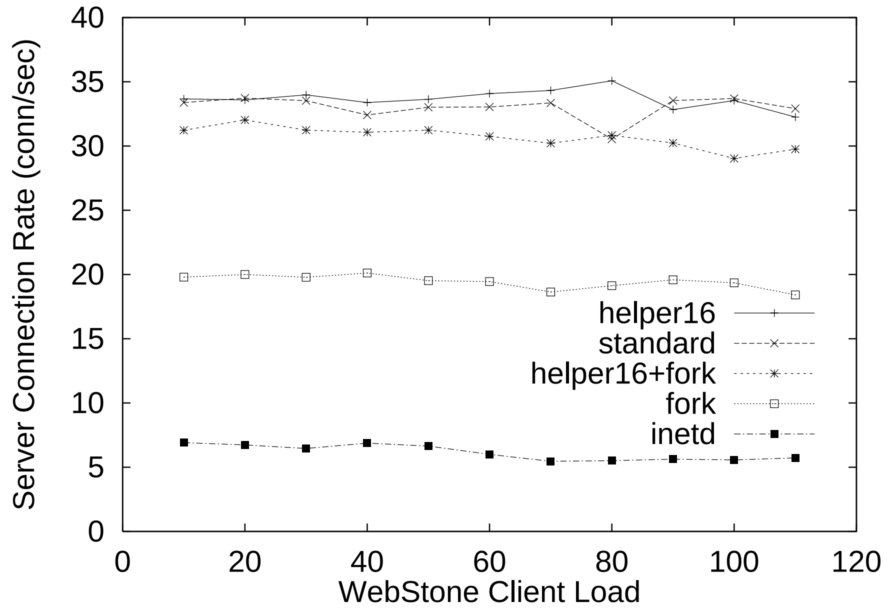

Figure 2.3: Effects of the way to prepare the HTTP daemon on the server connection rate.

When compared with the other *helper* and *helper+fork* cases, however, the overhead to create new process is a still burden as can be seen in the figures.

When the *helper* processes are utilized, requests are handled in a similar way to the *fork* case, but we can expect that the time of passing the connection request from the dispatcher to the helper process is much faster than the time of creating process by fork(). As can be observed in the figures, the effect of *helper* process is clear. Of course, in the *helper* case, it consumes the system resources such as memory to prepare the helper processes. It may lead to the performance degradation if we prepare many processes. We therefore need to seek the appropriate number of helper processes, which will be evaluated in the next subsection.

In the case of *helper+fork,* we observed worse performance than *helper* case. Its reason seems to be that the overhead to create the new process by fork() call is not negligible, and we will present only the case of *helper* in the experiments below. However, we should note here that since such an overhead depends on the performance of the Web server platform, different results may be obtained if we use other machines.

In the figures, we also present the result labeled as “standard”, which is obtained by using the default setting of Apache. It corresponds to “helper5+fork”, that is, the number of helper processes is initially set to be five and the new process is created by the fork() call if the helper processes are not available. From the figures, we can observe that it is not always true that the default setting gives best performance.

From a performance modeling point of view, we can find the following observations. In all cases, the response time increases almost linearly in proportion to the number of clients. On the other hand, the server connection rate is not changed regardless of the number of clients. Accordingly, we can confirm that the Web server system is a work conserving system. Further, we do not need to consider the server connection rate (server throughput) dependent on the number of clients.

2.2.2 Experiment 2: The effect of the number of helper processes

The appropriate number of helper processes must depend on the number of clients that concurrently access the Web server. If more helper processes are prepared, the Web server can handle more requests simultaneously. However, the increased number of helper processes may not help the performance improvement when the Web server platform only has a single CPU. Further, helper processes waste machine resources, and it may lead to the performance degradation. We therefore examine the effect of the number of helper processes. In this experiment, we used `filelist.standard` as the document set.

Figures 2.4 and 2.5 show the average response times and the server connection rates dependent on the number of clients. In the figures, four cases with respect to the number of helper processes (8, 16, 32, and 48) are presented. From Figure 2.4, we can observe that the average response times grow almost linearly. The server connection rates are almost the same in the cases of helper8, helper16 and helper32 (the number of helper processes is 8, 16 and 32). However, the case of helper48 shows the worse performance because of an exhaustive use of system resources.

Between two cases of helper16 and helper32, there are not very significant differences. Therefore, one may think that the appropriate number of helper processes is 16 to obtain the best performance and to limit the use of system resources at the same time. However, in the current experimental setting, we have used the `filelist.standard` offered by WebStone. The document size distribution defined in `filelist.standard` may or may not be adequate for the actual situation. Furthermore, a more important point from a viewpoint of performance modeling is that a smaller number of helper processes increases the queueing delay at the dispatcher when the number of clients increases. Even if the average response times are same in two cases; helper16 and helper32. However, the response time consists of two parts; the queueing delay at the dispatcher and the processing delay by the

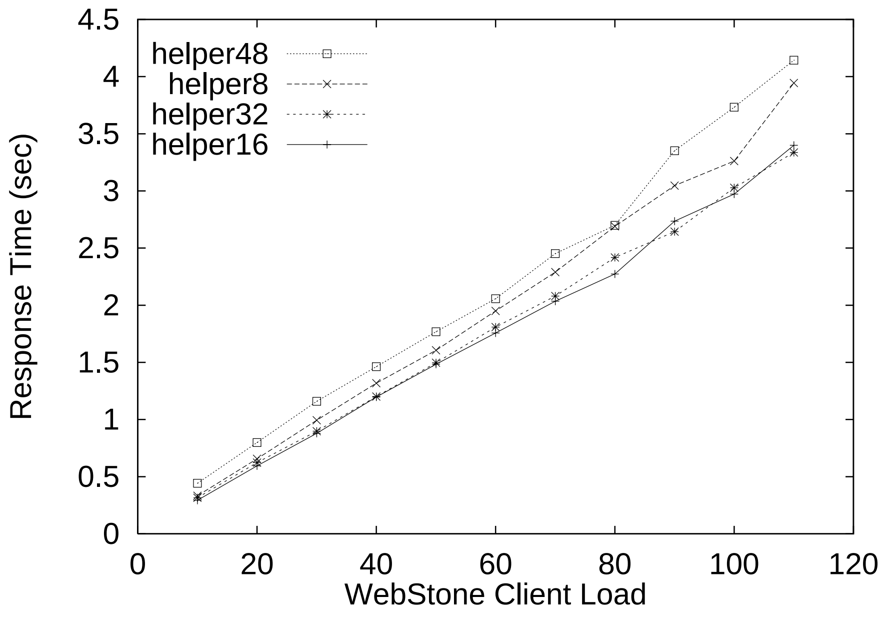

Figure 2.4: Effects of the number of helper processes on the average response time.

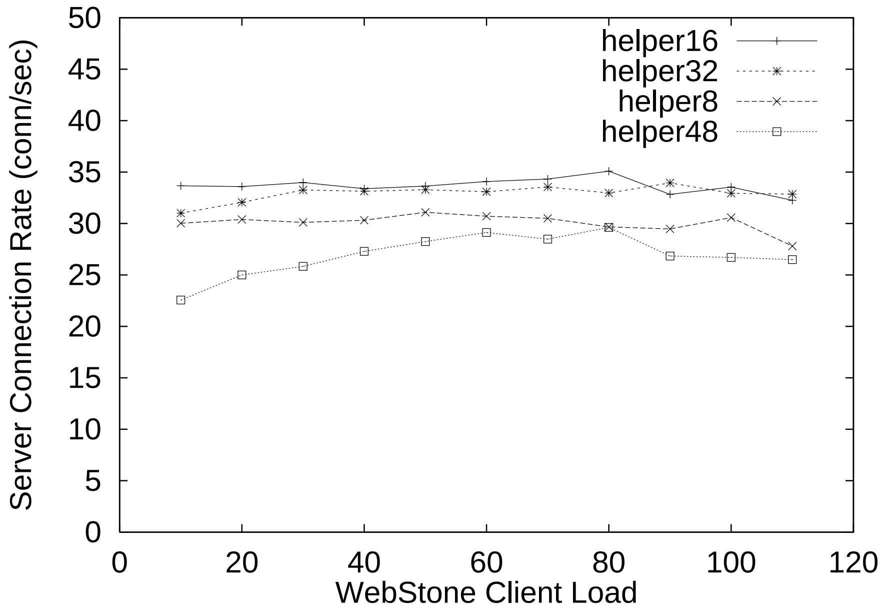

Figure 2.5: Effects of the number of helper processes on the server connection rate.

helper process. Suppose that the large–sized document requires much larger processing time than the small–sized document (this is true as will be shown in the next subsection). Since the dispatcher handles the requests in a FIFO manner, the large document arriving first may increase the delay of the small document arriving next. It is likely to happen when the number of helper processes is small. To examine these effects, we next investigate the relation between the document size and its response time in Subsection 2.2.3. Then, we consider the case of requesting different sizes of documents in Subsection 2.2.4.

2.2.3 Experiment 3: The relation between document sizes and response times

The Web site has many kinds of documents such as texts, images, and motion videos. Further, document sizes are widely varied dependent on contents. For example, the authors in [2] showed that the document size follows the log-normal distribution. Furthermore, the distribution of the document size stored in the Web server might change in the future due to a recent advancement of the multimedia application. Accordingly, we want to know the required processing time of the request with a given document size. In this subsection, we investigate the relation between the document size and the required processing time on the Web server, and show how the Web server performance is affected by the document size.

For the above purposes, we did not use the `filelist.standard` offered by WebStone, instead we set the document size to be fixed in each experiment. For other parameter settings, the number of helper processes is 16 and the number of clients are changed from 4 to 64.

Figures 2.6 through 2.8 depict the average response times, the server connection rates. We also display the throughput in Mbps, which is obtained by multiplying the connection rate [document / sec] by the document size [bit / document]. The hor-

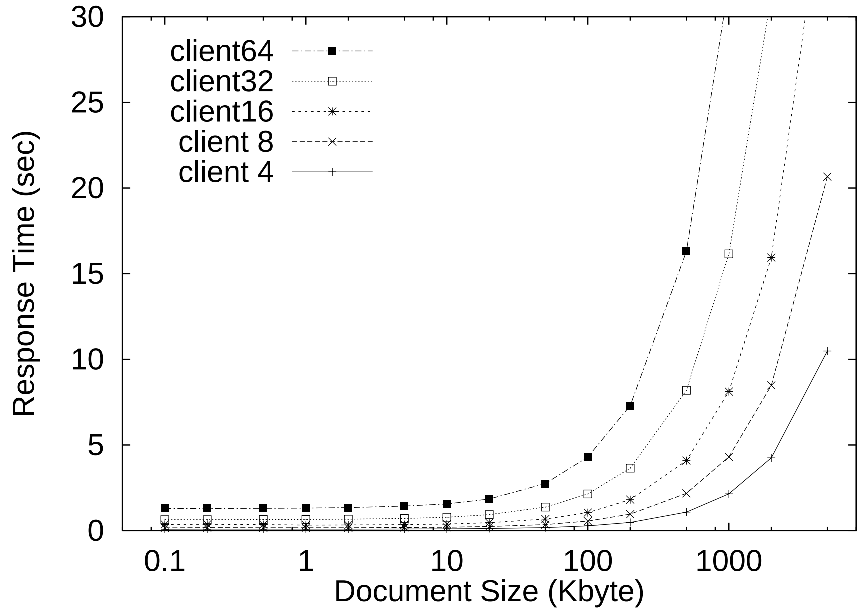

Figure 2.6: Effects of the document size on the average response time.

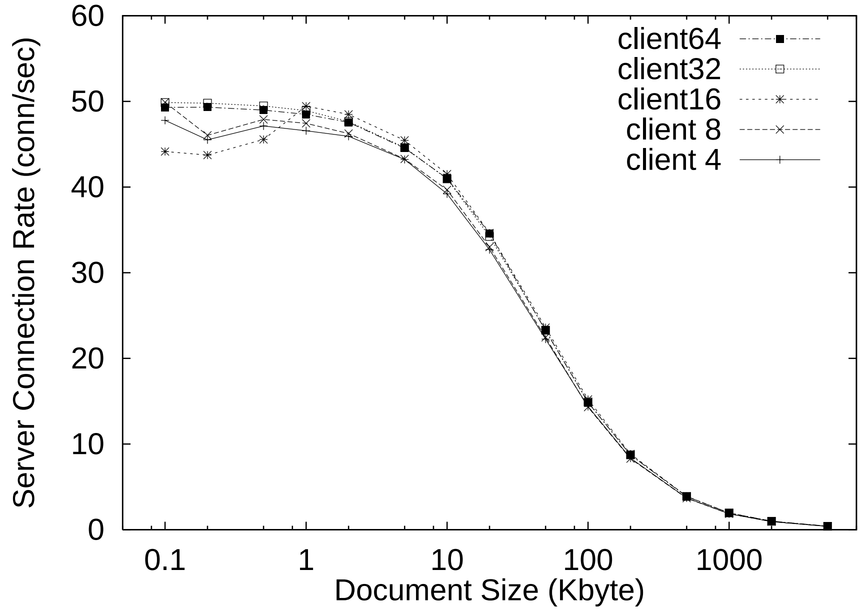

Figure 2.7: Effects of the document size on the server connection rate.

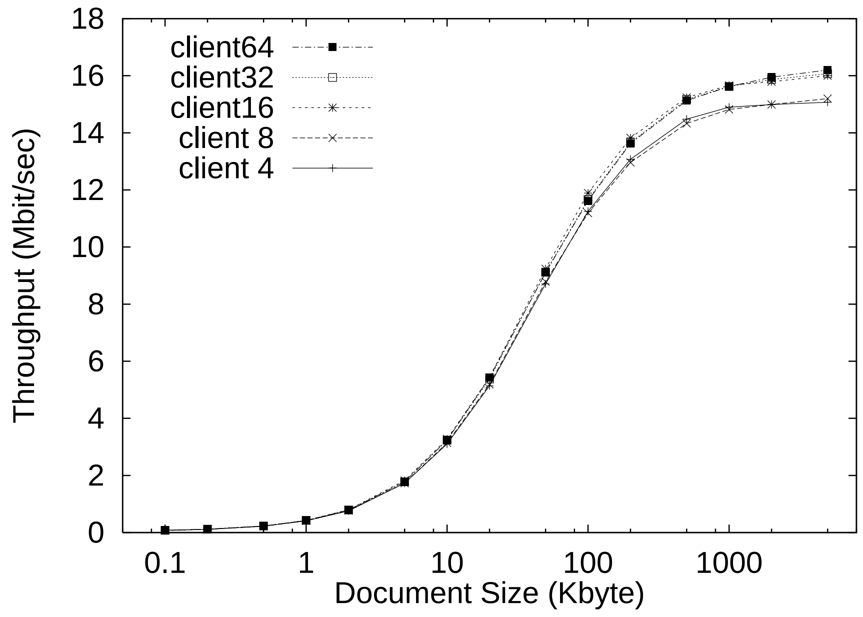

Figure 2.8: Effects of the document size on the server throughput.

izontal axis shows the document size. From the figures, we can see that the response times are almost constant when the document sizes are small. It is due to the processing overhead. Then, as the document size becomes large, the response time and throughput gradually increase from around the 10 Kbyte document size. Finally, the response times increase linearly. To present this fact more clearly, we depict the relation between the response time and the document size in Figure 2.9 by setting the vertical axis in log–scale.

From Figure 2.9, it is apparently inadequate to assume that the response time is in proportion to the document size. However, if the number of helper processes is fixed, the number of clients does not affect the server connection rate (and total system throughput). Further, the response time is almost proportional to the number of clients for a given document size. It can be conjectured from Figure 2.9, but to see it more clearly, we depict Figure 2.10 where work demands against document sizes are shown. Here, the work demand is obtained from the response time divided by

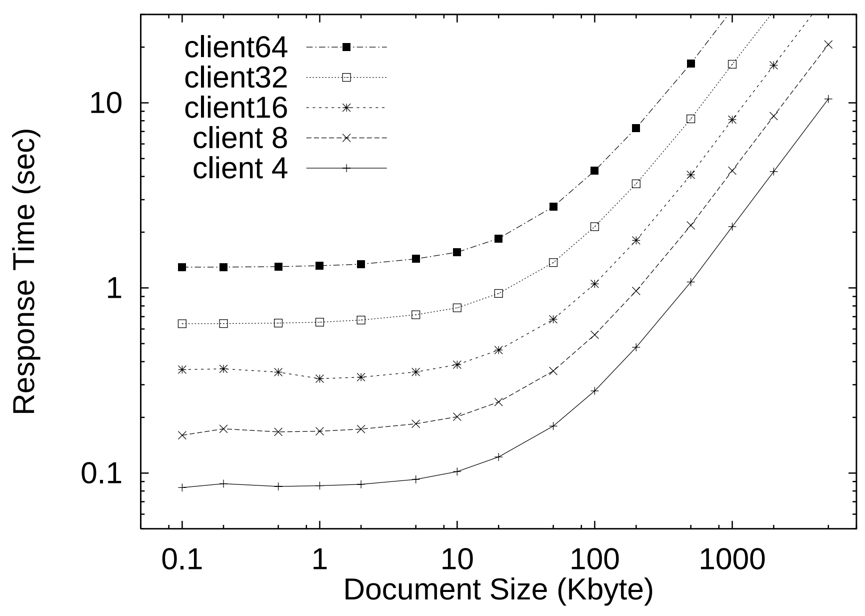

Figure 2.9: Effects of the document size on the average response time.

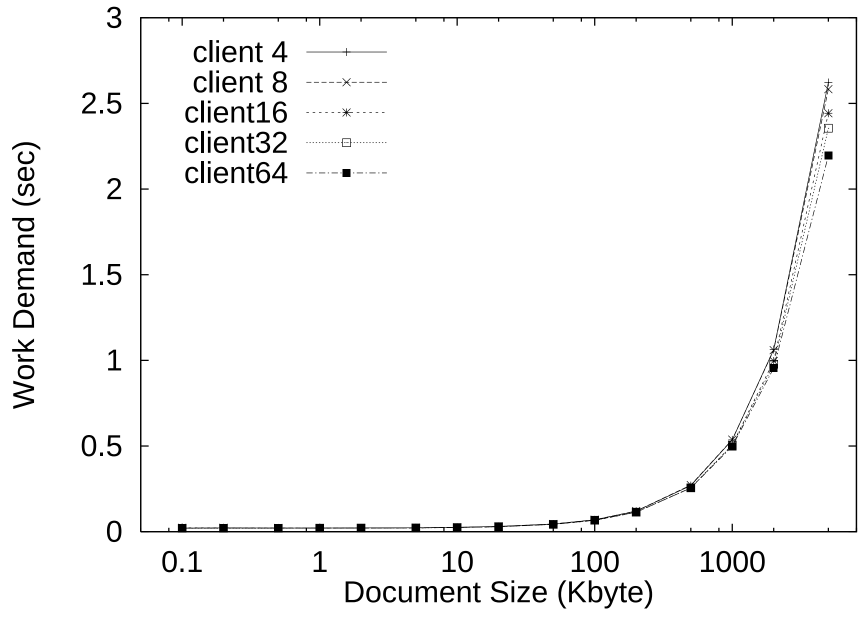

Figure 2.10: Work demand against the document size.

the number of clients using the results shown in Figure 2.6. Figure 2.10 shows that the work demand could be determined independently of the number of clients.

Recall that the number of helper processes does not significantly affect the response time as we observed in the previous subsection. Then, one may think that a processor sharing (PS) scheduling discipline is adequate to model the Web server. Its work demand for given document size can be determined using Figure 2.10. It would be applied when our concern is to obtain the response time averaged over all document sizes. However, it may not be true if we want to determine the response time for each document size. Its main reason is that a simple processor sharing scheduling discipline does not reflect the queueing delay at the dispatcher since the dispatcher schedules the requests in a FIFO manner. It is well known that the means of response times of FIFO and PS are identical, but the response time distributions are not. Then the response time distribution would be different from the results obtained by a simple PS scheduling discipline. The effect of the queueing delay at the dispatcher will be discussed in the next subsection.

2.2.4 Experiment 4: The effect of the document size distribution

To observe the effect of the delays at the dispatcher, we considered the following simple file mix.

`/file50K.html` x

`/file500K.html 1000-`x

where x is changed from 1000 to 0. Namely, the workload is built as a mixture of 50 Kbyte and 500 Kbyte documents, and the ratio is changed.

In this experiment, we set the number of helper processes to be 16, and the number of clients is changed as 4, 16, and 32. Results are shown in Figure 2.11 where the horizontal axis is the ratio of two document files. For example, “1000:0” means that the size of all documents is 50 Kbyte, and “500:500” shows that two files are accessed with an equal probability. The vertical axis shows the response time averaged over two kinds of files. In the figure, we first find that each line is straight. Then, if we know the work demand for the given document size, the average response time can be determined. Work demands for 50 Kbyte document and 500 Kbyte document are about 0.07 sec and 0.3 sec, which can be determined from Figure 2.10. Then, we can derive the response times for those two files when we are given the number of clients. If the number of clients is 32, the response times are 2.1 sec and 10 sec. Those correspond to the case of “1000:0” and “0:1000” of “client32” in Figure 2.11. Then, the response time for the case of “500:500” becomes the average of the above two cases, i.e., about 6 msec. It coincides the result shown in Figure 2.11. However, even this fact does not support the inappropriateness of PS scheduling discipline.

A more important fact is that the response time for each document size depends on the ratio when the number of clients is larger than the number of helper processes. Figure 2.12 presents the response time for each document as well as the overall response times. Recalling that we set the number of helper processes to be 16, we plot two cases in the figure; the numbers of clients are 4 and 32. In the case

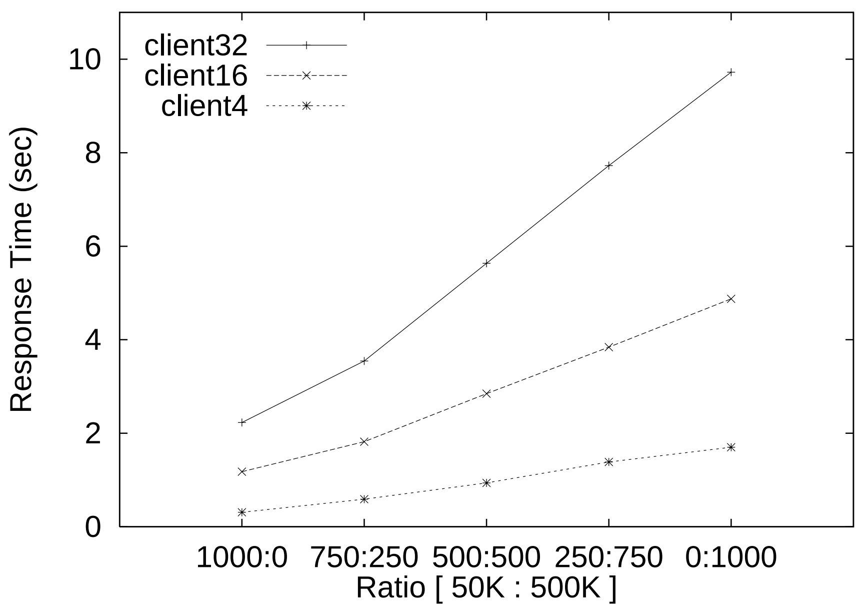

Figure 2.11: Response times of 50 Kbyte and 500 Kbyte documents dependent on the file mix.

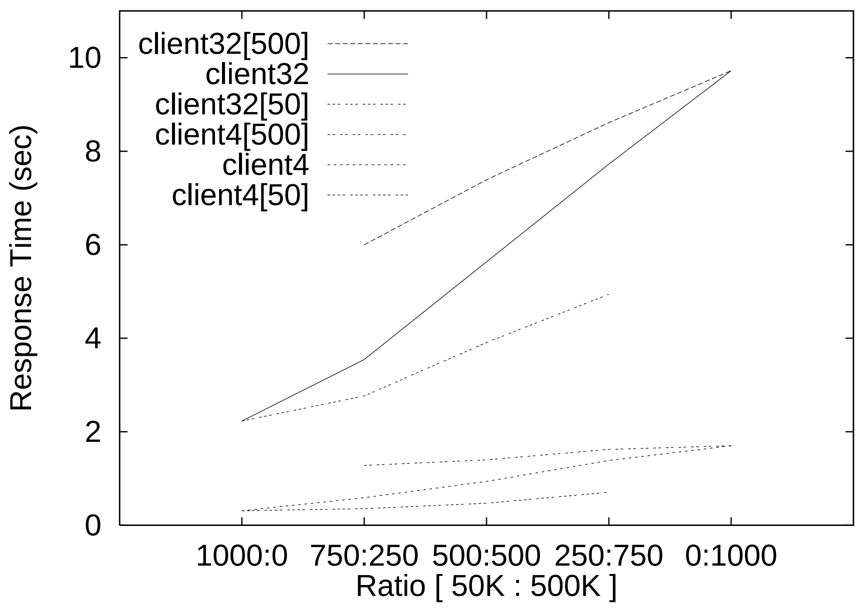

Figure 2.12: Response times of 50 Kbyte and 500 Kbyte documents.

of 32 clients, the response times of 50 Kbyte documents increase as the access frequency to 500 Kbyte documents increases. It is due to the queueing delay at the dispatcher since simple PS does not change the response time for 50 Kbyte file even if the file mix is changed. This observation is supported by results of the case of 4 clients. In this case, the response time of 50 Kbyte documents is affected by the increasing 500 Kbyte documents since the number of the helper processes is sufficient to simultaneously accept the requests from clients.

As another experiment, we used the actual traced data for the input data to WebStone to compare with the results obtained by the default file set, i.e., `filelist.standard` given by WebStone. Actual traced data was analyzed in [2] where the authors showed that the log-normal distribution is suitable to the document size. In Figures 2.13 and 2.14, we compare the average response times and throughput for the above two cases. In this experiment, both of the numbers of helper processes and clients are fixed at 16. In these figures, we can observe very different results. Of course, the averages of two distributions are different. In the case of `filelist.standard`, the average is 19 Kbyte while that of the log-normal distribution (i.e., actual traced data) is 9.5 Kbyte. For a comparison purpose, we also conducted the simulation where each size specified in `filelist.standard` is reduced so that its average becomes 9.5 Kbyte. The results are shown with the label "filelist.standard (average modified)." Average behaviors of two cases ("log-normal" and "filelist.standard (average modified)") are very similar.

Our observation is that the `filelist.standard` does not reflect the real world, and therefore, the benchmark test based on the `filelist.standard` cannot lead to the actual performance prediction. However, if we use the `filelist.standard` by appropriately setting its average, then we can have similar results to the case of the log-normal distribution. The problem is that the WebStone file mix does not allow to specify the document sizes with occurrence probability less than 1/1000. However, the log-normal distribution has a long tail, and its effect cannot be pre-

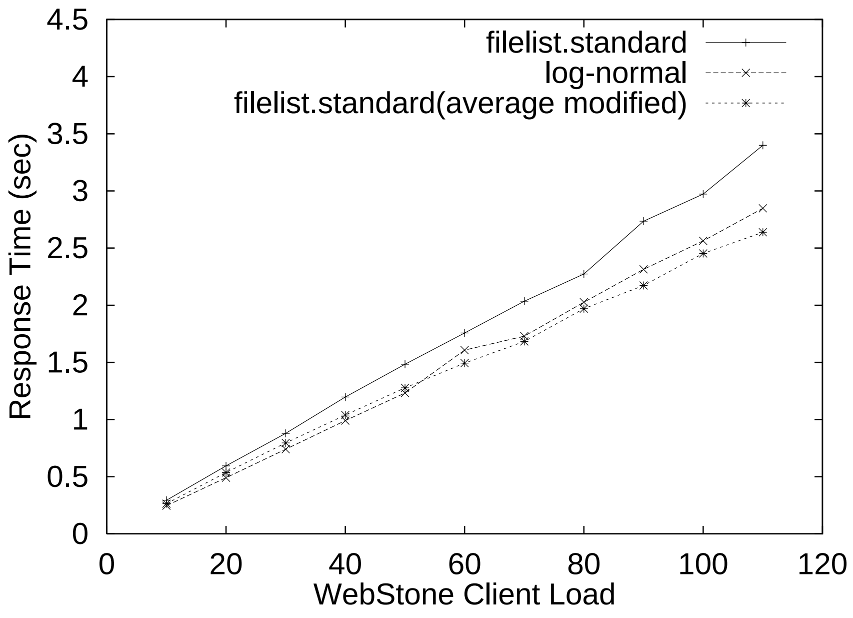

Figure 2.13: Effects of document size distributions on the average response time.

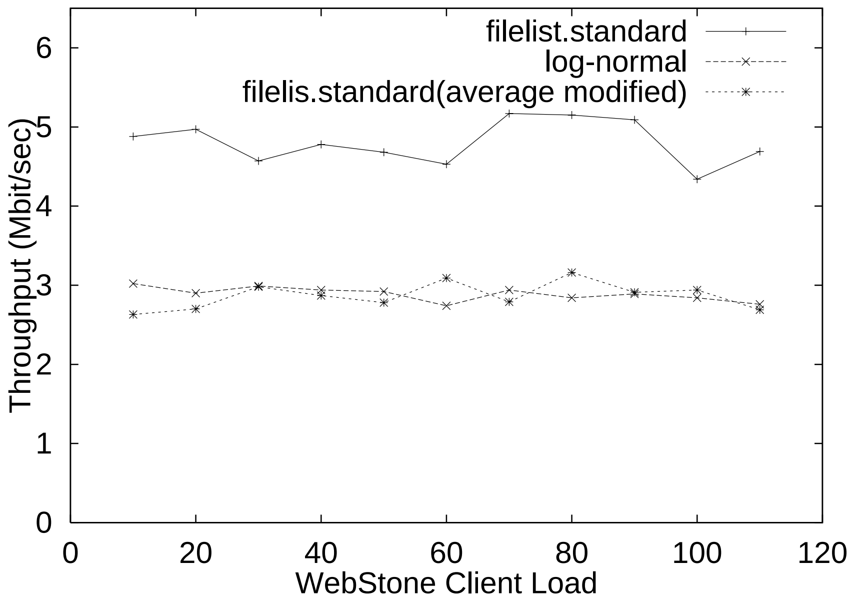

Figure 2.14: Effects of document size distributions on the server throughput.

cisely estimated when using WebStone. We thus develop the analytic method to predict the Web server performance, which will be presented in the next section.

2.2.5 Experiment 5: The effect of network capacity

So far, we have shown the results using the ATM switch such that the network does not become a bottleneck. It is because our primary concern in the current paper is to characterize the Web server performance. In the last experiment, we confirm that the network is also an important factor for the system performance. For this purpose, we replace the network with 10 Mbps Ethernet (10base-T). If we use the low–speed network like Ethernet, the network easily becomes a bottleneck, which is shown in Figure 2.15. In obtaining this figure, we set the number of helper processes to be 16 and the number of clients 24. As the document size becomes large, the obtained throughput is saturated around 7.5 Mbps in the case of Ethernet as shown in Figure 2.15. This value is reasonable for 10 Mbps Ethernet.

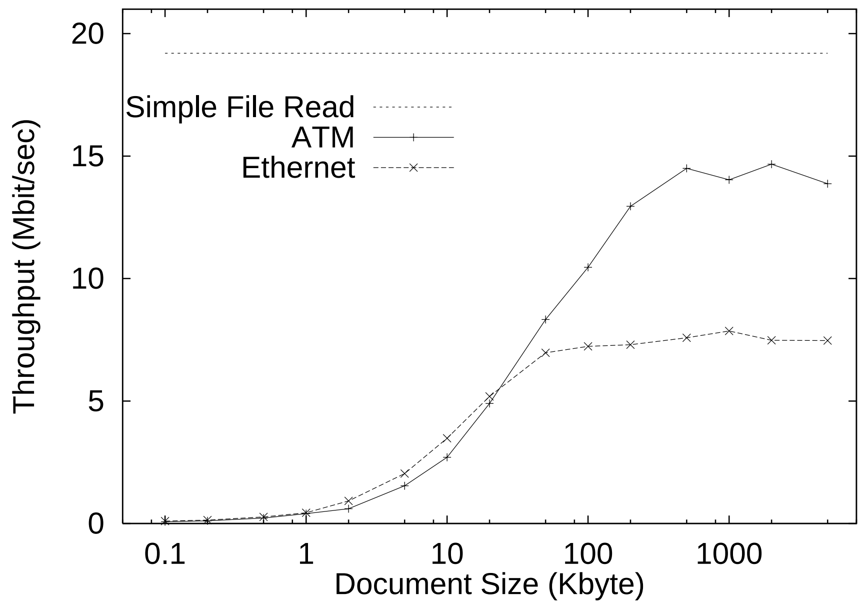

Figure 2.15: Effects of the network bandwidth.

Note that we can also observe the bounded throughput even in the ATM case. It is about 15 Mbps. To investigate its reason, we conducted the experiment in which simple file read operations from the client machine to the server machine were executed. The result is also shown in Figure 2.15 with label “Simple File Read”. As shown in the figure, its value is about 19 Mbps and we can confirm that the bottleneck resides in the server machine not in the network in the case of ATM.

2.3 Performance Modeling and Analysis of the Web Server

From discussions above, we have confirmed that the Web server can be modeled by the combination of FIFO queueing discipline at the dispatcher and PS scheduling discipline once the request is assigned to one of helper processes. Figure 2.16 illustrates a queueing model for the Web server. The quantitative parameters to represent the work demand for given document size can be determined from our experimental results. See Figure 2.10. In this section, we propose and examine the model of the Web server in detail.

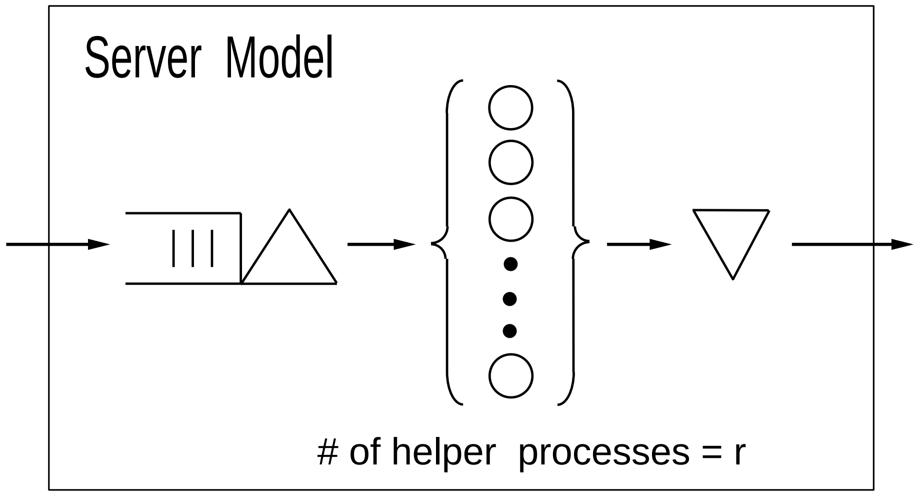

Figure 2.16: Queueing model for the Web server.

2.3.1 M/G/1/PS queue with a limited number of jobs in the server

In our modeling, we assume that jobs (document requests) arrive at the Web server following a Poisson process with rate λ. If the number of jobs in the server is less than or equal to r, the job can enter the server to receive the service. If the number of r jobs has already been in the server, the newly arriving job waits in the FIFO queue. The jobs in the server receive equal service; i.e., the server gives service to jobs in the server following the PS discipline. It is apparent that if $r = \infty$, the model becomes a simple PS server model. The PS scheduling had been a lot of attentions in the literature. See, e.g., [33]. However, in the case of $r < \infty$, only a few studies are known. In [34], the response time distribution is derived by assuming that the work demand follows an exponential distribution. The mean response time conditioned on the work demand is obtained in [35], but the authors [35] also assume an exponential work demand, and only an overall mean response time is approximately derived for general work demand distribution.

We thus need to newly analyze the model where the work demand follows the general distribution so that the result can be applied to our Web server model. Our main interest is to derive the conditional mean delay of this system for given work demand (i.e., document size). We first introduce S to represent the random variable for the work demand of each job, which follows a general distribution. The random variable representing the delay experienced in both of the FIFO queue and PS server is represented by R. Then we express the average response time in the system for given work demand x, $E[R|S = x]$, as

$$E[R|S = x] \;\;=\;\; W + T(x).$$

Namely, it consists of the job's waiting time in the queue, W, which is independent of the work demand, and the time duration spent in the server, $T(x)$. It is approximately represented as

$$E[R|S = x] \;\;=\;\; \frac{\rho^r}{1-\rho}\,\frac{E[S^2]}{2\,E[S]} + \Delta + \frac{1-\rho^r}{1-\rho}x, \tag{2.1}$$

where $E[S]$, $E[S^2]$ are the first and second moments of the work demand distribution, and ρ represents the traffic load given by $\rho = \lambda E[S]$. Δ is the processing time for dispatching. By letting the work demand distribution be $S(x)$, we have from Eq.(2.1)

$$E[R] \;\;=\;\; \int_0^\infty E[R|S = x]\,dS(x)$$

$$= \frac{\rho^r}{1-\rho}\frac{E[S^2]}{2\,E[S]} + \frac{1-\rho^r}{1-\rho}E[S], \tag{2.2}$$

which coincides an approximation result proposed in [35]. If $r = 1$, Eq.(2.1) becomes

$$E[R|S = x]_{FIFO} = \frac{\rho}{1-\rho}\frac{E[S^2]}{2E[S]} + x, \tag{2.3}$$

which is a result of M/G/1(/FIFO) queueing system (see, e.g., [36]). Further, if $r = \infty$, Eq.(2.1) is reduced to

$$E[R|S = x]_{PS} = \frac{x}{1-\rho}. \tag{2.4}$$

It is a well known conditional mean response time of M/G/1/PS queueing system.

In the case of $1 < r < \infty$, we need to assess the accuracy of Eq.(2.1), which will be shown in the next subsection.

2.3.2 Numerical examples and discussions

In the Web server, r corresponds to the number of helper processes, and the newly arriving document requests waits at the dispatch queue if the number of document requests in the server exceeds r. As has been shown in the previous section, the work demand depends on the document size, but it does not linearly increase against the document size. Keeping these facts in mind, we now give some discussions on how the Web server performance can be improved.

The simple PS system provides conditional mean response time as a linearly increasing function of the work demand as shown in Eq.(2.4). On the other hand, the processing delay is largely dependent on the distribution of work demands in the FIFO system as in Eq.(2.3). Then, our system provides the following features.

- As the larger number of helper processes, r, are prepared, the system approaches the PS system. It means that the small-sized document is less influenced by the large-sized documents. It is important in the Web system since the Web server had better quickly respond to retrieval requests of text documents and inline images. However, we should note here that it is not always true since the processing time of even short documents is not small in our case. We will demonstrate this aspect in the below.

- In the light traffic load, the fixed delay W, which is independent of the work demand, becomes negligible. Then, the response time is almost proportional to the work demand of the request. However, as the traffic load becomes heavy, the fixed part W relatively gets large, and the response time for even small-sized documents grows.
- The fixed part W is in proportion to $E[S^2]$. Namely, if the work demand follows the long tailed distribution such as the log-normal distribution, response times for requests of small-sized documents become large.

In what follows, we quantitatively evaluate the above observations. Our model assumes the Poisson process of job arrivals, which is based on [2] where the authors analyzed the access log gathered at the Web server and showed that the interarrival time of the document requests follows the log-normal distribution, but the Poisson arrival becomes adequate during busiest hours. To quantitatively evaluate the Web server, we take the following procedure.

(1) We approximately determine the work demand dependent on the document size. For this purpose, we can use the data obtained in 2.2.3. Using the results shown in Figure 2.10, we derive the work demand function $g(x)$ dependent on the document size x [Byte]. By assuming that the dispatcher overhead time is negligible, we obtain the following fitting function.

$$g(x) \quad = \quad ax + b, \tag{2.5}$$

where $a = 0.0004851$ and $b = 21.55$. See Fig.2.17 where we plot $g(x)$ and the values obtained in Figure 2.10.

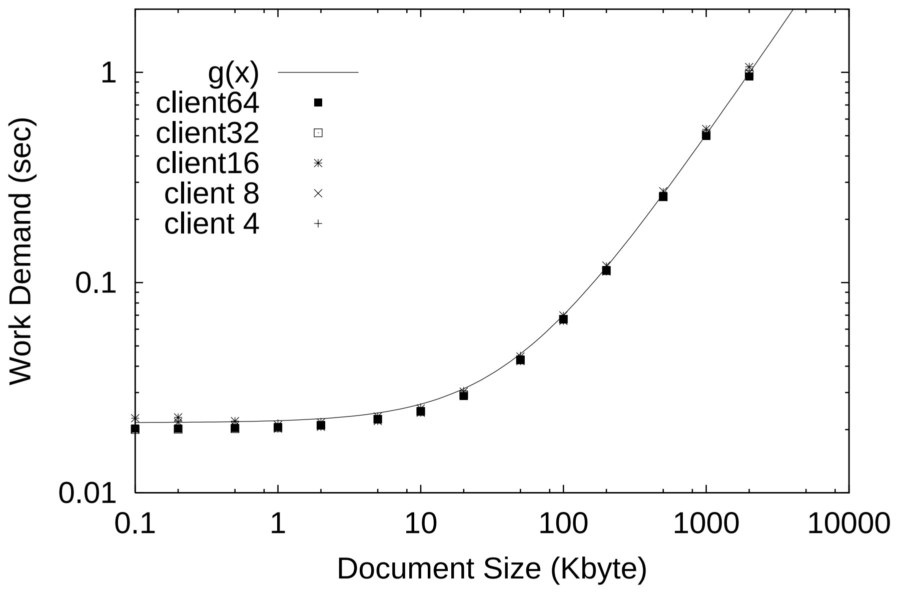

Figure 2.17: Approximation of work demand.

(2) We determine the actual work demand distribution by taking into account the document size distribution. By letting a probability density of the document size distribution be $f(x)$, we can determine p the probability distribution of work demands as

$$S(x) = \int_0^x f(h(y))h'(y)dy, \tag{2.6}$$

where $h(y)$ is an inverse function of $g(x)$, i.e.,

$$h(y) = \frac{y-b}{a}.$$

(3) The probability distribution $S(x)$ of the work demand can be now applied to the analytic result obtained in the previous subsection.

In what follows, we consider the case where the document size distribution $f(x)$ follows the log-normal distribution [2]. The random variable X follows the log-normal distribution if another random variable $Y(= \log X)$ follows the normal distribution $N(\mu, \sigma^2)$. We then have the probability density of the log-normal distribution as

$$f(x) = \frac{1}{\sqrt{2\pi}\sigma x} e^{-\frac{(\log x - \mu)^2}{2\sigma^2}}.$$

The mean and the second movement of the log-normal distribution are given as

$$\begin{aligned} E[Y] &= e^{\mu + \sigma^2/2} \\ E[Y^2] &= e^{2(\mu + \sigma^2)}. \end{aligned}$$

Consequently, the probability density function $s(y)$ of the work demand is determined from Eq.(2.6) as

$$s(y) = \frac{1}{\sqrt{2\pi}\sigma(y-b)} e^{-\frac{(\log(y-b) - \log a - \mu)^2}{2\sigma^2}}.$$

The authors in [2] have shown that the actual Web document size follows the log-normal distribution with parameters $\mu = 7.811$ and $\sigma = 1.573$. Since we have $a = 0.0004851$ and $b = 21.55$ the mean and the second moment of the actual work demand become

$$\begin{aligned} E[S] &= 25.68 \\ E[S^2] &= 844.7 \end{aligned}$$

Using Eq.(2.5), the conditional mean response time $d(x)$ for the document with size x is given as follows;

$$d(x) = W + T(a * x + b).$$

In the rest of this subsection, we present numerical examples using the above result. The accuracy of our approximate result obtained in Eq.(2.1) is also demonstrated. Since in our current modeling, the document requests are assumed to arrive according to the Poisson distribution. Since, our previous experimental results are based on the *closed queueing network model*, the direct comparisons of two results are impossible. We therefore conducted simulation experiments to assess the accuracy of our result. Each simulation was run until the 95% confidence interval of the response time is converged to 5% of its mean. In the following figures, simulation results will be marked with symbol '□' while analytic results are shown with solid lines. We should note here that once the accuracy of our result is validated, we can apply it to the closed queueing network model by approximately modeling the Web server as an IS (Infinite Servers) queue since the IS queue can allow the general service time distribution [36].

We first plot the mean conditional response time dependent on the document size in Figure 2.18 where the number of helper processes, r, is fixed at 4, and five cases of the traffic load are shown; $\rho = 0.2$, 0.4, 0.6, 0.8 and 0.9. From this figure, we can observe that the conditional mean response times of the small sized documents are dramatically increased as the traffic load becomes high.

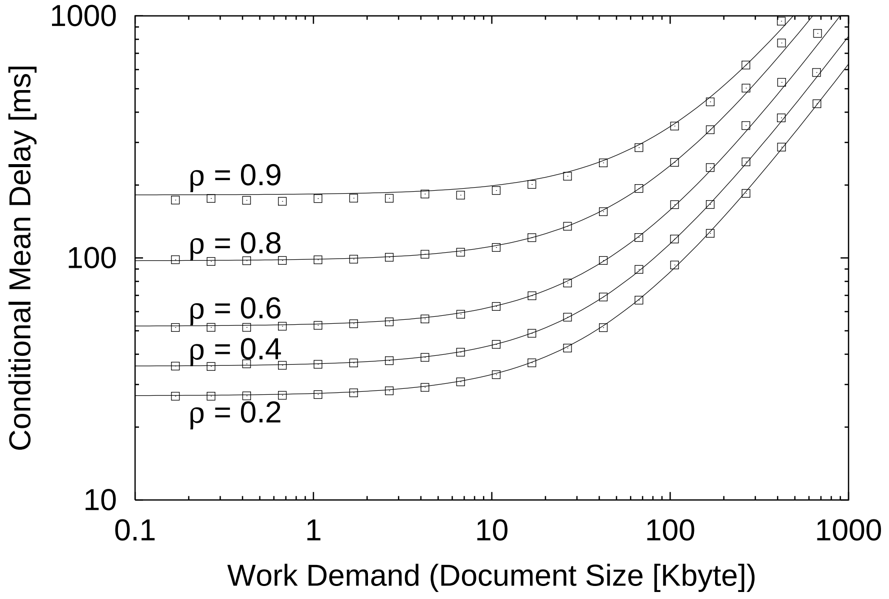

Figure 2.18: Conditional mean response time for different traffic load.

We next change the number of helper processes, r. In Figure 2.19, the conditional mean response times for various r are plotted. Here, we set the traffic load ρ to 0.8. From the figure, we can observe that as the number of helper processes r becomes large, the response time performance is degraded. Namely, the figure indicates that the smaller number of helper processes is preferred. It is a rather curious result if we consider the FIFO and PS systems. In the PS system ($r = \infty$), the conditional mean response time is in proportion to the document size (see Eq.(2.4)), while in the FIFO system ($r = 1$), the fixed value of the waiting time at the queue (the first term of right hand side of Eq.(2.3)) gives a dominant part, and the response time becomes independent of the work demand. Then, our system seems to fall between two systems. That is, the conditional mean response time for the small-sized document is decreased as r becomes large while those for the large-sized document are increased for larger r. However, it is not true as shown in the figure. A reason is that the work demand of small–sized documents (less than 10 Kbyte) are almost the same in our system, and in the log-normal distribution of the document size, many documents are small. Then such an effect cannot be observed in the figure.

To confirm this observation, we fictitiously enlarged the document size distribution. The mean of document size distribution is set to be ten times larger than the one observed at the actual system. The document size distribution remains to be the log-normal distribution and two parameters of the log-normal distribution were determined such that the coefficient of variation is not changed. The result is shown in Figure 2.20. In this case, the fixed part of the work demand (b in Eq.(2.5)) becomes negligible and then results show the one we expected.

From the above two results of Figures 2.19 and 2.20 , we may conclude that for the current document size distribution, the larger number of helper processes does not help to improve the Web server performance. However, in the future Web service, documents with large size are likely to be increased. Those include the motion video and audio data. In that case, the number of helper processes becomes an important factor to determine the quality of service of the Web server in terms of

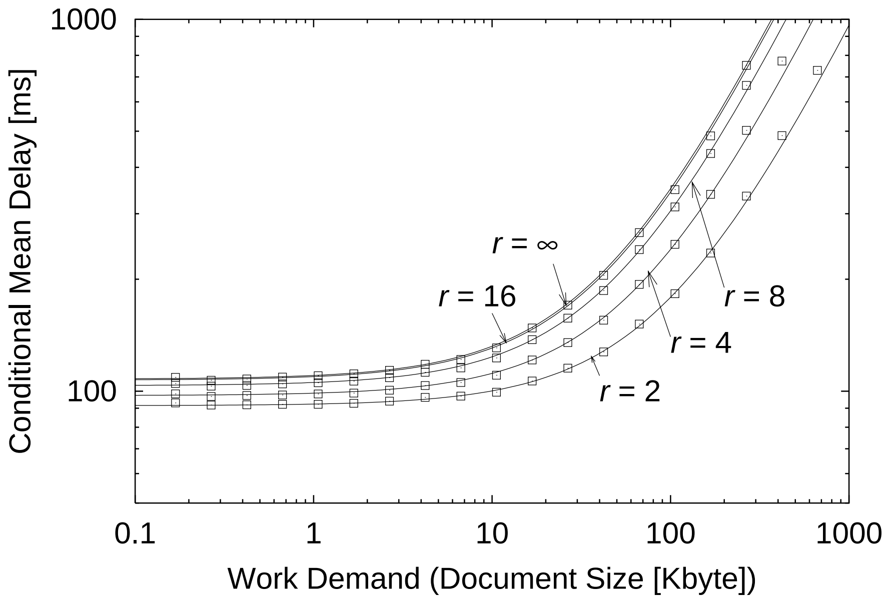

Figure 2.19: Conditional mean response time dependent on the number of helper processes.

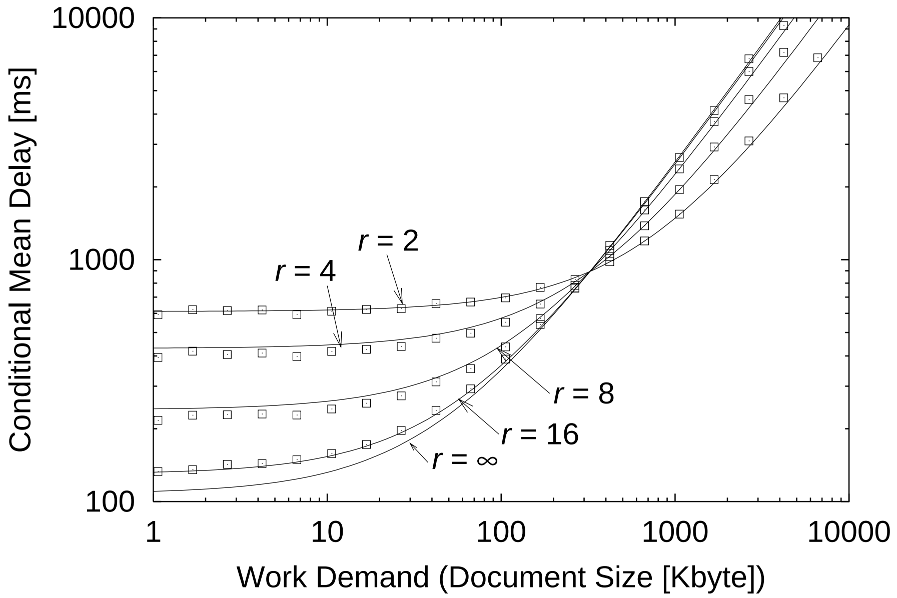

Figure 2.20: The case of ten times larger document size distribution.

document retrieval times. Namely, if one may want to keep the small response time of the short document such as text files, the number of helper processes should be increased at the expense of the degraded response times of the large documents.

We finally investigate the influence of the document size distribution on the Web server performance. For this purpose, we use the exponential distribution for the document size as well as the log-normal distribution. In the case of the exponential distribution, the mean of document sizes is set to identical, i.e., mean work demand is

$$E[S] \;=\; 25.68$$

The second moment becomes smaller than that of the log-normal distribution;

$$E[S^2] \;=\; 676.5$$

The comparative results are presented in Figure 2.21. From the figure, it is clear that the response time of the log-normal distribution becomes larger. However, when the number of helper processes becomes large, the difference of two distributions is quite low. It can be verified from the fact that in the PS system, the conditional mean response time for given work demand is independent of the work demand distribution.

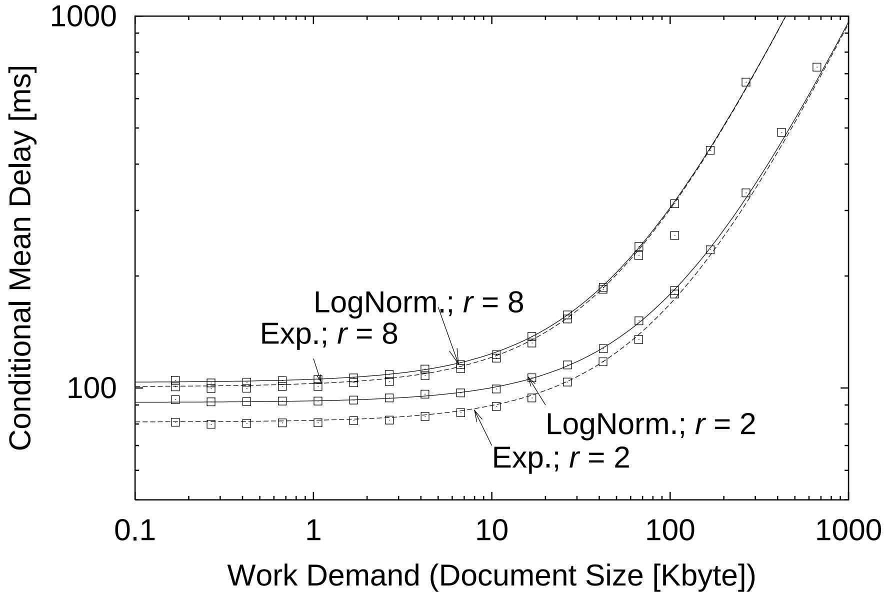

Figure 2.21: Comparisons of conditional mean response times for different document size distributions.

Chapter 3

Analysis and Modeling of Proxy Server Performace

3.1 Experimental Configuration of the Proxy Server

Our experimental system is illustrated in Figure 3.1. We used the ATM switch to interconnect the client, the Web server and the Proxy server to avoid the network being bottleneck. The document request from the client is sent to the Proxy server first. If the document does not exist in the Proxy server, the Proxy server forwards the request to the Web server. As the Proxy server retrieves the document, the Proxy server transfers its document to the client. Noting that we used the ATM switch, we found that the overhead due to the network delay is negligible when compared with the time elapsed at the servers. Thus, our experimental results can be used to discuss the performance of the servers.

In our experiments, we control the document hit ratio explicitly to investigate the performance capability of the Proxy server. When we set the hit ratio to be 100%, the client always requests the document located in the Proxy server. On the other hand, when the hit ratio is set to 0%, the client does not request the document on the Proxy server, but on the Web server. We set the Proxy server option that the Proxy server

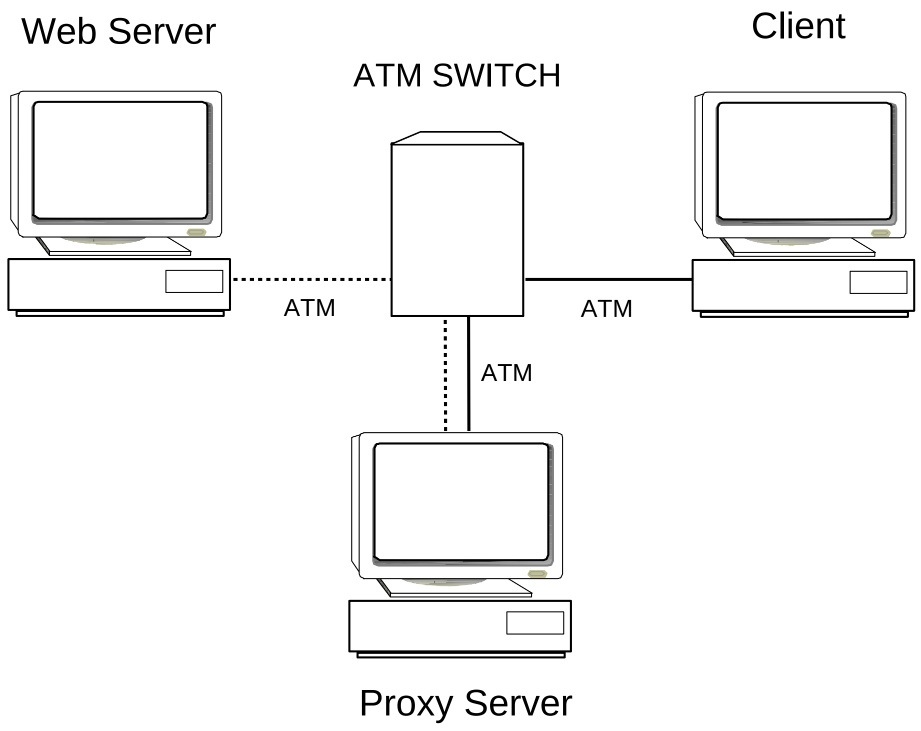

Figure 3.1: Experimental system configuration of the Proxy server.

does not cache the document in the latter case, by which the document requested from the client is always forwarded to the Web server. Then, the performance of the Proxy server can be found for given hit ratio. While the hit ratio must depend on the caching algorithm implemented within the Proxy server, the differences are not large as shown in [2].

The hardware and software configurations in our experiments are summarized in Table 1. We used Apache (version 1.2.5) [27] Squid (version 1.2.20) [37] for the HTTP server and for the Proxy server, respectively. Both of two servers are configured as follows; the logging option is set as default. Document files are stored in the local disks of the Proxy server and the Web server. On all the experimental machines, we did not modify the operating system for tuning TCP/IP stack. That is, we set up the experimental system similar to the commonly used Web servers and Proxy servers.

Table 3.1: Experimental system of the Proxy server.

	Web Server	Proxy Server	Client
Machine	Indigo2(SGI)	Indigo2(SGI)	Origin200(SGI)
CPU	IP26 75 MHZ	IP22 250 MHZ	IP27 180MHz *2
Main Memory	320 Mbytes	192 Mbytes	256 Mbytes
Web Server	Apache 1.2.4	—	—
Proxy Server	—	Squid 1.1.20	—
Benchmark	—	—	WebStone 2.0.1

We used WebStone (version 2.0.1) [12] by which we specify the distribution of requested document sizes, the frequency of access to the Proxy server, and the number of clients generating loads for the Proxy server. We did not follow the WebStone run rules [15] in some experiments since we aim at collecting the quantitative data for Proxy server modeling. In such cases, we will explicitly present the configuration. In each simulation setting, we tested five experiments, each of which runs ten minutes, to obtain the reliable results. Then, the average of those two experiments is presented in the below. Last, we note that in our experiments, we only considered the document transfers, and did not include requests that require the processing at the Web server such as cgi.

3.2 Experimental Results of the Proxy Server

In this section, we present the results of our experiments as follows;

- The effect of the number of clients to access the Proxy sever at the same time
- The effect of the hit ratio of the cache

In our experiments, we used the way that the helper processes are prepared on the Web server for *http* daemons. The number of helper processes is fixed 16. Discussions on the way to prepare helper process and its effectiveness can be found in Capter 2. In our approach, the size of all requested documents is fixed in each experiment.

3.2.1 Experiment 1: The effect of the number of clients to access the Proxy sever

First, we investigate the effect of the number of clients to access the Proxy server at the same time. We change the number of clients to be 8, 16, 32 and 64. The hit ratio of all documents is set to be 100%, by which we can identify the basic performance of the Proxy server because it can exclude the processing time at the Web server. We show the result in Figure 3.2. From the figure, we can observe that in the case of the small sized documents, the response time is not in proportion to the document size but is almost fixed. On the other hand, as the document size becomes over about 100Kbyte, the response times increase almost linearly with the document size. To see this more clearly, we present Figure 3.3 where the vertical axis is illustrated in the log-scale.

We next show Figure 3.4 where the response times shown in Figure 3.2 are divided by the number of the clients. Namely, Figure 3.4 represents the "work demand" for each request on the Proxy server against the document size. From the figure, we can confirm that the work demand cannot be modeled directly from the document size. However, the effect of the number of clients is negligible except very heavy traffic load conditions where the context switch for the processes at the server becomes overhead.

Next, we investigate the work demand in the case of cache miss. First, we request the document to the Proxy server setting the hit rate as 0% and measure the response time retrieving the document from the Web server via the Proxy server. Secondly, we request the same document to the Web server directly and measure the response time similarly as above. Finally, we use the difference time between these response

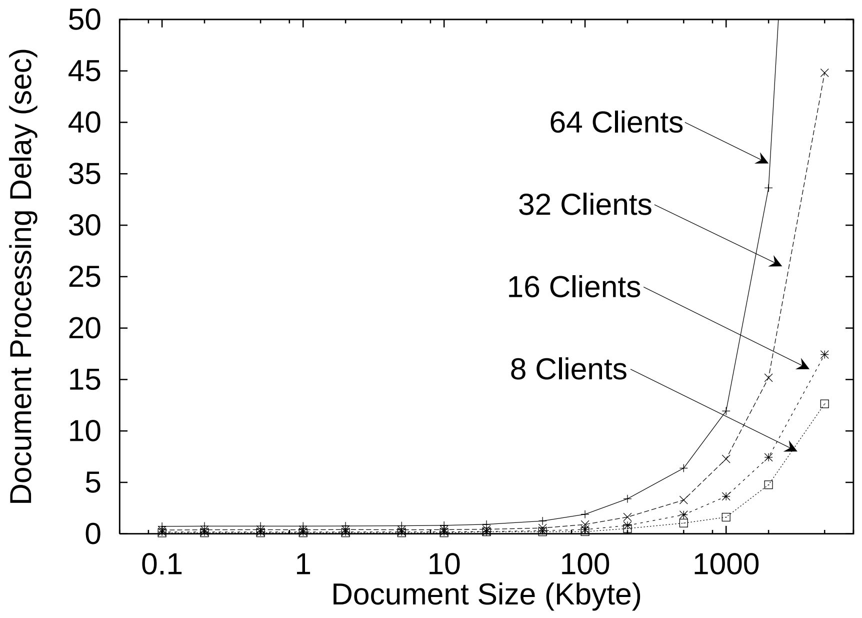

Figure 3.2: Effects of the number of clients [100% hit ratio].

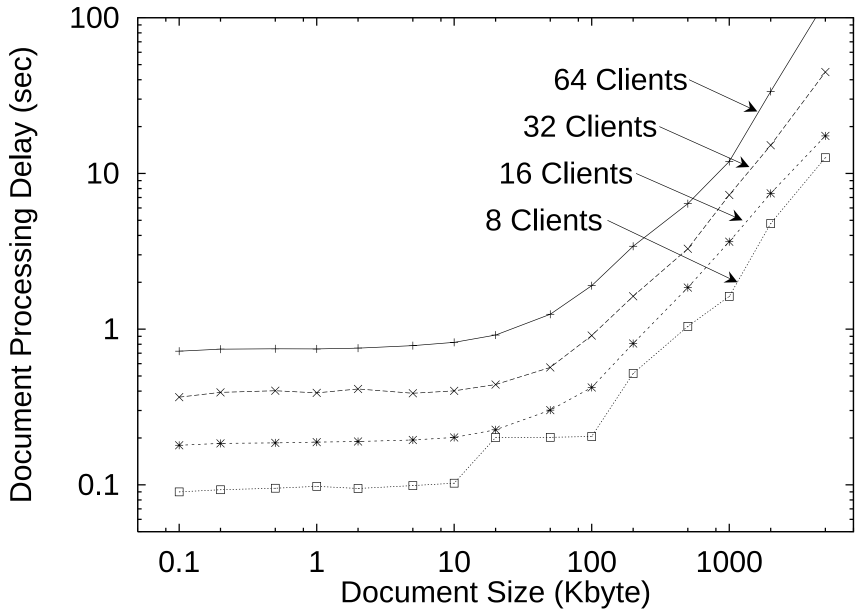

Figure 3.3: Effects of the number of clients [100% hit ratio] (log scale).

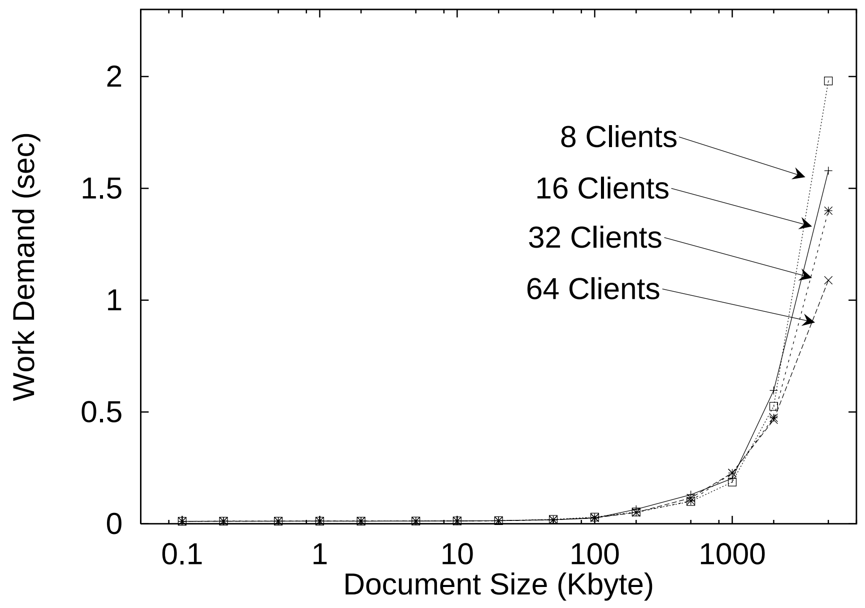

Figure 3.4: Work demand in the case of 100% cache hit.

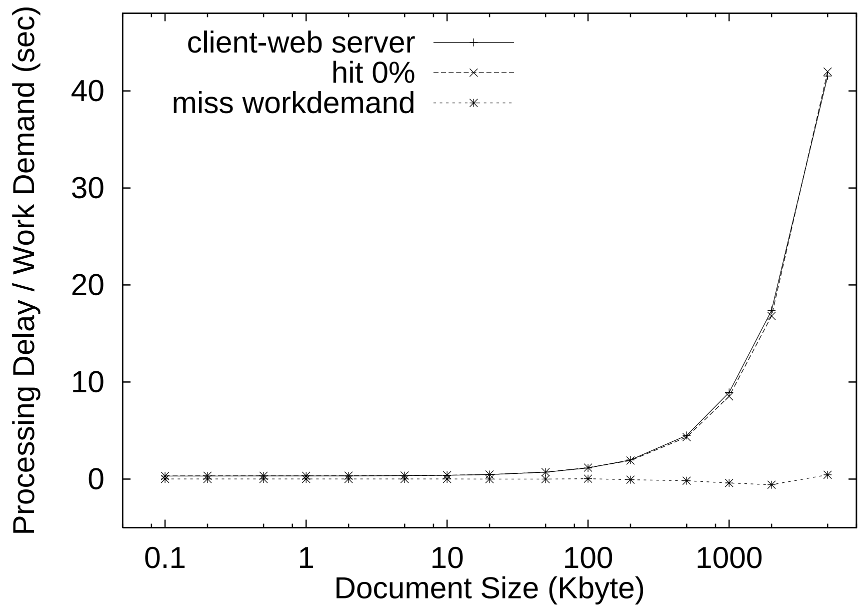

Figure 3.5: Work demand in the case of cache miss.

times as a parameter in the case of the cache miss. (see Figure 3.5)

3.2.2 Experiment 2: The effect of the cache hit ratio

In this subsection, we want to investigate the response time characteristics of cached and no–cached documents for given cache hit ratio. In this experiment, the hit ratio is controlled as follows. The client requests two kinds of documents; the cached document and no–cached documents on the Proxy server. The requested document cached on the Proxy server is directly returned from it. On the other hand, the document not on the Proxy server is transferred from the Web server, but the document is not cached on the Proxy server by setting the document name in the configuration file `squid.conf` on the Proxy server. By changing the access rate of the cached and no–cached documents in `filelist.standard` of WebStone, we can vary the hit ratio on the Proxy server. Example of setting 40% cache hit ratio is below.

```
/hitdoc.html 400
/misseddoc.html 600
```

When `misseddoc.html` is requested, the Proxy server, Squid, forwards the document request to the Web server, Apache, to retrieve the document, and forwards the document. In this experiment, the Proxy server does not keep the document, `misseddoc.html`, to its disk. As the client issues the requests according to the probability, as a result, the cache hit ratio becomes 40%.

Figure 3.6 shows the response times averaged over cached and no–cached documents for sixteen clients. The figure presents the fact that as the hit ratio becomes larger, the response time becomes smaller as expected. Figure 3.7 is the corresponding one where the vertical axis is in the log–scale. From the figure, we can observe the same tendency as in the previous case (100% hit ratio case, which is also shown in the current figure).

Of course, response times of only cached–documents exhibit different appear-

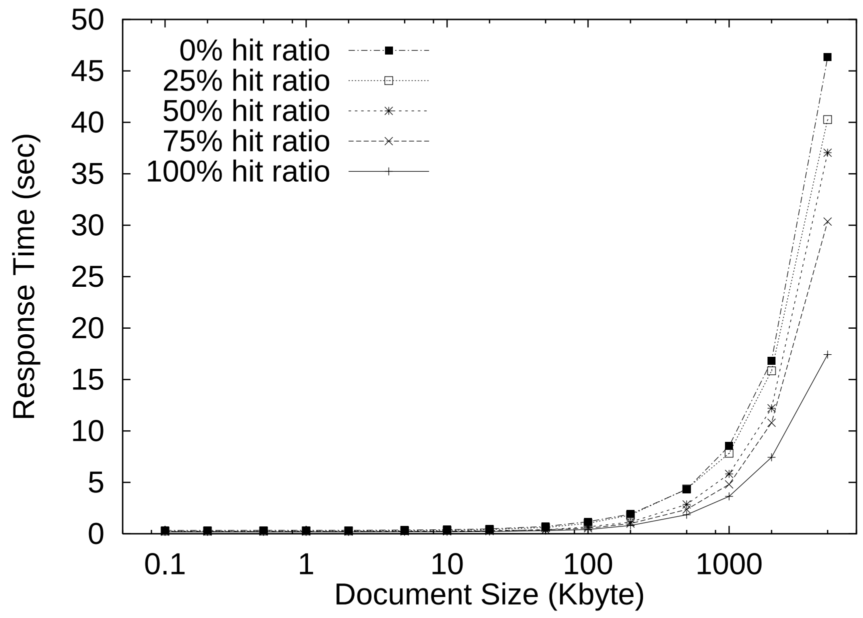

Figure 3.6: Effects of the cache hit ratio on the average response times.

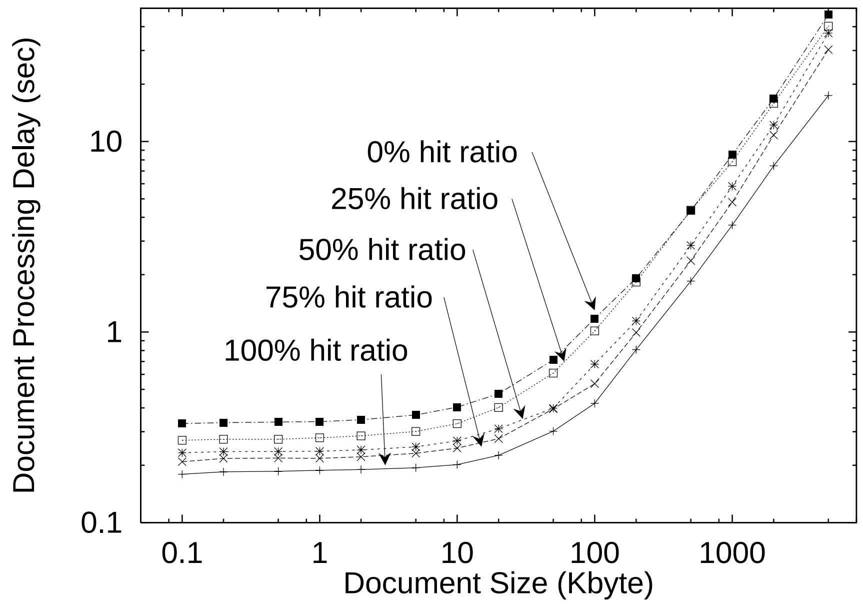

Figure 3.7: Effects of the cache hit ratio on the average response times (log scale).

ance as shown in Figure 3.8. When the document size is small, the response times become large as the hit ratio gets high. It is due to high processing load the Proxy server. However, when the document size becomes large, the lower hit ratio leads to the larger response times. In this case, the processing overhead is balanced between the Proxy and Web servers. Thus, we need to consider the effect of the document size as well as the hit ratio to determine the work demand on the Proxy server.

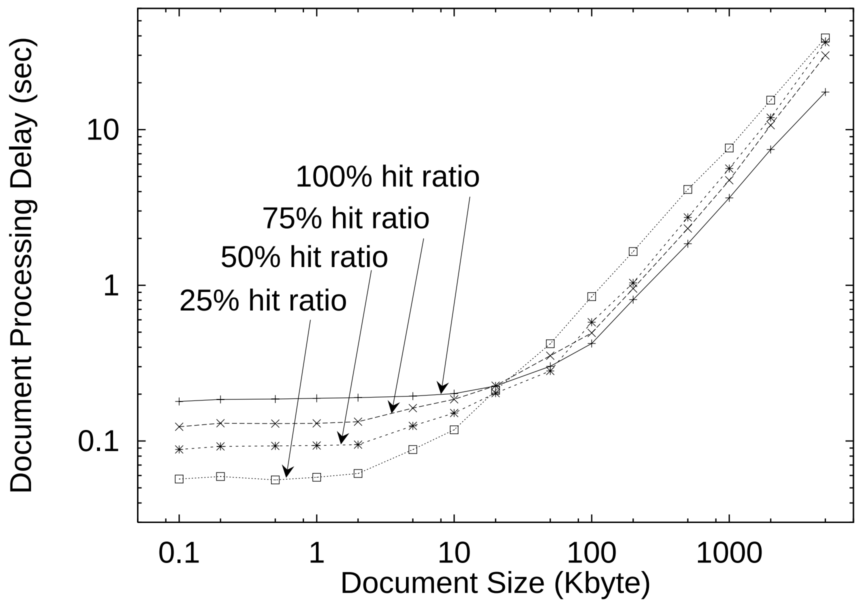

Figure 3.8: Response times for only cached documents.

For the above purpose, we take a following simple approach. As shown in the previous figure, the tendencies of the response times are changed when the document size is 20Kbyte. Thus, we consider two regions according to the document size. When the document size is smaller than 20Kbyte, the response times are fictitiously decreased as the hit ratios become small. On the other hand, when the document size is over 20Kbyte, the response times are enlarged according to the hit ratio. Fortunately, the relation between the response times and hit ratios are simple as shown in Figure3.9; i.e., a linear relation can be found as shown in the figure.

Thus, for given document size, we can use linear functions, $h_1(y)$ or $h_2(y)$,

$$h_1(y) = 0.125 + 0.00875y \tag{3.1}$$

$$h_2(y) = 2.25 - 0.0125y \tag{3.2}$$

for given hit ratio y, to determine the work demand once the work demand for the case of 100% hit ratio is found.

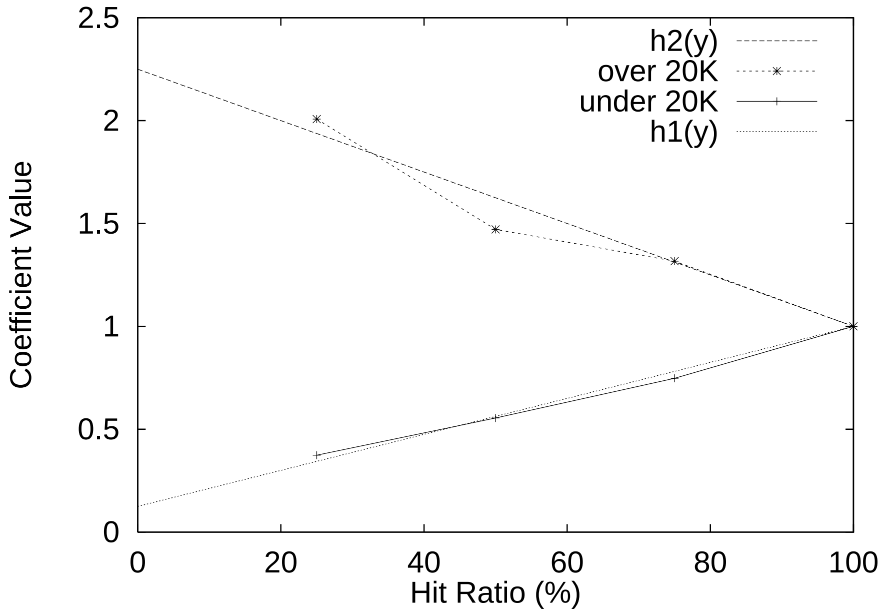

Figure 3.9: Coefficients of work demands against response times with 100% hit ratio.

The work demand for 100% ratio case has already been shown in Figure 3.4. By utilizing the curve fitting approach, we have the following relation.

$$g(x) = ax + b \tag{3.3}$$

where $a = 0.01121$, $b = 0.0002329$. See Figure 3.10, where we plot $g(x)$ on Figure 3.4.

From discussions above, we have confirmed the processing time on the Proxy server for the given cache hit ratio and document size. Furthermore, we consider the way to prepare helper process, and from the result of experiment 3.2.1, the Proxy

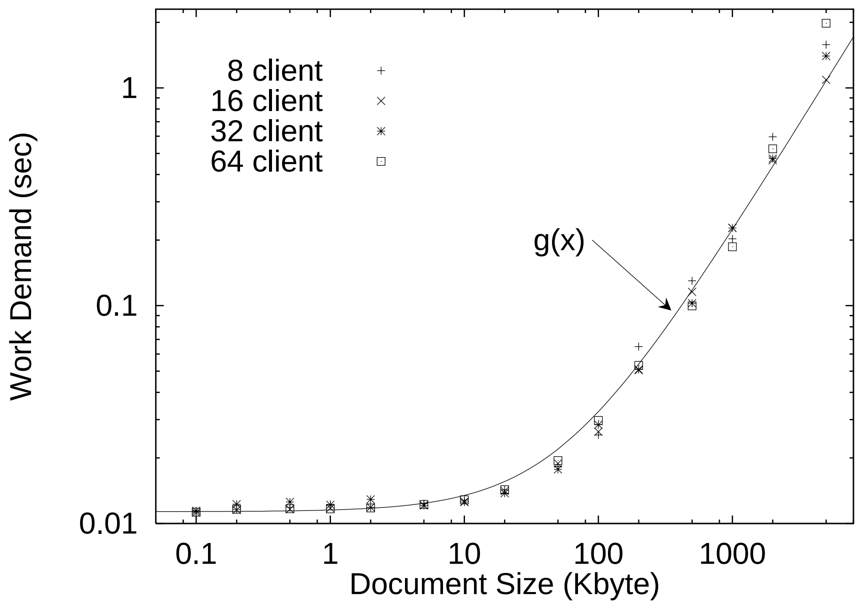

Figure 3.10: Approximated function for work demand.

server can be modeled by a processor sharing scheduling discipline. Actually, although the coefficient value is examined for all document size, we use the two functions, $h_1(y)$ and $h_2(y)$, which have the different tendencies in the point of the 20Kbyte document. In the next section, with these results, we show the examples of the performance evaluation of Web system.

Chapter 4

Performance Modeling and Examples of Web Server Systems

In this Chapter, we demonstrate applicabilities of our Web system modeling approach to evaluate performance of the Web system.

(1) The case where the Web site is publicly open to the Internet.

(2) The case where users within a certain local network (e.g., LAN and Internet service provider) access the Internet via the Proxy server.

In each model as mentioned above, the performance of the Web server and the Proxy server, the bandwidth of the Internet access line and the Internet backbone are the primary factors to affect the performance evaluation model. In the following, we show the simulation results of these two models.

4.1 Evaluation of the Web Server System

We consider the case where the Web site is publicly open to the Internet as shown in Figure 4.1. The Web server is modeled as an M/G/1/PS with a limited number of jobs. The Web server is accessed via the access line from the Internet users. The access line is modeled as a M/G/1/PS queue [2]. The performance results derived from our model are mainly affected by the Web server performance and the bandwidth of the access line connected to the Internet. Of course, the Internet backbone is also an important factor affecting the performance, but it is beyond our scope how the Internet backbone should be improved. In the current thesis, the Internet backbone is expressed by an IS (Infinite Server) queue and the delays follow the Erlangian distribution [38]. Its mean is set to be 195 msec according to [38]. As a result, we have an open queueing network model. However, we conducted simulation experiments, since the scheduling discipline taken at the Web server does not satisfy the product–form network. Note that in the current subsection, we do not have the

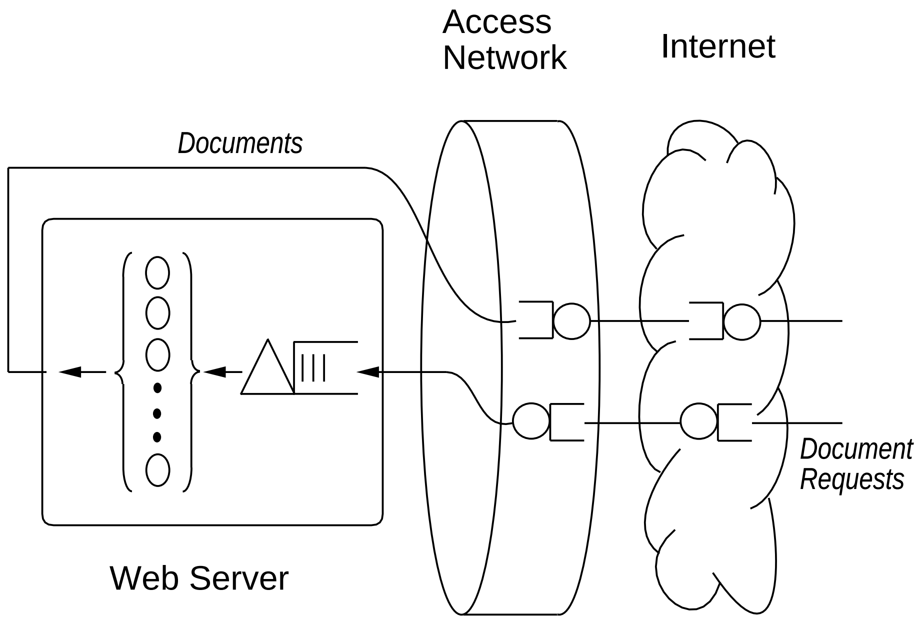

Figure 4.1: Web server system model.

Proxy server in the model. The effect of the Proxy caching will be investigated in the next subsection.

First, we investigate the delay of the access line connected to the Internet backbone. We use the average number of requests per second and the access line band-

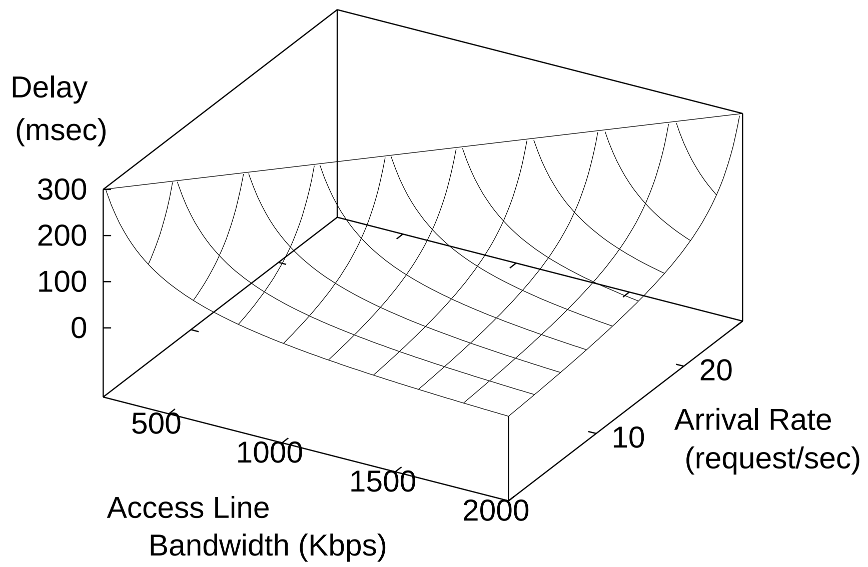

Figure 4.2: Mean delays on access line.

width as parameters. And so, Figure 4.2 shows the average transfer delay in the access line. Now, we present the case that the transfer delay in the access line is limited by “300ms”. From the figure, we can see that the transfer delays in the access line are dramatically increased as the number of requests becomes large, and we are able to find out the area that the access line becomes the bottleneck.

In Figure 4.3, we show the access line bandwidth necessary to obtain the pre–determined mean document transfer delays. That is, the figure represents the required bandwidth to satisfy “100ms”, “200ms”, or “300ms” mean document transfer delays for the Internet users. The horizontal axis shows the arrival rate of the requests from the Internet users. The figure indicates that, if the mean request arrival rate is 10 [requests per sec.], we need the access line bandwidth more than about 1.4Mbps to keep the average delay below 100 msec. We can also observe that the larger access line capacity can improve the delays especially when the request arrival rate is small. However, its effect is limited as the request arrival rate becomes

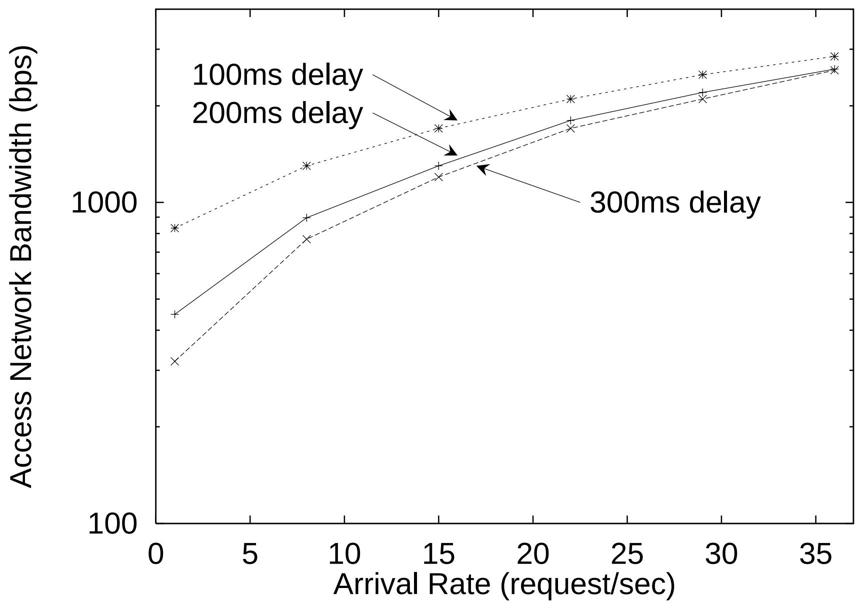

Figure 4.3: Required access line capacity for given delay constraint.

large. It is due to the fact that the other resources (in the current case, the Web server) become bottleneck.

We next see the effect of the Web server performance. Figure 4.4 shows the delays experienced at the Web server and the access line separately. The total delays are also shown in the figure. In the figure, two cases of the access line capacity are shown; 768 Kbps and 1.5 Mbps. The document request rate is fixed at 5 request / sec. The horizontal axis shows the fictitious relative server performance by setting our Web server to be 1. From the figure, we can observe that the improvement of the Web server performance is important to improve the total delays in the current parameters setting. However, dramatic improvements cannot be observed as the Web server processing power becomes large since the delay within the Internet becomes dominant in that region. Then, one must wait the advancement of the Internet backbone for further performance improvement after the access line and the Web server are adequately prepared.

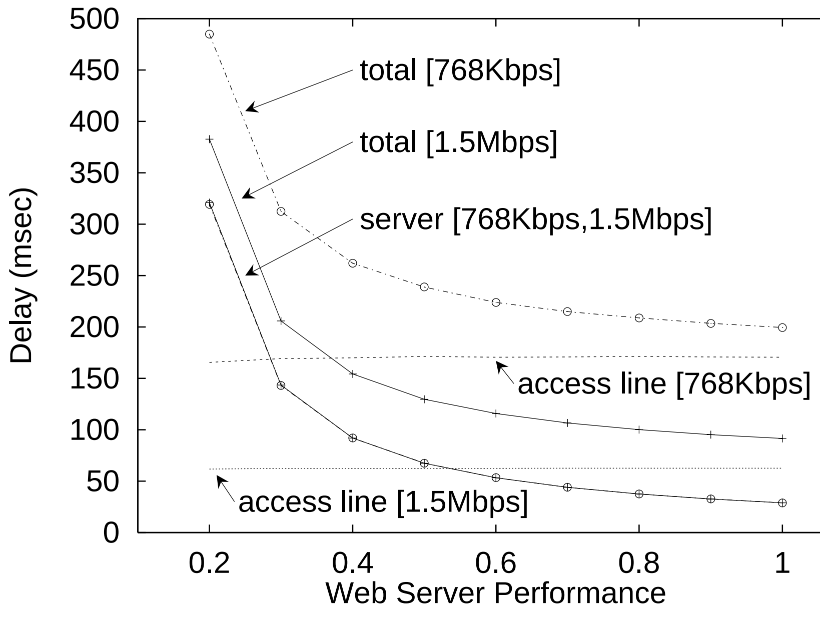

Figure 4.4: Delays dependent on the Web server performance.

So far, we have assumed that the delay within the Internet backbone is known. In numerical examples above, the delay is assumed to follow the Erlangian distribution with mean 195 msec [38]. It was obtained by `ping` command between the sites located in the East and the West of United States. For the Web system planning, we need the following quantities.

(1) the Web server performance

(2) the characteristics of the document size

(3) the access rate of the document requests

(4) the delay characteristics of the Internet backbone

The Web server performance is characterized in the current paper while it may depend on the server platform. The active researches on the WWW traffic have exhibited the characteristics of the document size (see, e.g., [2]). The access rate cannot be known a priori since it must be heavily dependent on the contents. The system monitoring is thus mandatory for accurately assessing the access rate. The last item, the delay characteristics of the Internet backbone, should be investigated more actively in the field for the Web system planning to be actually meaningful.

4.2 Evaluation of the Model Including the Proxy Server System

In this subsection, we show the method of modeling the Web system to evaluate the quality of service given to users within a certain network (e.g., the network of Internet service provider). It is illustrated in Figure 4.5. Recently, most of ISP (Internet Service Provider) provides the Proxy server to improve the document response times. The model is intended to show the applicability of our approach to investigate the effect of the Proxy caching.

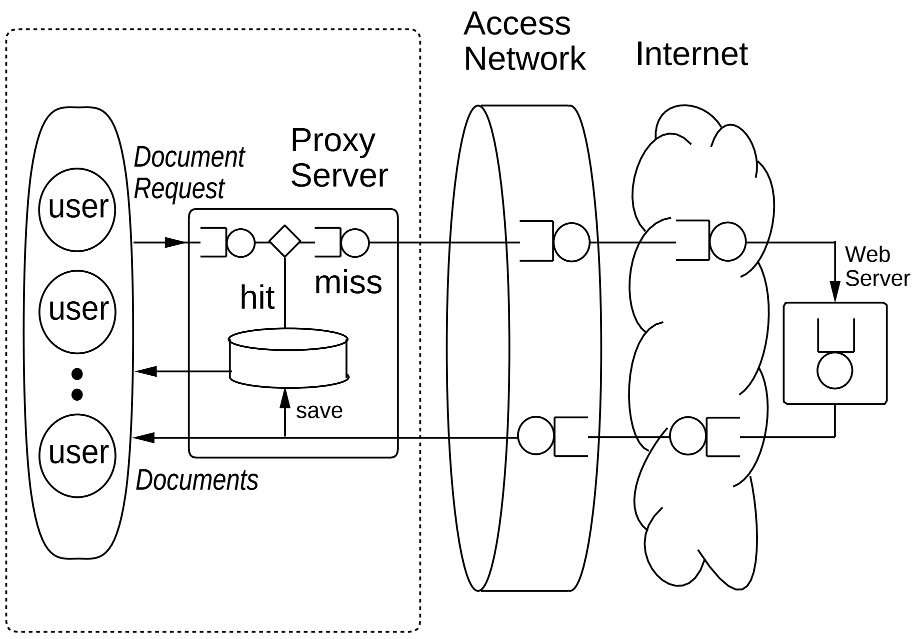

Figure 4.5: ISP model including the Proxy server.

See Figure 4.5 where the hit or miss of the document on the Proxy server is decided independently with given a hit ratio. The validation of this "independent assumption" is given in [2] where the authors show that a correlation effect of the caching algorithm is negligible. A rational behind this result is that if the caching table is large enough, the document misses are likely to happen only due to the wide spread of the WWW document popularity. Queueing models for the Web server, the access line and the Internet backbone are just the same as in the previous subsection. Here, we consider the mixed queueing network model where open and closed chains exist in the model. The users within some ISP are assumed to be fixed, and they request the document repeatedly after they get the response.

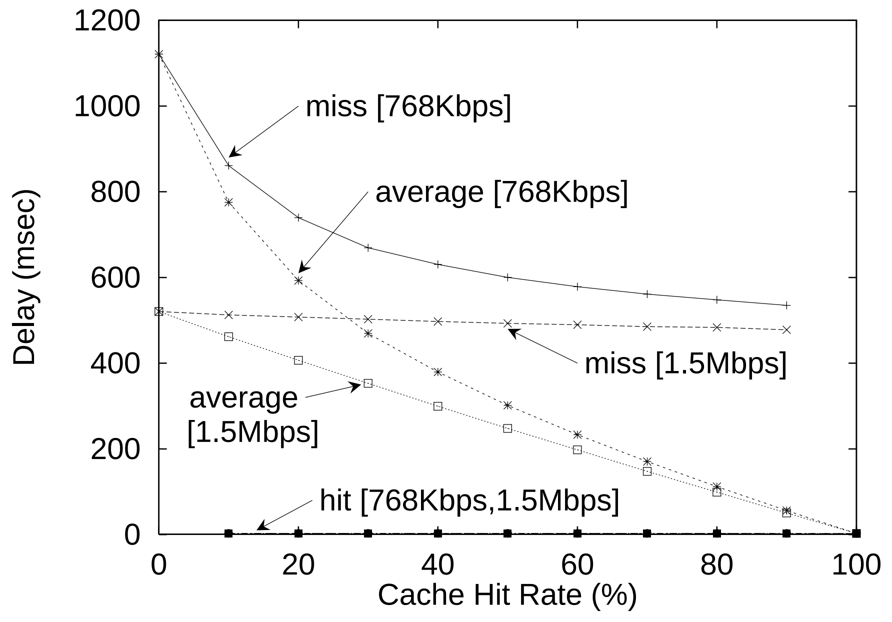

Figure 4.6: Mean delays on the access line dependent on the cache hit ratio.

We first show the delays experienced on the access line dependent on the cache hit ratio of the Proxy server in Figure 4.6. The delays averaged over cache–hit and cache–miss documents are also shown in the figure. The 768 Kbps and 1.5 Mbps access lines are considered. In obtaining the figure, the delay within the Internet backbone is assumed to follow an Erlangian distribution with mean 195msec.

In case where the access line is 1.5 Mbps, the mean delay of the documents is not changed independently of the cache hit ratio. This is because the access line is not the bottleneck in this case, i.e., there is enough bandwidth to transfer the requested document even if the miss ratio is 100%. On the other hand, when the access line bandwidth is decreased to 768 Kbps, the improved cache hit ratio can lead to the smaller document transfer delays to some extent. However, when the cache hit ratio exceeds about 50%, the delays are not very much improved due to the fact that other resources become bottleneck. Since the current caching algorithms offer 50% or 60% hit ratios, 1.5Mbps of the access line bandwidth is a critical value, but more bandwidth is not necessary for ISP to save the cost. Furthermore, the result implies that a more complicated and slightly improved caching algorithm does not help improving the delay within the access network.

Finally, we show the location of the performance bottleneck in our closed queueing network model. Figure 4.7 illustrates the delays at the access line, the Web server, the Proxy server and Internet backbone. The horizontal axis shows the delay experienced within the Internet backbone. Other parameters are not changed and the access line capacity is set to be 1.5Mbps. The hit ratio at the Proxy server is 51%. This percentage is shown in [3]. The delay at the Web server is only for document with cache miss. Figure 4.8 represents the ratio of each delay to the total delay. Noting that we have used 195 msec mean delay for the Internet backbone in the previous examples, the figures imply the effect of higher backbone networks. Of course, the faster Internet backbone improves the total response time as shown Figure 4.7. However, it does not always lead to the dramatic improvement. As shown in Figure 4.8, by the faster Internet backbone, the performance bottleneck moves to other location; it is the access line in this case. Our modeling method can identify it, which is one of main purposes of this thesis. Next, we show the case of that the capacity of the access line changes from 1.5Mbps to 6Mbps. In this case, the

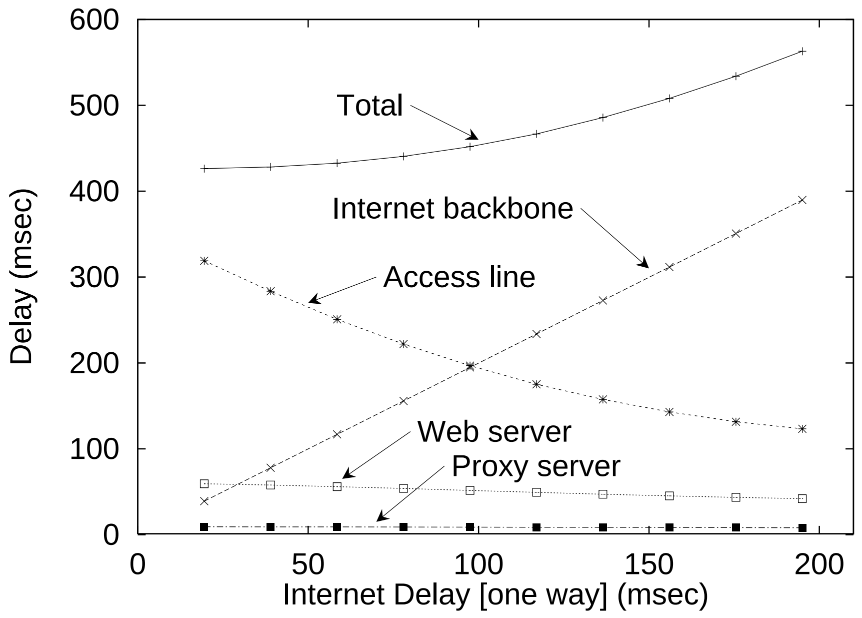

Figure 4.7: Speed–up effect of Internet backbone.

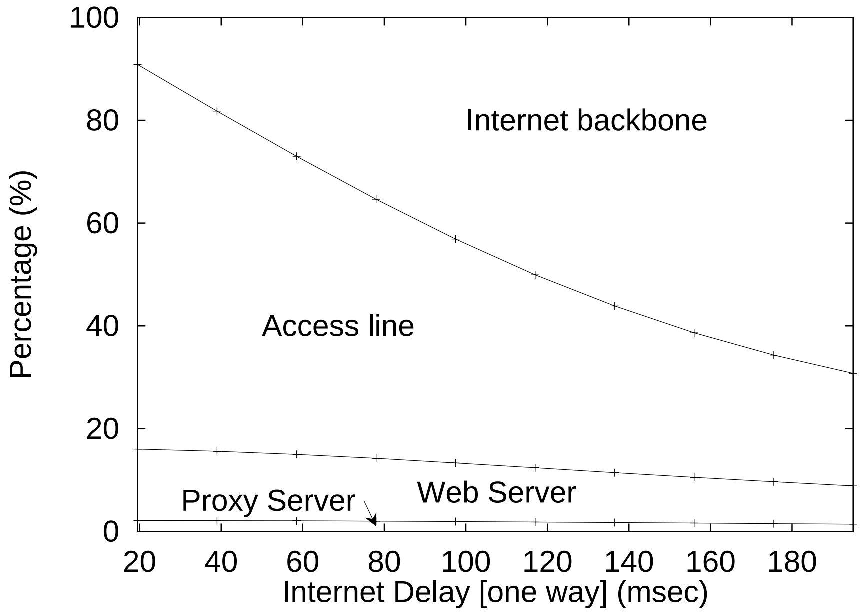

Figure 4.8: Ratios of document processing and transfer delays.

performance bottleneck moves to the Web server by the faster Internet backbone. Figure 4.9 shows the effect of the the access line capacity. And Figure 4.10 illustrates this result precisely. If the delay at the Internet backbone indicates “100msec”, the total delay becomes “300msec”. That is, the capacity of the access line changes 4 times, but the total delay does not improve so much. It is about three fourth as long as the total delay in case that the access line is 1.5Mbps. Next, the faster access line is expected. In this case, it is necessary to evaluate the required capacity and the effective use of the access line by using our Web server system model shown in this thesis.

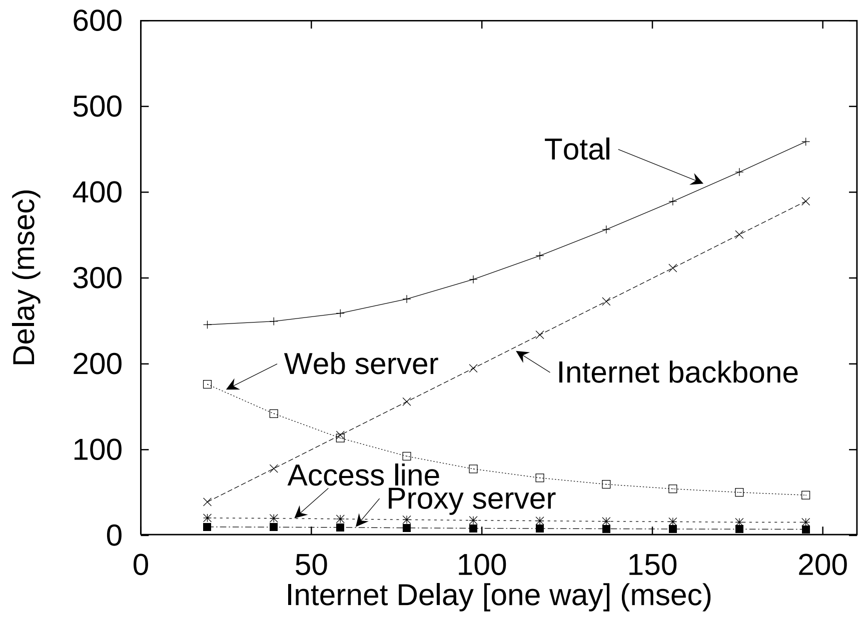

Figure 4.9: Speed–up effect of Internet backbone [The case of 6Mbps access line].

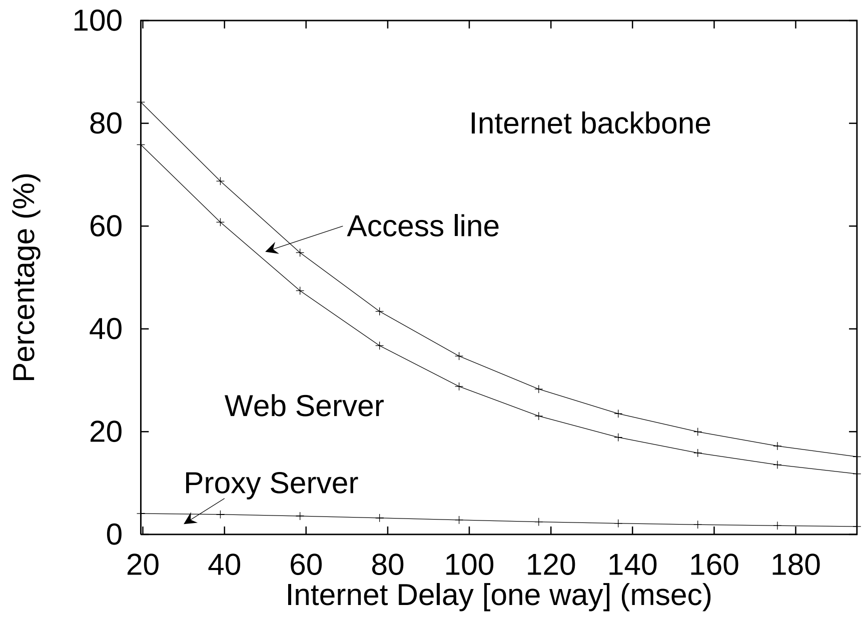

Figure 4.10: Ratios of document processing and transfer delays [The case of 6Mbps access line].

4.3 Accuracies of an Approximate Analytical Method

So far, we have used the simulation technique to evaluate the queueing network model. It is because the scheduling discipline in the Web server does not satisfy the condition of a product–form solution (see, e.g., [36]). In this subsection, we investigate the applicability of the analytical method to solve the queueing network model. For this purpose, we model the Web server as the Infinite Server (IS) queue where the work demand at the IS queue is obtained from a separate analysis of the Web server. The response time at the Web server can be obtained by using the method presented in Chapter 2 where the arrivals of document requests follow the Poisson distribution. Then, the queueing network models shown in this paper become product–form networks, and we can utilize the convolution algorithm and / or the MVA method for effective numerical computation [36]. However, the arrival rate at the Web server is not known a priori when it is applied to the closed queueing network model. Thus, we need an iterative calculation as follows;

(1) An initial value of the throughput of the queueing network model is decided. It can be achieved by, e.g., setting the delay at the Web server is zero. Set the obtained throughput to λ.

(2) λ is used as the arrival rate of the document request at the Web server. Then, the Web server is solely analyzed 2. The obtained response time is used as a work demand of the Web server modeled as IS queue within the closed queueing network model.

(3) The closed queueing network model is analyzed by some appropriate queueing network analysis method, through which we can obtain the throughput at queues. In our numerical example below, we will use the MVA method.

(4) If the throughput obtained in Step (3) is converged to the arrival rate of Step (2), the iteration is terminated. Otherwise, return to Step (2) for next iteration.

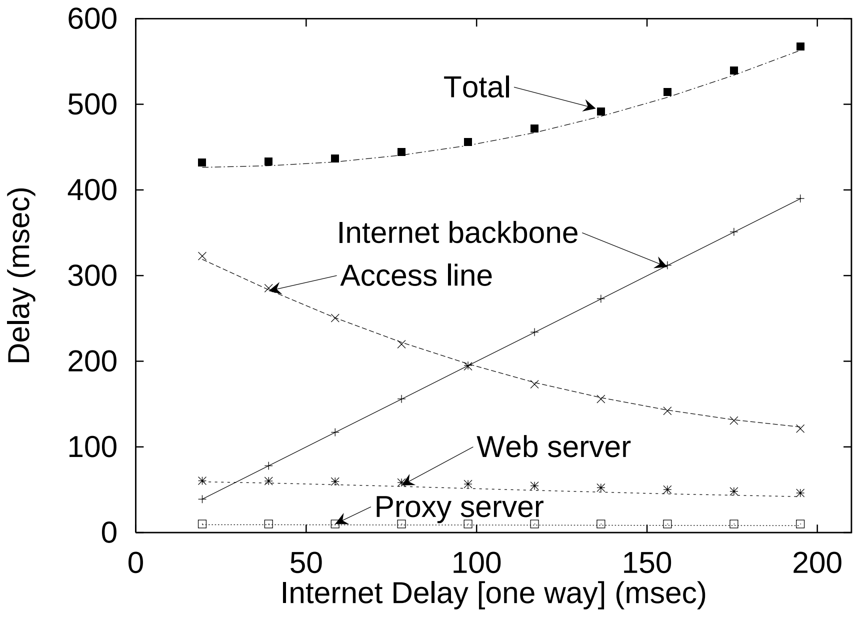

Figure 4.11: Comparisons of analytical and simulation results.

To assess the accuracy of the above approximate analysis, we compare those with simulation experiments. The access line is set to be 1.5 Mbps. The delays at the Internet backbone, the access line, the Web server and the Proxy server are compared in Figure 4.5. The lines show the analytical results while the symbols are for the simulation results. In obtaining Figure 4.11, the delay of the Internet backbone is changed; i.e., the figure corresponds to Figure 4.7. In Figure 4.11, good accuracy of the analytic method can be observed. In this case, however, either Internet backbone and the access line is the bottleneck. Since we introduced the approximation in the Web server queue, we need to investigate the situation that the Web server also becomes the bottleneck.

For this purpose, we change the access line from 1.5 Mbps to 6 Mbps. The comparative results for delays are shown in the Figure 4.12. The Figure shows the cases that the requested documents exist in the cache of the Proxy server and do not. In the following figures, simulation results will be marked with points while analytic results are shown with solid lines. The corresponding results for the throughput in the above two cases are summarized in Figure 4.13. In this case, the accuracies are lost, but are sill kept in the reasonable level. Hence, it is obvious that we are able to estimate the performance characteristics simply, by using our approximate analytical method.

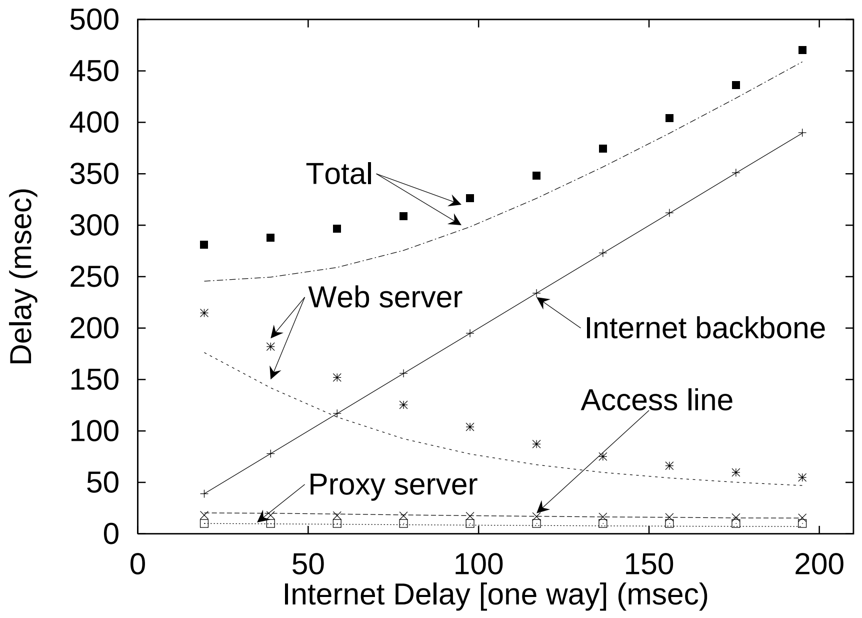

Figure 4.12: The case of 6Mbps access line.

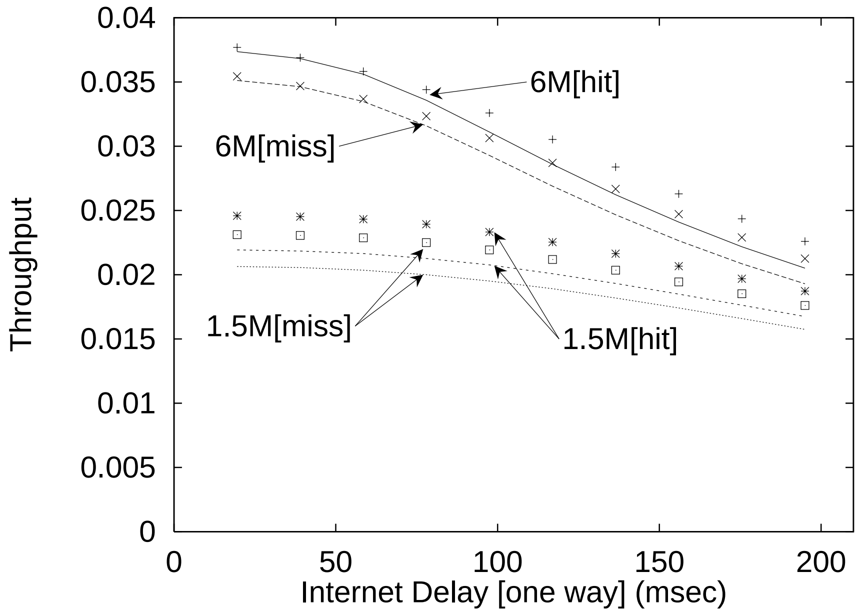

Figure 4.13: Comparisons of throughput.

Chapter 5

Conclusion

First, in Chapter 2, we have obtained the various experimental results to investigate effects of (1) the way to prepare the HTTP daemon, (2) the number of helper processes, (3) the document size and response time, (4) the document distribution on the Web server, and (5) the network capacity. From our experimental results, we have shown that the performance of the Web server can be improved by preparing the helper process, which has represented the effective and highly efficient performance, for the `http` daemon on the Web server. The response time is almost proportional to the number of clients for a given document size. So, one may think that a PS scheduling discipline is adequate to model the Web server. And its work demand is obtained from the response time divided by the number of clients. It is applied when our concern is to obtain the response time averaged over all document sizes. However, in experiment (4), by using the two files as a workload composed of a mixture of two different size documents, we have shown that the response time for a given document size is influenced by the other document size. So, we have confirmed the effect of the queueing delay at the dispatcher. Hence, we have proposed that the Web server can be modeled by the combination of FIFO queueing discipline at the dispatcher and PS scheduling discipline once the request is assigned to one of helper processes. The quantitative parameters to represent the work demand for a given document size has been determined from above mentioned.

Next, we have proposed the performance model and its analytical method of the Web server based on those experiments. Then, we have newly analyzed the model where the work demand follows the general distribution so that the result can be applied to our proposed Web server model. And we have expressed the equation to derive the conditional mean delay of this model for a given work demand. The accuracy of our approximate result has also demonstrated with our simulation experiments. Through numerical examples, we have discussed the performance engineering problem of the Web server. Then, we have observed that the conditional mean response times of the small sized documents are dramatically increased as the traffic load becomes high. Furthermore, for the current document size distribution, the larger number of helper processes does not help to improve the Web server performance. However, in the future Web service, the average of the document size distribution is likely to be increased. Those include the motion video and audio data. In that case, the number of helper processes becomes an important factor to determine the quality of service of the Web server in terms of the response time.

In Chapter 3, we have presented experimental results to investigate the effect of (1) the number of clients to access the Proxy server at the same time, and (2) the hit ratio of the cache. Based on the experimental results, we have built the model of the Proxy server. In experiment (1), the response time is not in proportion to the document size. To identify the basic performance of the Proxy server, the cache hit ratio at the Proxy server is set to be 100%. In this case, we have obtained the work demand for each request on the Proxy server against the document size. And also, the work demand for no–cached documents has been obtained. In experiment (2), we have investigated the response time characteristics of cached and no–cached documents for given cache hit ratio. As the cache hit ratio becomes larger, the response time becomes smaller as expected. On the other hand, response times of only cached documents exhibit different appearance. Thus, we need to consider the effect of the document size as well as the cache hit ratio to determine the work demand on the Proxy server. So, we used two linear functions for given document size. Accordingly, as the way of preparing the http daemon is helper process, the Proxy server can be modeled by a PS scheduling discipline.

We have modeled the Web server system and the Proxy server system in previous Chapters. In Chapter 4, we have constructed the performance evaluation model of the Web server systems. From our results, we have demonstrated applications of the Web server model to performance evaluation of the Web server system. We haven't considered about the delay which the document transmission and the http protocol cause. Our modeling approach is then demonstrated by using two models; (1) the model in which the Web site is publicly open to the Internet users, and (2) the Proxy server is provided for ISP users within the Internet access line. From our model (1), dramatic improvements of the response time cannot be observed as the Web server processing power becomes large. So, one must wait the advancement of the Internet backbone for further performance improvement, that is the response time, after the access line and the Web server are adequately prepared. In our model (2), the faster Internet backbone improves the total response time, but it does not always lead to the dramatic improvement. The main reason is that the performance bottleneck moves to other location. Finally, we have investigated the applicability of the analytic method to solve the queueing network model. We have modeled the Web server as the IS queue where the work demand at the IS queue has obtained from the result of Chapter 2. We have accessed the accuracy of our approximate analysis by comparing with simulation experiments.

Our approach can identify the performance bottleneck of the Web system and can be used for its performance planning. We have presented the evaluation result of the Web server system and the Proxy server system separately. Accordingly, the entire model would become the one illustrated in Figure 5.1, and it can be easily evaluated.

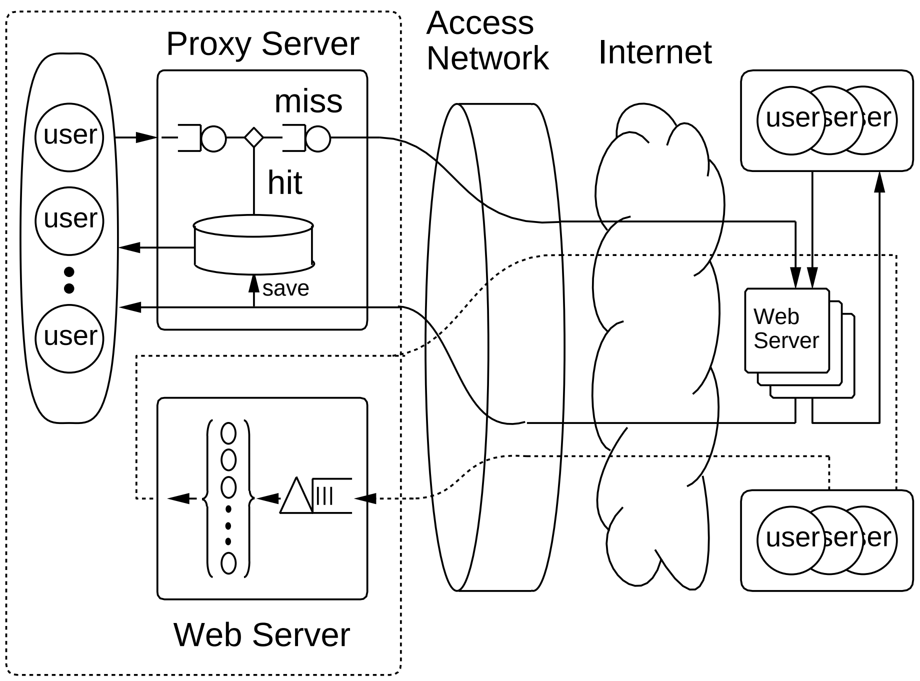

Figure 5.1: The entire Web server system model.

We have discussed with the problems of the resource allocation on the Web system, and only aimed at the document transfer delay as an index of the performance in the numerical examples. Actually, when we construct the system with evaluating quantitatively, it is true to design with considering the cost. In this thesis, we have presented the evaluation examples of the modeling the transmission delay of the document, and have only considered the Web document transfer. It is true that Web traffic dominates the Internet in recent days, but the Web system allows various applications. It is especially important to take account of `cgi` and similar tools since it affects the Web server performance. In the future, we should construct the model with the effect of these applications. Further investigations are necessary regarding this problem.

Bibliography

[1] Network Wizards, "Internet Domain Survey." available at `http://nw.com/zone/WWW`.

[2] M. Nabe, M. Murata, and H. Miyahara, "Analysis and Modeling of World Wide Web Traffic for Capacity Dimensioning for Internet Access Lines," *in Proceedings of SPIE Conference on Performance and Control of Network Systems*, vol. 3231, pp. 2–12, November 1997.

[3] M. Nabe, M. Murata, and H. Miyahara, "Analysis and Modeling of WWW Traffic Characteristics with Document Caching," *The Transactions of the Institute of Electronics, Information, and Communication Engineers of Japan*, vol. J81-B-1, pp. 325–334, May 1998. (in Japanese).

[4] A. Iyengar, E. MacNair, and T. Nguyen, "An Analysis of Web Server Performance," *in Proceedings of GLOBECOM '97*, 1997.

[5] J. C. Hu, I. Pyarali, and D. C. Schmidt, "Measuring the Impact of Event Dispatching and Concurrency Models on Web Server Performance Over High-speed Networks," *in Proceedings of GLOBECOM '97*, 1997.

[6] R. McGrath, "Performance of Several Web Server Platforms." available at `http://www.ncsa.uiuc.edu/InformationServers/Performance/Platforms/report.html`.

[7] N. J. Yeager, and R. E. McGrath, *Web Server Technology*. San Francisco: Morgan Kaufmann Publishers Inc., 1996.

[8] L. P. Slothouber, "A Model of Web Server Performance." available at `http://www.starnine.com/webstar/overview.html`.

[9] R. P. Wooster and M. Abrams, "Proxy Caching that Estimates Page Load Delays," 1997.

[10] C. Maltzahn and K. J. Richardson, "Performance Issues of Enterprise Level Web Proxies," *in Proceedings of ACM SIGMETRICS '97*, 1997.

[11] L. Zhang, S. Floyd, and V. Jacobson, "Adaptive Web Caching," 1997.

[12] Mindcraft, Inc., "WebStone : The Benchmark for Web Server." available at `http://www.mindcraft.com/webstone`.

[13] Standard Performance Evaluation, Co., "SPECweb96 Benchmark." available at `http://www.specbench.org/osg/web96`.

[14] Ziff-Davis, Inc., "WebBench 3.0." available at `http://www.zdnet.com/zdbop/webbench/webbench.html`.

[15] Mindcraft, Inc., "WebStone : Standard WebStone 2.0.1 Run Rules." available at `http://www.mindcraft.com/webstone/ws2.0.1_runprocs.html`.

[16] T. Berners-Lee, R. Fielding, and H. Frystyk, "Hypertext Transfer Protocol – HTTP/1.0 (RFC1945)." available at `http://www.ds.internic.net/rfc1945.txt`.

[17] Y. Fujita, M. Murata, and H. Miyahara, "Analysis of Web Server Performance towards Modeling and Performance Evaluation of Web Systems," *Technical Report of IEICE*, vol. CQ97-12, pp. 77–84, December 1997. (in Japanese).

[18] Y. Fujita, M. Murata, and H. Miyahara, "Performance Modeling and Evaluation of Web Systems," *Technical Report of IEICE*, vol. SSE97-215, pp. 133–138, March 1998. (in Japanese).

[19] Y. Fujita, M. Murata, and H. Miyahara, "Performance Modeling and Evaluation of Web Systems," *in Proceedings of 1998 IEEE Communication Quality and Reliability Workshop*, May 1998.

[20] Y. Fujita, M. Murata, and H. Miyahara, "Analysis of Web Server Performance towards Modeling and Evaluation of Web Systems," *in Proceedings of 1998 IEEE SICON*, pp. 221–224, July 1998.

[21] Y. Fujita, M. Nabe, M. Murata, and H. Miyahara, "Building the Performance Model of Web Systems based on Experimental Benchmark," *in Proceedings of ITC-CSCC '98*, vol. 1, pp. 517–520, July 1998.

[22] Y. Fujita, M. Murata, and H. Miyahara, "Building the Performance Model of Web Server and the Application to Performance Evaluation of Web Systems," *in Proceedings of the First AEARU Workshop on Web Technology*, pp. 91–98, November 1998.

[23] Y. Fujita, M. Murata, and H. Miyahara, "Performance Modeling and Evaluation of Web Server Systems," *The Transactions of the Institute of Electronics, Information and Communication Engineers of Japan*, vol. J82–B, pp. 347–357, March 1999. (in Japanese).

[24] Y. Fujita, M. Murata, and H. Miyahara, "Performance Modeling and Evaluation of Web Systems with Proxy Server Caching," *Technical Report of IEICE*, vol. CQ98-4, pp. 21–28, May 1998. (in Japanese).

[25] Y. Fujita, M. Murata, and H. Miyahara, "Performance Modeling and Evaluation of Web Systems with Proxy Caching," *in Proceedings of ITC-16th*, vol. 3b, pp. 1179–1188, June 1999.

[26] Y. Fujita, M. Murata, and H. Miyahara, "Performance Modeling and Evaluation of Web Systems with Proxy Caching," *The Transactions of the Institute of Electronics, Information and Communication Engineers of Japan*, vol. J82–B, pp. 1449–1461, August 1999. (in Japanese).

[27] Apache HTTP Server Project, "The Apache Software Foundation." available at `http://www.apache.org/`.

[28] Microsoft, Co., "Internet Information Server." available at `http://www.microsoft.com/iis`.

[29] Netscape Communications, Co., "Netscape Enterprise Server 3.0." available at `http://live.netscape.com/comprod/announce/dss_ente.html`.

[30] NCSA HTTPd. available at `http://hoohoo.ncsa.uiuc.edu/`.

[31] CERN httpd. available at `http://www.w3.org/Daemon/`.

[32] Netcraft, Ltd., "The Netcraft Web Server Survey." available at `http://www.netcraft.com/Survey/`.

[33] S. F. Yashkov, "Processor–Sharing Queues : Some Progress in Analysis," *Queueing Systems*, vol. 2, pp. 1–17, 1987.

[34] K. M. Rege and B. Sengupta, "Sojourn Time Distribution in a Multiprogrammed Computer System," *AT&T Technical Journal*, vol. 64, pp. 1077–1090, May–June 1985.

[35] B. Avi-Itzhak and S. Halfin, “Expected Response Times in a Non-Symmetric Time Sharing Queue with a Limited Number of Service Positions,” *in Proceedings of ITC-12*, pp. 1485–1493, 1989.

[36] S. S. Lavenverg, *Computer Performance Modeling Handbook.* New York: Academic Press, 1983.

[37] Squid Internet Object Cache. available at `http://squid.nlanr.net/Squid`.

[38] P. Manzoni and D. Ghosal, “Impact of Mobility on TCP/IP : An Integrated Performance Study,” *IEEE Journal on Selected Areas in Communications*, vol. 13, pp. 858–867, June 1995.

Biography

Yasuyuki Fujita was born in Kyoto, Japan on September 16, 1968. He received the B.E. in Engineering from Doshisha University, Kyoto, Japan, in 1992. In April 1992, he joined the Kansai Electric Power Co., Inc., Japan, where he was engaged in the development and implementation of information systems. In June 1997, he entered the Graduate School of Engineering Science, Osaka University, Osaka, Japan, as a researcher. Since 1999, he has been studying at Osaka University, Japan, as a Ph.D. candidate. His research interests include the communication network design and the Web server system.