WORKLOAD CHARACTERIZATION FOR RESOURCE MANAGEMENT AT WORLD WIDE WEB SERVERS

A Thesis Submitted to the College of

Graduate Studies and Research

in Partial Fulfillment of the Requirements

For the Degree of Master of Science

in the Department of Computer Science

University of Saskatchewan

Saskatoon, Saskatchewan

by

Adeniyi Adebowale Oke

Spring 2001

Permission To Use

Head of the Department of Computer Science
University of Saskatchewan
Saskatoon, Saskatchewan, Canada
S7N 5A9

Abstract

There is a preponderance of evidence that the World Wide Web (or Web) exhibits dramatic growth on the Internet today, despite its non-existence a decade ago. This growth in Web activity has motivated a prodigious amount of research work directed at managing both resources and the performance of the Web.

Central to this thesis research is a new concept called "Web Server Access Hierarchy (WSAH)" which considers the Web server traffic at various access levels of aggregation—as traffic from multiple clients, as traffic from individual clients and as traffic within sessions of individual clients. The primary objective of this research is to characterize the WSAH in an attempt to answer two questions. First, how do the characteristics of Web traffic change over the levels of the access hierarchy? Second, how can this information be used to manage server resources? It is hypothesized that decomposing the server workload into access levels and carrying out a workload characterization study on each of the levels can be beneficial in managing Web server resources effectively.

For the first part, the results from characterizing the aggregate server workload show that some behavioural characteristics of the server workload have changed over time, while others have not. The characteristics of the request stream of individual heavy-access clients show that there is a dramatic difference in the behavioural characteristics of "human" clients and "non-human" clients such as proxies or Web crawlers. Characterizing sessions within request streams of single clients indicates that "human" clients utilize HTTP persistent connections less effectively than "non-human" clients. Moreover, there is overwhelming evidence that phase-transition behaviour is more prominent in the reference patterns of "heavy-access" Web clients at the server. The phase-transition behaviour in the request stream of "non-human" clients is difficult to predict while that of "human" clients can be predicted. In retrospect, the behavioural properties that serve as the fundamental basis for caching strategies are less strong as one moves from the first level of the WSAH to the third level.

The second part of the thesis focuses on a Web server performance study using trace-driven simulation. With global caching, the results show that a SIZE replacement policy (with a frequency-based aging policy for breaking ties) outperforms an LFU-Aging replacement policy to produce the best cache size-to-performance ratio. For partitioned caching, the performance results with document size(s) show some performance improvement over global caching, while the performance results with client Internet Protocol addresses yield no significant performance improvement over global caching. The performance study further reveals the potential value of using client information for effective management of server resources.

Acknowledgements

"Let the name of God be blessed forever and ever, for wisdom and power belong to Him. He gives wisdom to wise men and knowledge to men of understanding[1]*."*

I wish to express my profound gratitude to my able and meticulous supervisor, Professor Rick Bunt for his support and contribution during the course of my study. Without his patience and co-operation it would have been impossible for me to finish this thesis. My thanks also go to Prof. Doug Degenstein (external examiner), Prof. Carey Williamson and Prof. Eric Neufeld for serving as members of the thesis defence committee. Their comments and suggestions have gone a long way to improve the quality of the thesis.

Special thanks go to the Telecommunication Research Laboratories (TRLabs) for providing the funding, as well as a very conducive working environment to carry out the research, with state of the art equipment. Thanks to Jack Hanson (system administrator), Trudy Gall (secretary) and Cliff Klein (president of TRlabs, Saskatoon) for making my stay in TRLabs memorable.

I am indebted to Maureen Desjardins for her motherly advice and love. I will always remember the day you invited me to your house for christmas. Thanks to Greg Oster for his tremendous technical asistance, Jan Thompson for her encourangement and Dr. David Sumner for painstakingly proof-reading my thesis.

I am full of praise to my entire family, most especially my darling wife and daughter, Yetunde and Opeyemi for being a source of inspiration and motivation to complete this thesis on time. Finally, I am forever grateful to my late father, Mr Adewole Oke, who invested so much in my life but died untimely without enjoying the fruit of his labour. I remember those memorable days you sat down with me and taught me to be hardworking, God-fearing and to aim high in life. I am happy to learn all these from you because they were all helpful in completing my thesis. Thank you dad, a million times!

[1] *Zondervan New American Standard Study Bible, Daniel 2: 20-21*

Table Of Contents

List of Tables

List of Figures

Chapter 1

Introduction

There is a preponderance of evidence that the World Wide Web (WWW or Web or W3) [19] [20] [21] exhibits dramatic growth on the Internet today, despite its non-existence a decade ago. The reasons for this growth have been associated with the simplicity of accessing and disseminating information among researchers, educational institutions and commercial organizations on the Web. Other factors include the availability of colourful graphical user-friendly interfaces for navigation and the prevalence of the Transmission Control Protocol/Internet Protocol (TCP/IP) on which the Web runs [10] [40] [82].

The phenomenal growth in Web activity, tagged the "killer application" [18] for the next generation of the Internet, continues to increase the number of hosts[1] and clients on the Internet [58]. This has therefore culminated in over-stretching of existing network infrastructures and systems—a situation where the Web server becomes overloaded with client requests, the network becomes congested with traffic, and the end-users perceive increased latency. In combating these problems, there has been a prodigious amount of research activity directed at managing resources as well as improving the performance and scalability of the Web [4] [5] [10] [11] [25] [26] [36] [64]. These research studies were based on workload characterization that suggest "Web caching" as a promising approach to manage Web resources and to improve its performance.

This research introduces a new concept called "Web Server Access Hierarchy"

[1] The number of hosts on the Internet was estimated to be 72,398,092 as of January 2000 [70].

which considers the levels by which Web servers receive requests from prospective clients: aggregate traffic from multiple clients, traffic from individual clients and traffic within sessions from individual clients. The aim of this research is to characterize the Web server access hierarchy in an attempt to answer two questions. First, how do the characteristics of Web traffic change over the levels of the access hierarchy? Second, how can this information be used to manage server resources? It is hypothesized that decomposing the server workload into access levels and carrying out a workload characterization study on each of the levels can be beneficial in managing Web server resources effectively. In addition, this research will provide insight into the design and provisioning of Web services for performance improvement.

In many respects, the research is similar to [8] [10] [11] [12] [36], but it differs in that workloads from different levels of the server access hierarchy are used to understand how the characteristics change as one moves from one level of access granularity to another at the server. The research ultimately demonstrates how the behavioural characteristics obtained at different levels of access granularity at the server can be useful in managing the server resources as well as improving its performance.

The first part of this thesis focuses on the workload characterization of a typical busy Web server: (i) characterizing the aggregate pattern of service demand from multiple clients; (ii) characterizing the pattern of service demand from individual clients; and (iii) characterizing the patterns of service demand in sessions from individual clients. The results from this part show that a busy server has some peculiar characteristics that make it different from the "normal" server workload characterized in [7] [9] [10]. For instance, there exists a small set of documents (about 5%) in the aggregate server workload that were responsible for 93% of total requests received and 80% of the volume of data transferred by the server. Due to this peculiar characteristic, some behavioural characteristics of the aggregate workload have changed in comparison to the observations reported in [7] [9] [10], while others have not. Table 1.1 summarizes some of the workload characteristics observed. The implication of the aggregate characteristics is that caching strategies can be used

Table 1.1: Summary of Web Server Workload Characteristics

Characteristic	
Success Rate	About 80% of requests are successful, and these account for 98% of the volume of bytes transferred.
File Types	HTML and Image documents account for 96% of valid requests.
Mean Transfer Size	$\approx$ 28 kB
Median Transfer Size	$\approx$ 9 kB
One-Timer Client	$\approx$ 5.36% of total clients at the server. Accesses account for 0.10% of the bytes transferred.
One-Timer Document	About 12% of documents are referenced once. This accounts for 9% of total document disk space and $\approx$ 0.02% of total document requests to the server.
Size Distribution	Document and transfer size distributions are heavy-tailed.
Concentration of References	10% of documents accessed account for 96% of of valid requests and 93% of the bytes transferred.
Inter-Reference Times	Web crawlers can invalidate the measure of temporal locality using inter-reference times. Document inter-reference times are independent and exponentially distributed at 10-minute and 1-hour intervals.

to manage server resources effectively. However, there is a tradeoff in cache size between cache policies that optimize request throughput and cache policies that optimize data throughput. In addition, results show that if a large cache is placed at the server and managed with policies that optimize data throughput, there is no need to determine which of the resources is the bottleneck since both CPU cycles and disk I/Os can be managed effectively at the same time. However, cache polices that optimize request throughput are still the best for managing resources at a busy server because such policies require a relatively small cache size.

For individual client request streams characterized, a few clients at the server are found to be responsible for generating most of the requests at the server. More importantly, the distribution of clients at the server based on both the number of requests and the volume of data transfered by each client follow a Zipf-like distribution. Using four selected clients to understand the nature of reference patterns generated by "heavy-access" clients, the results show that there is a dramatic difference in the behavioural characteristics of “human” clients and “non-human” clients such as proxies or Web crawlers at the server. Among the human clients characterized, the degree of upstream caching is a factor that affects how the behavioural characteristics differ from one another. Mixing the traffic from these two categories of clients may not augur well for the effective management of the server. The implication of these results is that resource management at the server may differ across the categories of clients that are accessing the server.

The characterization of sessions within client workloads reveals several results. A Persistent-Connection HyperText Transfer Protocol (P-HTTP) can be beneficial at the server at higher timeout values of 100 seconds or more, if the server memory is bottleneck free. However, the characteristics of the request stream of “human” clients show that P-HTTP is grossly underutilized at short timeout values while “non-human” clients do not. The implication of this observation is that the utilization of HTTP persistent connections by “human” clients may affect server performance if the server memory is a bottleneck. This is because the server reserves resources to maintain state of TCP connections at the server. Moreover, there is overwhelming evidence that phase-transition behaviour is more prominent in the reference patterns within sessions of “heavy-access” clients. The phase-transition behaviour in the request stream of “non-human” clients is difficult to predict while that of “human” clients can be predicted.

The second part of the thesis deals with the performance of the server, using a trace-driven simulation to demonstrate how the behavioural characteristics at various access levels of Web server traffic aggregation can be useful in managing server resources effectively. The performance study considers two important cache configurations, global caching and partitioned caching along with document placement policies that use document size and client Internet Protocol (IP) address. In addition, the management of selected client request streams that are identified as human and non-human clients are separately managed with the use of appropriate replacement policies. For all the simulation experiments, two frequency-based replacement policies (LFU-Aging and SIZE-F) and two recency-based replacement policies (LRU and SIZE-R) are used to evaluate Web server performance.

With global caching, the performance study shows that a SIZE replacement policy (with a frequency-based aging policy to break ties), outperforms an LFU-Aging replacement policy to produce the best cache size-to-performance ratio at the server. At relatively large cache sizes (above 256 MB), however, there is no significant difference between recency-based and frequency-based replacement policies. When global caching and document size thresholds are combined, there is no performance improvement. For partitioned caching, the performance results with document size(s) reveal some performance improvement over global caching, while the performance results with client IP addresses yield no significant performance improvement over global caching. Managing the request streams of selected clients separately shows there is a relationship between characteristics of client reference patterns and the performance of replacement policies. The implication of this relationship is that a client-based resource management approach at the server can be effective if individual client request streams are managed separately by policies that take account of their inherent properties rather than just relying on client IP address. Therefore, there is an indication that the use of client information at the server can be an effective way of managing the resources at the server.

The remainder of this thesis is organized as follows. Chapter 2 provides background information on the World Wide Web and how the present Web cache configuration has contributed to the nature of server workloads. Chapter 3 reviews the nature of the patterns of service demand and resource management techniques on the Web. Chapter 4 presents the behavioural characteristics of aggregate server workload. Chapter 5 focuses on the behavioural characteristics of client reference patterns within individual client request streams as well as the characteristics of sessions within the request stream of selected "heavy-access" clients. Chapter 6 discusses the results of a performance study conducted for the Web server using a trace-driven simulation approach. Chapter 7 concludes the thesis with a summary of the research including an outline of research contributions and intended future research work. Appendix A is a glossary of terms used in the thesis.

Chapter 2

Background

The dramatic growth of the World Wide Web (or Web) has stimulated research interest in studying its performance and scalability. This chapter is broken down into three sub-sections. Section 2.1 provides some background into the architecture of the Web with the purpose of describing how servers, proxies, accelerators, and clients operate. Section 2.2 describes the cache hierarchy configuration as it applies to Web server workloads. Section 2.3 summarizes the chapter.

2.1 World Wide Web

The World Wide Web is a distributed information access and retrieval system developed at CERN, the European Center for Nuclear Research, in Geneva, Switzerland [19] [21] [58] [85]. The Web initiative is a vision to bring a global information universe into existence using hypertext technology on the Internet. Hypertext describes the technology underlying the links between pieces of information in text form (or other media) in a way that mimics human association of ideas [20]. These pieces of information are transferred in the form of Web pages or documents. Figure 2.1 depicts a simple architectural model of the Web. The framework is based on the client-server model. The server is the response node where information requested is stored in a Web format called HyperText Markup Language (HTML). The client is a request-initiating node, a program that enables the user to view and navigate the contents of the servers as a single, integrated system. The model (depicted in Figure 2.1) works in such a way that the Web user initiates a transaction via the Web client, popularly known as a browser—a program used to view and navigate Web documents. The Web browser sends the transaction request to the Web server via a communication protocol called HyperText Transfer Protocol (HTTP). This protocol runs on top of the TCP/IP protocol. The Web server receives transaction request(s) and responds to the Web client by obtaining the appropriate document. This service model treats all transactions equally according to the Internet's best-effort service.

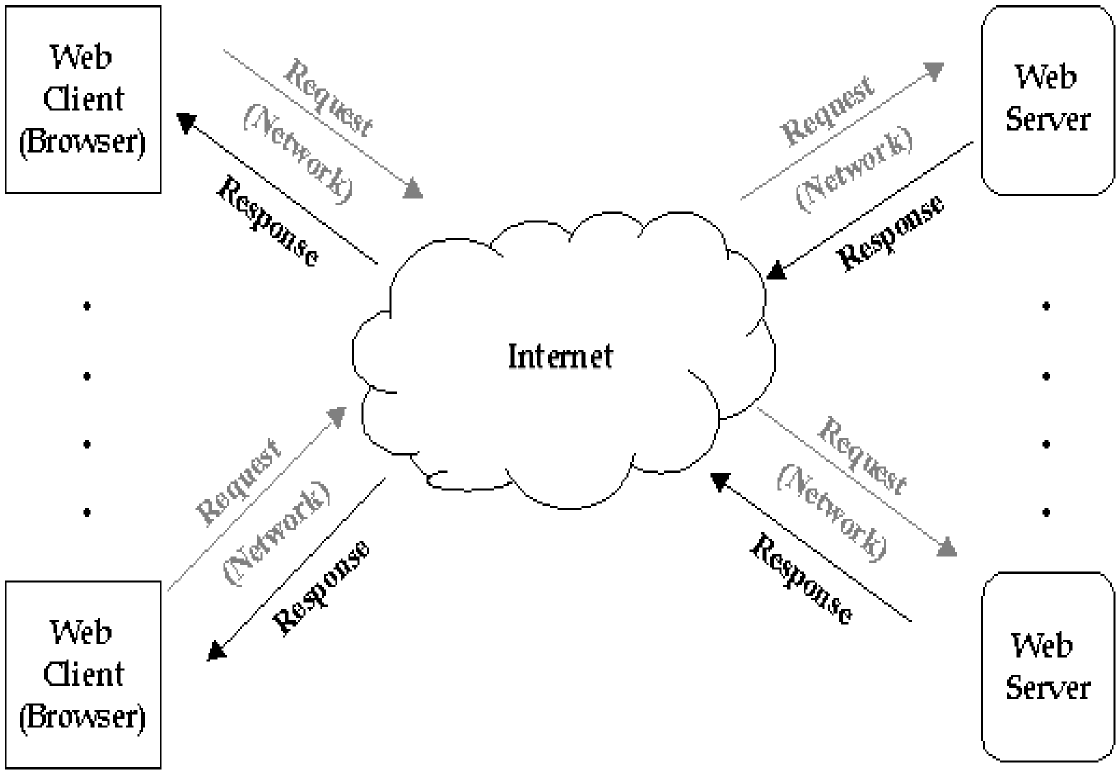


Figure 2.1: World Wide Web Client-Server Architecture

Communication between the client and server is established through a TCP connection before the request and response can be exchanged between them. Once the transport connection is opened, the client request is passed on to the server, and the server parses the request and sends a response back to the client. On completion of this transaction, the TCP connection is closed to make way for another client transaction. Recently, a new version of the HTTP protocol (HTTP/1.1) has implemented persistent connection, a situation in which one single TCP connection can be used to satisfy several transaction requests from the same Web client [44] [65].

2.1.1 The Web Client

A Web client is defined as a program that establishes connections for the purpose of sending requests. The client can take many forms, including user agents, browsers, editors, and spiders (or Web-traversing robots) [44]. Two popular Web browsers are Microsoft Internet Explorer and Netscape Navigator. The general format of a request message is of the form:

Client Request = Request-Line
[General-header|Request-header|Entity-header]
CRLF
[Message-body].

The request line begins with the method (as shown in Table 2.1), which specifies the action requested, followed by a Uniform Resource Locator (URL)[1] that serves as the document's world-wide name, and the HTTP protocol version to be used for communication. General-header fields provide additional information for both request and response messages, but do not apply to the entity being transferred. They apply only to the message being transmitted. The Request-header fields allow the client to pass additional information about the request and itself to the server. Unrecognized header fields are treated as Entity-header and they are optional. For example, the Request-header can be used to authenticate clients with servers, to indicate the data formats that the client will accept, to indicate what character sets or content coding are acceptable for the response and so forth. The entity header can also be used to list the set of methods supported by the resource identified by the request URL. CRLF is an acronym for Carriage Return and Line Feed which is

[1] A Uniform Resource Locator refers to the format used by Web documents to locate other objects and is currently the standard for specifying an object or resource on the Internet [85]. The format is: *scheme://host.domain[:port]/path/filename.*

Table 2.1: The Built-in HTTP/ 1.0 Request Methods

Method	Description
GET	Request to read the Web page
HEAD	Request to read a Web page's header
PUT	Request to store a Web page
POST	Append to a named resource (e.g., a Web page)
DELETE	Remove the Web page
LINK	Connect two existing resources
UNLINK	Break an existing connection between two resources

used to demarcate the request line from the message body.

For instance, URL http://www.cs.usask.ca/grads/aao804/index.html is a typical example of a client request through the browser. This URL can be broken into three parts: the name of the protocol (http), the name of the machine where the document or page is located (www.cs.usask.ca) and the name of the document or page (grads/aao804/index.html). This request would result in the following message being sent by the client:

GET /grads/aao804/index.html HTTP/1.0

Date: Tue, 15 Nov 1999 08:12:31 GMT

Referrer: http//www.cs.usask.ca/

User-Agent: Netscape/1.1

Accept: */*

In this example, the request line specifies that the client has asked the Web server (www.cs.usask.ca) for the document grads/aao804/index.html, using HTTP version 1.0. The date header specifies the date and the time the request originated from the client. The referrer header indicates the address (URL) of the resource from which /grads/aao804/index.html was obtained. It also allows a server to generate lists of back links to resources for interest, logging, and caching. The user-agent header notifies the server that the client is using a Netscape browser version 1.1. The Accept header indicates that all media types are acceptable for response. Browsers work by determining the URL (by seeing what was selected). They then ask a Domain

Name Service (DNS) for the Internet Protocol (IP) address of www.cs.usask.ca. The DNS will reply with the appropriate IP address, after which the browser makes a TCP connection to port 80 on the identified IP address node (server). It then sends a GET /grads/aao804/index.html command after which the server sends the document index.html. The TCP connection is then released and the browser displays all the text in index.html. The browser also fetches and displays all the in-line images in index.html.

Web browsers can also be used to activate information agents, which act as clients that directly request documents from a server. Information agents are deployed on the Web to act as a mediator between a browser client and a server. Their presence are meant to enhance the capability of Web browsers in searching accurately for information on the Web [49]. An information agent (also called a Web crawler[2]) is very common on the Web and can visit a large number of documents in a very short period of time [10].

2.1.2 The Web Server

A Web server is defined as an application program that accepts connections in order to service requests by sending back responses [44]. Examples of Web servers are Apache [6], CERN-httpd [55], Netscape Enterprise [69], and Zeus [89]. Web servers have a process listening to TCP port 80 for incoming connections from clients. Once the client has established a connection with the server, the server must respond to the request to complete the two-way communication behaviour as dictated by the Web model. The response [44] [82] includes a status code to inform the client if the request was successful or otherwise give a reason via the code why the request was not granted. Other additional header information is returned by the server in answer to the client's requests included in the header fields. A standard format for a response is:

Server Response = Status-Line

[2]A Web crawler is a program that retrieves Web pages, commonly for use by a search engine. It runs several processes at a time, which open several hundred connections that access Web pages.

Table 2.2: Categorization of Status Response Codes and Descriptions

Status Code	Description
1xx	Informational— Request received, continuing process
2xx	Success—The action was successfully received, understood, and accepted
3xx	Redirection—Further action must be taken in order to complete the request
4xx	Client Error—The request contains bad syntax or cannot be fulfilled
5xx	Server Error—The server failed to fulfill an apparently valid request

[General-header|Response-header|Entity-header]

CRLF

[Message-body].

The Status-Line consists of the protocol version, followed by a 3-digit numeric status code and its associated textual phrase intended to give a short description of the status code. The first digit of the status code defines the class of response. Table 2.2 shows five classes of response codes based on the first digit as described by [44]. The server must be able to understand the class of any status code, as indicated by the first digit. Any unrecognized response is treated as x00 status code. The response header allows the server to pass additional information about the response, which cannot be included in the Status-Line. Such information includes that of the server and about further access to the resource identified by the request-URL. The entity-header fields define meta-information about the requested resource. The following is an example of a server response:

HTTP/1.0 200 OK

MIME-version: 1.0

Date: Tue, 20 Nov 1999 10:12:33 GMT

Server: CERN/3.0

Content-Type: Text/HTML

Content-Length: 8247

In this example, the Status-Line specifies the protocol being used as the same as the client (HTTP/1.0) and the response code 200 falls into the second class meaning successful. The description of the status code 200 is OK. The date and the time of response, as well as the type of the server (CERN server, version 3.0) are indicated. The content-length indicates that the size of the document transferred is 8247 bytes and the content-type header (encoded using Multipurpose Internet Mail Extension (MIME) version 1.0 format) identifies the document to be Text/HTML. The MIME types allow arbitrary documents to be sent in a standard way. The TCP/IP transport mechanism does not require the MIME content-transfer-encoding header because all the byte streams including pictures are sent without modification [82].

2.1.3 The Web Proxy Server

A Web proxy server (or "proxy" for short) [60] is an intermediary program which acts as both a server and a client for the purpose of making requests on behalf of other clients [44]. The initial use of proxies was to provide access to the Web for people on closed subnets (or Intranet) who access the Internet through a firewall machine. Today, proxies are also used for the efficient caching of remote documents that are requested by a large number of clients. The idea behind caching proxies is to speed up Web access without requiring more bandwidth so that end users enjoy faster response times [48]. It is even possible to have proxies that combine a firewall function of protecting clients from outside attack on their network installations and facilities, with the performance function of caching documents closer to Web users.

Proxies can be viewed from two important perspectives. First, a proxy can be viewed as a server if an incoming document request through the browser is being serviced from its backing store. Second, when the proxy fails to service an incoming document request from its backing store, its behaviour becomes that of a browser or client sending direct requests to the origin server. Once the origin server responds, the proxy stores the document requested on disk and relays back the origin server response to the origin client or browser. Figure 2.2 depicts the configuration of the Web architecture with the inclusion of a proxy. Setting up a proxy server requires a browser that already has proxy support built in. Once the browser is configured to point at the proxy, requests that come from this browser would be cached and responded to by the proxy. However, if the proxy server is down, the client may reconfigure its browser to gain access to the outside world. With transparent configuration, a router or Local Area Network (LAN) switch inspects headers and redirects HTTP traffic to the proxy server without the browser pointing at the proxy server. This mechanism allows the router and switch to perform automatic redirection of client requests that mask the proxy from browsers. Recently, new techniques have evolved where there is no need to configure Web browsers with proxy addresses or to command routers to redirect HTTP packets [48].

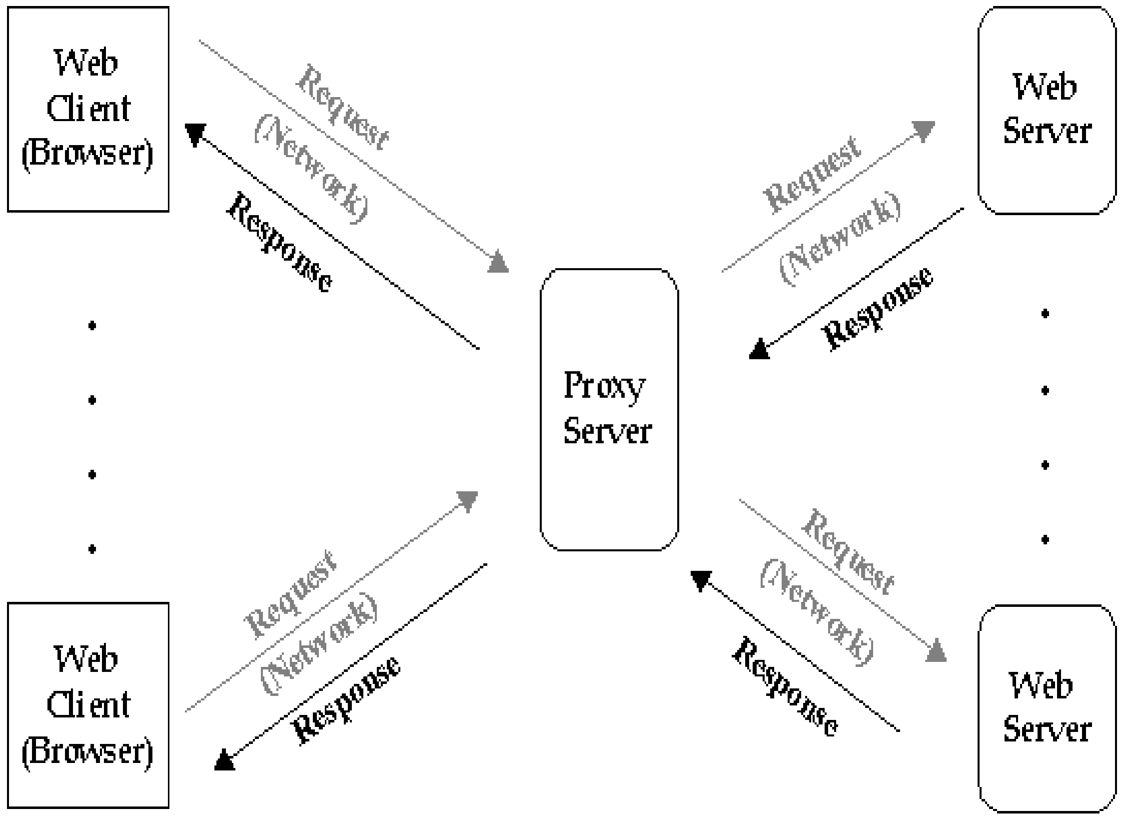


Figure 2.2: World Wide Web Proxy Configuration

The usefulness of the proxy stems from the fact that a single Web client that is linked to a proxy server can handle other protocols such as FTP [60] [78]. This makes it possible to have a "lightweight" client that understands only the HTTP protocol. In addition, clients without a Domain Name Service (DNS) can still access the Web in as much as the browser has been configured to point at a proxy server. Caching proxies [4] also enable clients to surf the Web even when the remote server can no longer be reached, hence enhancing the quality of service of remote network resources. The deployment of Web proxies on the Web, however, can create additional performance problems [48]. Nevertheless, the ability of proxies to conserve bandwidth and reduce server load to enhance the performance and scalability of the Web justify their widespread use. Notable Web proxies include the CERN-http server [60], Dynacache [48] and Squid [79].

Another type of proxy server is known as an *accelerator* [59]. The primary purpose of an accelerator is to speed up access to a local Web site, whereas the primary purpose of a cache proxy is to speed up access to remote Web sites by storing data from the remote sites. This is achieved by allowing clients within a corporate Intranet to point to the proxy to facilitate easy access to remote documents. In this case, it is assumed that most of the clients share most of the remote documents.

2.2 Web Cache Hierarchy Configuration

Web caching has been strongly influenced by traditional caching approaches. For example, caching in memory systems was designed to deal with speed differences between the levels in a storage hierarchy, while caching in the Web is meant to reduce end-user response times by reducing network traffic and server loads. Presently, caches have been placed in all the three possible locations in the Web: at the client [22] [24], in the network [4] and at the server [64]. Repeated requests for popular documents from individual Web users are resolved in cache at the client side. The requests that find their way into the network are referred to as first level cache misses. The aggregated stream of requests from individual clients is passed into local and regional networks. However, in order to conserve network bandwidth and reduce server load, regional or geographical caching in the network has been added. A caching proxy [4] is an example of a server deployed in the network to cache missing documents from many clients. A caching proxy stores copies of Web documents on its disk closer to the requesting clients and caches and replicates popular documents by controlling servers that "push or pull". Caching in the network seems to be a good option because many client requests can be served and recently cached objects can be shared among these clients. In this way, the end-user perceives an improvement in latency. Caching proxies in the network further filter most sharable documents that emanate as first-level cache misses. The remaining few document requests that manage to reach the server from various clients are known as second-level cache misses (if the requests pass through the proxy). These cache misses at the server comprise dynamic requests that are not cached, first-time document requests, Web crawler document requests, proxy requests (or firewall requests) and documents which have been replaced in the upstream caches. Figure 2.3 shows the cache hierarchy configuration of managing Web resources to improve the overall performance and scalability of the Web.

2.3 Summary

In this chapter, some background information on World Wide Web was presented. The architecture of the Web with regard to the client, proxy and server was described, and the interaction that exists among the three components was explained. The nature of server workloads was discussed using the hierarchical configuration of Web caches.

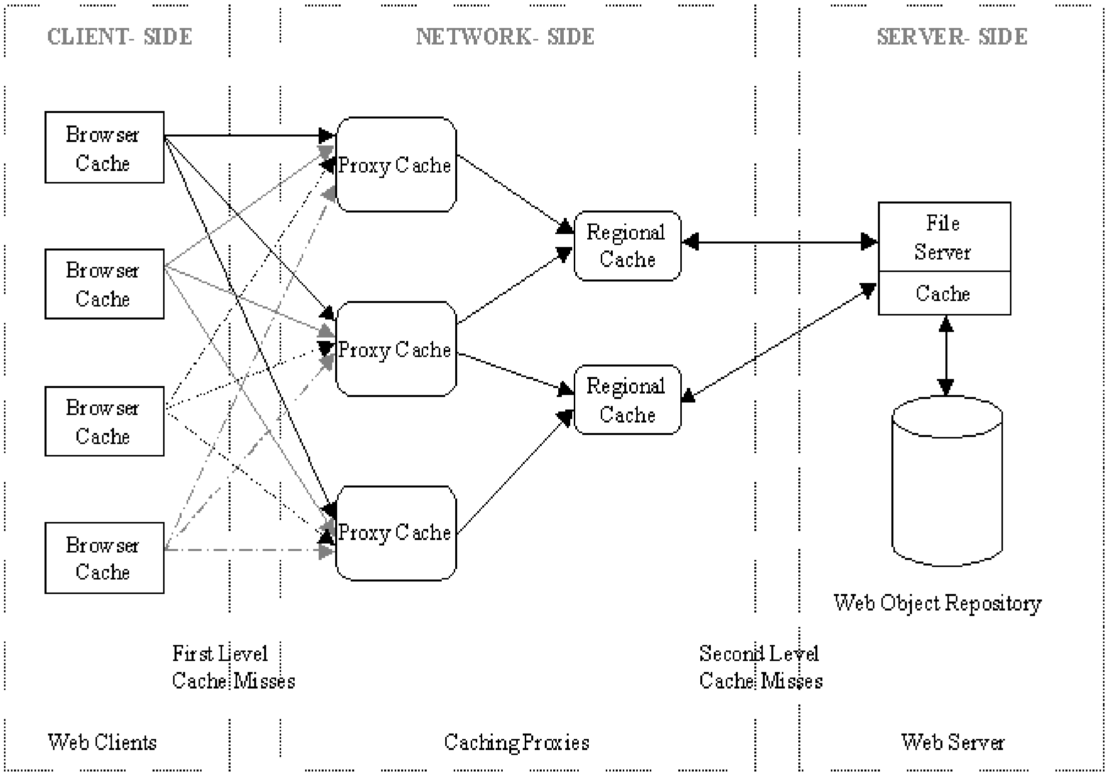


Figure 2.3: Three-Level Cache Hierarchy Configuration of the World Wide Web

Chapter 3

World Wide Web Resource Management

Since the performance of the World Wide Web depends on the combined effectiveness of managing Web resources at the client, in the network and at the server, many strategies have been devised to improve these individual components. This research aims to understand the behavioural characteristics of workloads at different levels of granularity of Web server accesses and to suggest means of managing server resources accordingly. The Web behavioural characteristics are useful in managing Web resources because the knowledge resulting from characterizing the Web workloads at these three levels is helpful in designing appropriate resource management policies.

Section 3.1 describes the issues that relate to understanding Web server resource management. Section 3.2 reviews related work on behavioural characteristics of Web workloads and the resulting resource management strategies. Section 3.3 summarizes the chapter.

3.1 Understanding Server Resource Management

Web server resource management seeks to ensure that the currently available server resources are used efficiently in order to obtain the highest attainable performance. An important step to achieve this objective is often by collecting the server access log and carrying out a detailed characterization. The focus of such a characterization is to understand the service demand for the system's resources and to suggest policies that can effectively handle the management of the resources, as well as the performance issues. Web workloads have been studied at the client [15] [34] [36], at the proxy [2] [4] [61] and at the Web server [10] [11] [29]. The effectiveness of policies that emanate from these characterizations, however, is only as good as the ability to estimate accurately the behavioural characteristics of the workload.

3.2 Web Related Work

This section reviews the behavioural characteristics of document reference patterns and the various approaches to managing Web resources at the client, proxy and server. Having described the work on managing resources in the Web, new design techniques that can manage the server resources such as documents, disk, processor and cache will be suggested with the intention of improving the overall response times. Section 3.2.1 reviews the behavioural characteristics of Web reference patterns for design and provisioning of Web services. Section 3.2.2 reviews the various approaches to managing Web resources at the client, proxy and server with respect to Web caching. Section 3.2.3 describes some other strategies of managing Web server resources apart from caching and their performance implications.

3.2.1 The Nature of Web Document Reference Patterns

The first attempt to characterize Web client behaviour was conducted at the Georgia Institute of Technology [30] for the purpose of understanding user navigation strategies as well as to provide real data on interface usage. Web user logs were captured for three weeks using a NCSA XMosaic browser. The results established 25.5 minutes as the time that separates two user sessions and categorized user navigation patterns as serendipitous (random), browsing (looking for items of interest) and searching (investigating specific topics). These client-based request patterns were very useful in establishing design and usability guidelines for Web documents, sites and browsers.

The Boston University Oceans group conducted a much larger study over a period of six months, collecting client access logs by instrumenting a NCSA Mosaic browser [36]. The group's interest was user-level performance metrics such as service time and access patterns in the Web. Two types of user modes were identified as "mostly surfing users" and "mostly working users". The results showed that Web users' demands could be modeled by a heavy-tailed distribution. In addition, the implication of this distribution with respect to how it can be used to guide system designers in the provision of efficient Web services at the client side was also discussed. In more recent work, Barford *et al.* [15] investigated how the salient characteristics of Web users' workloads may be changing with time. The method used to capture client access logs was different from earlier work [36]. Since the browser used could not be modified to avoid any form of caching before the collection of the access log at the client, the user requests that were seen after caching were recorded in the access log. This approach allowed the effectiveness of client caching on Web users' workloads over time to be examined. The results showed that documents that were recently examined are likely to be requested again in the near future (i.e. temporal locality) and that a small number of documents are extremely popular (Zipf distribution). However, these characteristics have been weakened in Web users' reference patterns. Efficient browser caching, changes in Web resources, changes in user population (e.g. more diverse interests) and more sophisticated usage of the Web were reported to contribute to the changes.

Hine *et al.* [50] characterized client behaviour based on the number of client accesses to a server within a single client browsing session. The method used was to distinguish clients by unique machine identifiers; thus, machines supporting multiple browser clients such as a proxy cache are treated as a single client. To detect dependencies between resources at a server, the probability that the time between successive accesses is less than a threshold window time in the range of 5 to 120 seconds was estimated. A set of client browsing modes was observed, which were termed Wanderer (client "bouncing" between sites with few requests made to one server), Sojourner (client requests are not concentrated on any particular site, but the use is not quickly bouncing), and Resident (client requests made to a single server such as reading material on a digital library). Abdulla *et al.* [3] studied client[1] accesses to Web Information Retrieval Systems—special search tools and cataloging systems developed to help users locate information on the Web. Using several server logs, a timeout[2] of 5 minutes between document requests was defined for identifying client sessions, even during the busiest hours. The results of the characterization showed that increased searching corresponds to reduced network and server traffic.

Web traffic in the network was characterized to understand the stress placed on the network due to exponential traffic growth. Crovella and Bestavros [35] investigated the presence of self-similarity behaviour in client-generated Web traffic with a view to explaining the causes of the behaviour. The data sets collected in [36] were used to examine the nature of Web traffic. The results showed the dominant behaviour of a heavy-tailed distribution characterizing Web transmission times, Web silent times and Web file sizes as the cause of self-similarity. Self-similar behaviour has also been observed in Web server workloads [5] [7] [9].

An in-depth characterization of Web server workload was carried out at the University of Saskatchewan [7] [9] [10]. The approach employed six Web server access logs collected from academic environments, scientific research institutions and commercial Internet providers, with periods ranging from one week to one year. The goal of the research was to search for workload invariants across the six data sets. The results identified ten invariant characteristics of Web server workloads. The document reference pattern across the six workloads showed high concentration of reference, with 10% of the documents accounting for 90% of server requests and bytes transferred. Another invariant characteristic was that Web servers are accessed by thousands of domains (Web clients) with 10% of the Web clients accounting for more than 75% of usage. Temporal locality of document reference, heavy-tailed

[1] A client in this case, refers to several users running multiple instances of a browser on a multi-user machine. This differs from one instance of a browser running on a single machine like PCs with Windows 95.

[2] A timeout is a threshold window time between successive requests from a single client, which is used for classifying requests into client sessions.

distribution of document sizes, and Zipf distribution of document popularity were also observed.

Arlitt and Jin [11] [12] and Arlitt [8] carried out similar workload characterization of the 1998 World Cup Web site to determine how Web server workloads are evolving as the Web increases in popularity and utilizes new technologies. The data were collected over a three-month period and 1.4 billion requests from clients made this Web server workload the largest to be characterized to that point in time. The work analyzed the usage of the World Cup site by characterizing (i) the aggregated workload from multiple clients; (ii) the workload presented by individual clients; and (iii) the busy segments of the workload. The analysis was undertaken to understand changes in the behavioural characteristics of the server workload, the effectiveness of newly introduced persistent connections in HTTP/1.1 and the changes, if any, occurring to the aggregate workload characteristics when the traffic is exceptionally heavy. The results showed that (i) improvements in the caching architecture of the Web have changed some of the Web server characteristics; (ii) a significant number of user sessions (17% for a timeout of 100 seconds) within individual client traffic contained only a single request during the lifetime of the session and a fixed timeout for opening a persistent connection is ineffective; and (iii) the volume of cache consistency traffic increased dramatically when the server was extremely busy, meaning that lack of an effective cache consistency mechanism prevents Web servers from fully benefiting from upstream caching. The approach used by [8] [10] [11] [12] was similar to this research but differs because of the decomposition of Web server traffic into different levels of access granularity.

In addition to characterizing real server workloads, artificial server workloads have been developed more recently using analytical modeling [40] [52] [80] and synthetic workload generation tools [16]. These approaches are used in research seeking to understand how Web servers and networks respond to variation in load requests from multiple Web clients. Other research has also focused on workload characterization of Web proxy document reference patterns as a step required to manage resources in a network [2] [4] [27] [28] [41] [46] [61].

3.2.2 Web Client, Proxy and Server Resource Management

Three approaches have been popularized in Web research as a way to manage Web resources with the ultimate goal of improving performance. The approaches focus on how to reduce the latency of Web document access, the network bandwidth demand of document transfers, and the demand for document transfer at the server. Document caching and replication techniques have been widely used by Web researchers interested in improving the scalability and performance of Web resource management.

Bestavros *et al.* [23] conducted research at Boston University on the potential of document caching. The work included a simulation study of the Web at three levels of caching granularity. These were session-level caching (a single user on a single workstation), host-level caching (several users on a single workstation) and network-level caching (multiple co-operative users on a single LAN). The results showed that session-level caching performs better when the cache size is small while LAN caching performs better when the cache size is large. Host-level caching did not show a significant performance improvement when compared with session-level caching, but, if the resource requirements of the three caching levels are considered, caching at the LAN level is the preferred option.

In order to minimize latency, without restricting the user via Web browsers, browser caching has been used to manage resources at the client. Browser or client caching manages Web resources through client-initiated or server-initiated speculative service. A client-initiated service involves the use of a caching policy in which a document is cached after it is first retrieved. On the other hand, a speculative server-initiated service involves the use of a caching policy in which a server (or proxy) responds to a client's request by retrieving other documents likely to be requested by the client in the near future. In general, speculative caching has been termed "prefetching". Prefetching documents is very useful in hiding network and server latency, although at a cost of introducing additional traffic into the network. Speculative caching was investigated at the client [24] based on statistical information maintained at the server for each document serviced. The research showed that prefetching can improve performance. Despite the additional traffic being injected into the network as a result of prefetching, Bestavros [22] found that 5% extra bandwidth results in a 30% reduction in server load and a 23% reduction in latency. Padmanabhan and Mogul [72] investigated a client-initiated service scheme where the server observes the pattern of accesses from several clients and uses this observation to make predictions. This scheme did not really reduce the retrieval latency of document accesses, but it significantly reduced the overall response times perceived by users at a cost of increasing network traffic. The only drawback of managing resources at the client through prefetching (either by client-initiated or server-initiated schemes) is that these approaches are very effective for "frequently traversed documents" but ineffective for "newly/rarely traversed documents".

Recent studies [29] [74] have shown the potential of managing resources in a clustered server environment—an environment of transparently connected multiple servers that appear to the outside world as a single server. A clustered server architecture provides scalability, which enables more resources (servers) to be deployed at the server side to handle the high volume of document requests in a manner that yields a proportional improvement in performance. The key to successfully managing resources in a clustered environment depends on how loads are distributed and balanced among the individual servers that make up the architecture. Pai *et al.* [74] presented a Locality-Aware Request Distribution (LARD) technique by caching at the individual servers in a cluster. With LARD, the front-end server distributes incoming requests from clients in a manner that achieves high locality at the back-ends' memory caches as well as load balancing. The placement of a cache inside each back-end server's main memory reduces latency, and having several servers achieves scalability. Kwan *et al.* [58] earlier carried out similar research using four servers that appeared to the outside world as a single server. A distributed caching of documents that enables cooperative interaction among the back-end servers was used. Bunt *et al.* [29] studied the use of cache affinity in load distribution and balancing of Web requests in a clustered server.

Apart from adding more resources at the server side, caching has also been suggested in various forms as an effective resource management strategy in a single server environment. For instance, Markatos [64] proposed dynamic caching of documents in a server's main memory to reduce the overhead imposed as a result of frequent document requests via disk accesses at the server. The technique works by adjusting to the client access pattern and cache size at the server. A trace-driven simulation showed that 60% of the documents requested were serviced from 512 kB of the server's main memory. Iyengar *et al.* [52] analyzed the performance of Web servers using simulations that incorporate input parameters from real systems. The results showed that the CPU time spent to service dynamic documents under heavy server loads is almost 10 times larger than that for static documents on many platforms for simple Common Gateway Interface (CGI) programs[3].

Iyengar and Challenger [51] demonstrate that caching dynamic documents can reduce CPU load on a server and can increase the hit ratio at the server to around 80%[4]. In contrast, Tatarinov *et al.* [83] suggested static caching as an effective resource management technique to reduce Web server CPU overhead and memory fragmentation. Static caching means that the set of documents in the cache is determined periodically by analyzing the request log file for the previous period. A trace-driven simulation showed that the strategy is good for busy Web servers that store a large amount of rarely modified documents. Web content adaptation has also been suggested to improve the server performance when the CPU bandwidth and network communication link-capacity are saturated [1]. The approach operates by voluntarily rejecting requests that are meant for documents which are time-consuming to service by the server. With this approach, less time-consuming documents such as HTML and text can be accessed more quickly at the server depending on the size of the client's cache and the length of the client's session.

Another approach to managing resources at the server is to cache incoming

[3]Common Gateway Interface programs are used to invoke server programs.

[4]This performance was achieved when the dynamic Web cache was deployed to support the official Internet Web Site for the 1996 Atlanta Olympic games.

documents at a location closer to some community of clients [25] [26]. This reduces the number of requests to the server, and hence, facilitates quick response from the server. In addition, network bandwidth is conserved which reduces the possibility for network congestion. Gwertzman and Seltzer [47] suggest geographical push caching, in which the server chooses to replicate documents as a function of observed traffic patterns. Braun and Claffy [26] showed that geographically distributed caching can save 40% to 70% of the bytes transferred from central servers. Zhang *et al.* [90] also suggested an adaptive caching technique for servicing different groups of users. The strategy allows documents that are accessed by multiple users to be fetched once from the origin server and forwarded via a multicast tree to all the interested users. This means the higher the demand for a document, the closer to end-users the document would be cached, and the more copies would be made available. Much research has suggested the need to manage resources in the network using caching proxies [4]. These caching proxies are managed by policies such as hierarchical caching [31], hierarchical collaborative caching [88], partitioned caching [67], and Space-Time-Working-Set (STWS) [75].

Modification of the HTTP/1.0 protocol has also been suggested as an effective way of reducing the latency perceived by Web users. Since retrieval of a complete Web document requires separate connection requests for text and embedded images, frequent connection setup and tear-down costs for each document request burden servers, and cause the latency for retrieving Web documents to be high. Padmanabhan and Mogul [71] conducted experiments to quantify the cost of using TCP as HTTP's transport mechanism. The results showed that for a typical request-response, the throughput for a short response was as small as 10% of the throughput obtainable by bulk data transfer under similar network conditions. The poor throughput for short responses was traced to the overhead of TCP's connection setup and the slow-start algorithm that was designed to optimize usage for large-scale bulk data transport. Padmanabhan and Mogul [71] proposed a "Persistent-Connection" HTTP or P-HTTP, a variant of HTTP that opens or caches one TCP connection between the client and server, to carry multiple HTTP requests for latency reduction.

The results showed that P-HTTP reduces the overhead cost attributed to TCP. This approach is referred to as "caching in HTTP" [44]. Mogul [65] investigated the performance implications of P-HTTP using trace-driven simulation. The outcome of the experiment was that P-HTTP could avoid these setup costs and thereby achieve latency reduction between 5% and 15%. Pipelining is another resource management mechanism added to the newly released HTTP protocol version 1.1 to reduce the need for the server to acknowledge responses to client requests [44]. It allows Web users to send multiple requests without waiting for individual acknowledgments. Barford and Crovella [17] showed that HTTP 1.1 performed better than HTTP 1.0 if the CPU and network are bottlenecks. However, if disk I/O is the bottleneck, then HTTP 1.0 performed better than HTTP 1.1. A more technical comparison of the two protocols can be found in [57] [87].

3.2.3 Other Strategies for Managing Resources at the Web Server

Several other techniques have been developed to manage resources at the Web server. Eggert and Heidemann [42] proposed a method in which different levels of services are provided to client requests at the server. This implies that the "best effort only" service model employed by the Web is no longer desirable. With two different levels of services (regular and low priority), simulation experiments showed that servicing low-priority requests in the background was not sufficient to saturate the network. Thus, user-level mechanisms such as limiting the size of the request pool, lowering priorities of requests, and limiting transmission rate were found to be able to provide different service classes, even in the absence of operating system and network support. Similar work on how to manage server resources under heavy loads can be found in [14] [52] [68].

3.3 Summary

This chapter reviewed related work on workload characterization of the Web, as well as strategies for managing Web resources at the client, proxy and server. Web caching as a strategy for Web resource management and its performance implications at the client, in the network and at the server were reviewed. The focus of this research, however, is to understand the behavioural characteristics of the *Web server access hierarchy* (WSAH)—aggregate traffic from multiple clients, traffic from individual clients and traffic within sessions of individual clients. The analysis of these behaviours will provide insight into the pattern of service demands for a typical busy server and how to manage the resources at the server effectively.

Chapter 4

Web Server Workload Characterization

This chapter provides a detailed Web server workload characterization from the perspective of the aggregate traffic from multiple clients. Section 4.1 describes how the server workload was collected, reduced and analyzed to obtain useful information about the service demand. Section 4.2 presents document and transfer size distributions. It also includes the procedure for modeling document size distribution at the server. Section 4.3 focuses on the aggregate characterization of document reference pattern and Section 4.4 discusses the issues relating to document caching and resource management at the server. Section 4.5 summarizes the chapter.

4.1 Workload Collection, Reduction and Analysis

This section presents a statistical overview of the server workload used in the study and summarizes the server access log collected. It explains where and how the server access log was collected, and the type of software and hardware used, including an entry description of the server access log. Other items include separate analyses of the raw and reduced server workloads, document types and sizes and one-timer documents and clients, respectively.

4.1.1 Web Server Access Log

A Web server access log was obtained from a busy (anonymous) commercial Internet Web site located in the USA. At this location, a small number of Web servers appear to the world as a single machine, with each server having HP K-class multi-processor HP-UX boxes running the Open-Market-Secure-Web Server/2.0.5.RCO. The server access logs were collected over a seven-day period from Sunday June 14, 1998 to Saturday June 20, 1998, with logs being transferred and archived every 12 hours. Thereafter, the individual 12-hour archived logs were concatenated, preprocessed to preserve the anonymity of users, clients and URLs, and compressed into a one-gigabyte file.

An entry in the compressed server access log has the following format:
[Timestamp] [Elapsed] [URL] [ClientID] [Size] [ServerID] [Content] [Code] [Method]

- Timestamp: The time when the request is completed (socket closed), specified in seconds, since January 1, 1970 with microsecond resolution. In the server access log collected, the timestamp was preprocessed the midnight before the starting date of the collection by offsetting the timestamp from zero seconds.

- Elapsed: The time between the accept( ) and close( ) of the server socket, usually in milliseconds. This is approximated to the nearest second in the access log used.

- URL: The URL of the requested document. It is represented as unique numerical values in the access log.

- ClientID: The IP address of the connecting client. It is represented as unique numerical values in the access log.

- Size: The amount of data written to the client in bytes.

- ServerID: The hostname of the machine where the requested document was obtained.

- Content: The content type of the document specified using the MIME format.

- Code: The status code in the HTTP reply sent to the client in response to its request.

- Method: The HTTP method employed by the client.

Of all the fields represented above, only the Elapsed field was exempted from the analysis.

4.1.2 Raw Workload Analysis

The raw server access log contained information on 79,747,430 requests communicated to the server by 644,187 Web clients[1] over a period of seven days. These requests were responsible for 328 GB of data transfer across the network. The level of server activity represents two orders of magnitude more workload than the heaviest server activity characterized in [10] (tens of millions of requests per day, compared to hundreds of thousands of requests per day). The raw data have 104,575 unique documents that constitute a large working set of documents requested by the clients, and occupy 2,819 MB of storage space. The average number of requests received per day was 11,392,489, while the average daily transfer was 47 GB. The median of requests and transfer sizes per day were 13,565,362 and 55 GB, respectively. Since the medians exceed the means, it implies that weekend effects are present in the raw workload collected—the server was busier during working days than on weekends. Figure 4.1 illustrates the weekend effect in the traffic profile of the trace. The workload shows that the server is busiest during working hours, less busy in the evening and nighttime hours, and least busy on weekends. In addition, a high coefficient of variation of 9.74 for unique document size shows that some documents were very small while others were extremely large. Table 4.1 gives summary statistical information on the raw server workload.

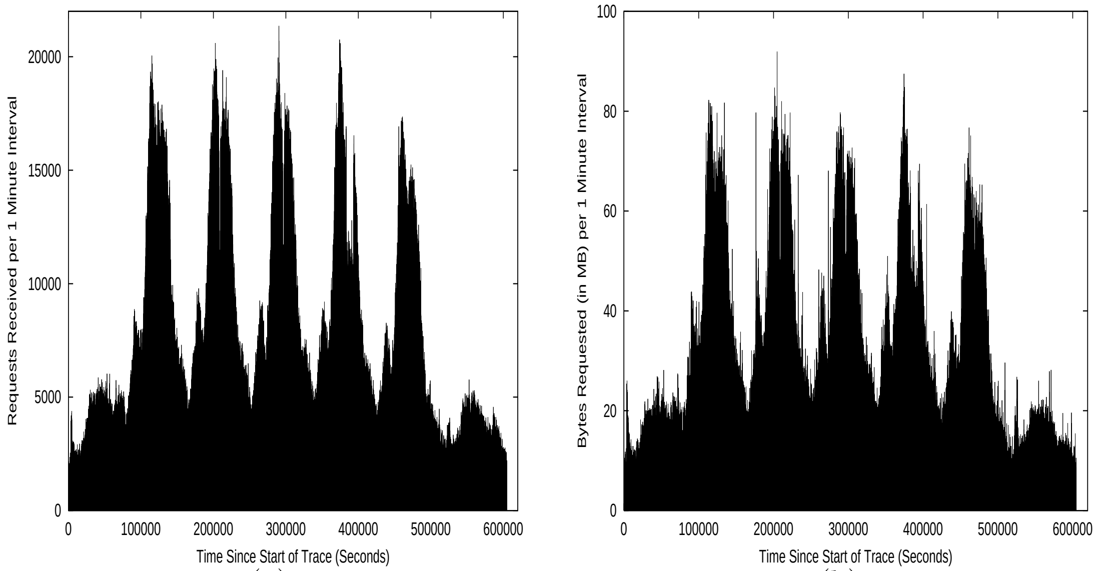
4.1.3 Reduced Workload Analysis

The first step in carrying out a workload characterization is to fine-tune the raw workload by removing unwanted information. In this regard, useful server workload

[1]A client is a host having a distinct Internet Protocol (IP) address in the access log. In this case, a client can be a single-user host, a timesharing cluster host or a relay host.

Table 4.1: Statistical Summary of Raw and Reduced Access Logs

Item	Raw Access Log	Reduced Access Log
Access Log Duration	7 days	7 days
Access Log Starting Date	June 14, 1998	June 14, 1998
Access Log End Date	June 20, 1998	June 20, 1998
Document Requests		
Total Requests	79,747,430	56,458,479
Mean Requests/ Day	11,392,489	8,065,497
Median Requests/ Day	13,565,362	9,517,399
Standard Deviation	3,992,015	2,811,914
Coefficient of Variation	0.35	0.35
Client Requests		
Distinct Clients	664,187	620,041
Mean Requests Per Client	124	91
Median Requests Per Client	62	52
Standard Deviation	1013	490
Coefficient of Variation	8.18	5.38
Client Bytes Transferred		
Mean Transfer Size (Bytes)	529,268	521,829
Median Transfer Size (Bytes)	267,969	266,534
Standard Deviation (Bytes)	5,061,640	4,998,393
Coefficient of Variation	9.96	9.58
Document Bytes Transferred		
Total Bytes Transferred (GB)	328	323
Mean Transfer Size (Bytes)	5,773	5,722
Median Transfer Size (Bytes)	1,236	1,227
Standard Deviation (Bytes)	32,292	31,485
Coefficient of Variation	6.59	5.50
Bytes Transferred per Day		
Mean Transfer Size / Day (MB)	47,326	46,105
Median Transfer Size / Day (MB)	55,226	54,308
Standard Deviation (MB)	15,611	16,009
Coefficient of Variation	0.33	0.35
Unique Document Size		
Total Unique Documents	104,575	101,008
Total Storage Size (MB)	2,819	2,819
Mean Document Size (Bytes)	27,918	27,918
Median Document Size (Bytes)	8,506	8,506
Standard Deviation (Bytes)	271,930	271,930
Coefficient of Variation	9.74	9.74



(a) (b)

Figure 4.1: Seven Days Traffic Profile of Web Server Workload Trace: (a) Requests per Minute; (b) Bytes Transferred per Minute

that contains information pertinent to the research was obtained by analyzing the response codes and resource methods in each client request found in the Web server access log. The distribution of requests by resource methods shows that 99.17% of all requests contain the GET method, while other resource methods account for 0.83%. This simply shows that the stated URLs were retrieved documents. Table 4.2 shows the distribution of response codes. Successful requests account for 71.56% of the total requests found in the access log, which represents 97.63% of bytes transferred by the server. This differs from the success rate reported in [7] [10], but agrees with the bytes transferred rate in [11], suggesting that the server workloads have changed over time.

The second most common status code was the Not Modified (304) response code which accounts for 18.33% of all the responses to client requests. This represents a substantial increase over the fraction reported in [10], but is almost the same as reported in [11]. This increase was attributed to improved caching in the Web, including persistent caches in browsers, proxies and network. The third most common

Table 4.2: Breakdown of Server Response Codes

Response Code	% of Requests	% of Content Data Transferred
200 (Successful)	71.56	97.63
206 (Partial Content)	0.37	1.48
304 (Not Modified)	18.33	0.00
4xx (Client Error)	6.15	0.00
5xx (Server Error)	0.00	0.00
Other Codes	3.59	0.89
Total	100.00	100.00

status code in Table 4.2 is the Client Error code (4xx), which accounts for 6.15% of all responses to client requests. The increase over workload characterized in [7] [10] can possibly be due to incorrect URLs specified by clients. In addition, the table shows that the server did not fail to accept and process any valid request from clients. Other response codes observed can easily be ignored because of their very low representation. The reduced access log, however, comprises all requests with the GET method that have a successful response code. Although there were requests with zero[2] transfer size in the reduced workload, their proportion was negligible. Therefore, any document whose size is zero is assumed to be error-prone and is excluded from statistical computations.

Table 4.1 gives detailed summary statistics of the reduced workload. These can be subdivided into information relating to requests, transfer sizes and clients. The information on requests and transfer sizes deals with document referencing behaviour and bytes transferred. The reduced workload contains 56,458,479 requests, which represents 70.79% of the raw workload, with 620,041 clients. These requests accounted for 323 GB of data transfer across the network. In the reduced log, the daily means of server activity were reduced to 8,065,497 requests and 46 GB. Similarly, daily medians of server activity were 9,517,399 requests and 54 GB. These imply that the weekend effect present in the raw workload is preserved in the re-

[2]The reduced access log has 57,497 requests with zero transfer size, which account for 0.1% of the total reduced access log requests.

duced workload. The number of unique documents reduces to 101,008 (101,007 static documents and a single dynamic document) and occupied 2,818 MB of disk space[3] on the server. The total number of unique documents found in the reduced workload was 0.18% of total document requests. However, there is no significant difference in the disk space occupied by unique documents found in the raw and reduced workload. In both cases, mean and median document sizes were 28 kB and 9 kB, respectively. This observation implies that a few large documents skewed the mean document sizes.

In terms of transfer sizes, total unique document sizes represent 0.88% of the total bytes transferred by the server. The persistence of a high coefficient of variation for unique document sizes after the reduction process indicates the preservation of the document size distribution found in the raw workload. Interestingly, the statistical information for both raw unique documents and reduced unique documents are the same, which suggests that most invalid requests are error requests or requests of zero length. The overall mean transfer size is 5.7 kB, which falls within the range found in [7] [10] [26] [83]. The median transfer size is 1.2 kB, which is close to that reported in [7] [10] where significant difference between mean and median was attributed to the presence of a few large transfers that skewed the mean transfer size. This observation has been reported in other workload analyses in [7] [9] [10] [12].

Table 4.1 shows that there are 620,041 clients in the reduced access log. The mean and median requests per client are 91 and 52, respectively, while the mean and median transfer sizes per client are 521 kB and 267 kB, respectively. This shows that the service demand for most clients is low during the period of trace collection. However, the coefficients of variation for total requests and transfer size per client are 5.38 and 9.59, respectively, implying that there is a high degree of variation in client service demands[4]. A similar observation was reported in [66].

[3]For static documents, unique document size is estimated using the maximum transfer size of the document over the duration of the trace. For dynamic documents, document size is estimated using the mean transfer sizes.

[4]This observation does not necessarily imply that clients that generate heavy requests also

4.1.4 Document Types and Sizes

The analysis of document types and sizes found in the reduced access log of the server is very useful in order to distinguish document types that are heavily demanded from those that are lightly demanded. The grouping of document types in the access log is similar to what was reported in [7] [9] [10] [11] [61]. Table 4.3 shows the breakdown of document types as requested by the clients. The table reveals that most client requests (95.60%) were for either HTML (9.51%) or Image (86.09%) documents. This is different from results reported in [10] where the proportion of clients' requests to HTML and Images documents were nearly the same. In terms of the volume of content data transferred, HTML documents account for 23.66% and Image documents account for 63.28%. This implies that Image documents have greater impact than HTML documents, whereas the reverse was reported in [7]. However, both HTML and Image documents account for almost 87% of the volume of content data transferred. This shows an increase in the rate at which Web users are abandoning text-based documents in favour of multimedia documents. Directory documents account for 2.68% of the client requests and 4.80% of the volume of content data transferred, while formatted documents account for 0.12% of client requests and 5.01% of the volume of content data transferred.

In the reduced workload, requests for documents which are regarded as uncacheable (i.e. dynamic documents) account for a small proportion of client requests (0.52%) and volume of content data transferred (0.35%). However, this class of document type has the highest coefficient of variation (13.08) in terms of transfer sizes. This document type does not seem to pose a serious threat to the network bandwidth. Following these trends of variability are documents of Other Types, which are not classified but have a high degree of variability in terms of transfer sizes with a coefficient of variation of 11.29. Compressed, Programs, Video, Formatted and Audio types of documents (in this order) have high mean transfer sizes, while Dynamic documents have the lowest. Although these document types have a very

transfer heavy data (in bytes) across the network. Further analysis will be required to investigate the hypothesis.

Table 4.3: Breakdown of Document Types and Sizes Transferred

File Type	% of Requests	% of Bytes	Mean of Bytes	Median of Bytes	StD of Bytes	CoV of Bytes
HTML	9.506	23.66	14,243	11,878	12,297	0.86
Images	86.089	63.28	4,205	922	8,455	2.01
Audio	0.005	0.17	210,038	11,775	933,290	4.44
Video	0.003	0.33	733,857	147,456	1,861,538	2.54
Dynamic	0.518	0.35	3,899	1,333	50,992	13.08
Formatted	0.120	5.07	241,949	61,440	540,939	2.24
Compressed	0.001	0.24	1,357,324	483,328	3,449,985	2.54
Programs	0.006	1.01	1,021,536	392,641	1,221,071	1.20
Directory	2.681	4.80	10,258	11,451	4,918	0.48
Other Types	1.072	1.10	5,878	223	66,336	11.29
Total	100.000	100.00				

low profile in the workload, a future increase in their demand may have an adverse effect on network resource and server management. For instance, the huge discrepancy between the percentage of requests (0.001%) for Compressed documents and the percentage of content data transferred (0.24%) is an indication of the effect that large documents might have on the workload of a server and network.

4.1.5 One-Timer Referencing

Another important observation which is common to other related work [7] [10] [61] is the phenomenon of *one-timers*. Table 4.4 gives detailed statistics on one-timer documents and one-timer clients. One-timer documents are unique documents at the server that appear only once in the reference stream. The analysis of one-timer document referencing behaviour is important because of its adverse effect on caching performance. Previous observations of Web server workloads have shown that one-timer referencing of documents is an invariant characteristic that accounts for approximately one-third of document references and bytes transferred by the server [10]. The observations from this reduced access log suggest that one-timer document referencing may have changed over time. Table 4.4 shows that 12,357 unique documents are one-timer documents and these account for 348 MB of disk space on the

Table 4.4: Statistics on One-Timer Clients and Documents

Item Description	Reduced Workload
Unique Documents	101,008
One-Timer Documents	12,357
One-Timer/Unique Documents(%)	12.23
Total Requests	56,458,479
One-Timer Requests	12,357
One-Timer/Total Requests (%)	0.02
Total Clients	620,041
One-Timer Clients	33,205
One-Timer/Total Clients (%)	5.36
One-Timer Documents Transfer Sizes	
Total Bytes Transferred (GB)	323
Total Bytes for Unique Documents (MB)	2,818
Total Bytes for One-Timer Documents (MB)	348
Mean Transfer Size (Bytes)	28,236
Median Transfer Size (Bytes)	7,123
Standard Deviation (Bytes)	224,637
Coefficient of Variation	7.96
One-Timer Bytes/Unique Document Bytes (%)	8.78
One-Timer Bytes/Total Bytes (%)	0.11
One-Timer Clients Transfer Sizes	
Total Bytes Transferred (GB)	323
Total Bytes for One-Timer Clients (MB)	333
Mean Transfer Size (Bytes)	10,396
Median Transfer Size (Bytes)	6,619
Standard Deviation (Bytes)	72,878
Coefficient of Variation	7.01
One-Timer Bytes/ Total Bytes (%)	0.10

server. This means about 12.23% of unique documents accessed were one-timers, while the volume of one-timer documents transferred was 8.78% of unique document bytes transferred. The results imply that one-timer document reference is less in this workload than has been reported previously.

Analysis of *one-timer clients* is also important because of their adverse effect on persistent connection (P-HTTP) behaviour. One-timer clients are those clients which are responsible for a single request over the period studied. In the reduced workload, one-timer clients are 33,205 in number and account for 5.35% of distinct clients. The total data transfer size associated with one-timer clients is 333 MB, which represents 0.10% of the total data transfer to distinct clients. This implies that the number of requests and total transfer sizes of one-timer clients are very insignificant, and so the negative implications of persistent connections may not be present at this server. Both the one-timer documents and clients have high coefficients of variation for transfer sizes; 7.01 and 7.96, respectively. This is an indication that one-timers' transfer sizes observed in this workload are widely spread relative to their one-timer mean transfer size. Because of this wide variation, any size-based caching policy may not discriminate against one-timer documents effectively. On the other hand, policies that manage server resources from the client point of view may not have to consider the impact of one-timer clients since their proportion is negligible.

4.1.6 Summary

Statistical characteristics of the server workload have been presented. The results show that the characteristic features of the raw workload are preserved when the workload is reduced. For instance, the weekend effect is preserved and the server traffic is very bursty during the working days of the week, while during the weekend, the traffic is less bursty. However, some important statistical characteristics of server workload have changed from what have been reported previously—such as the number of one-timer documents and the proportion of total requests to Image documents. The client requests being characterized in the following sections exclude dynamic documents, error requests and requests for documents of zero length.

4.2 Document Size Distribution

This section describes in detail the size distribution of the documents successfully transferred over the network, and the size distribution of documents that appear once. Investigating the size distribution (unique and transfer documents) can give more insight into understanding factors that influence the properties of network traffic and the nature of documents available at the Web server. In addition, the empirical document size distribution will be compared with analytical models.

4.2.1 Document and Transfer Size Distributions

One of the most frequently asked research questions is whether any of the document and transfer size distributions exhibits a heavy-tailed property. Such a property is very important because it has been shown that its presence can cause network traffic to be self-similar, which implies burstiness at a wide range of time scales [35]. Intuitively, a heavy-tailed document size distribution implies the existence of a small set of large documents which accounts for a significant percentage of the volume of bytes transferred over the network. Mathematically, a distribution F having a random variable X is considered to be heavy-tailed if

$$P[X > x] \sim x^{-\alpha}, \quad \text{as} \quad x \to \infty, 0 < \alpha < 2 \tag{4.1}$$

where α measures the heaviness of the tail. Equation (4.1) means that if the asymptotic shape of the size distribution is hyperbolic, then the distribution is said to be heavy-tailed, regardless of the distribution for small values. The Pareto distribution (also known as the power-law distribution) is the simplest heavy-tailed distribution in the literature. It arose out of attempts to model the distribution of incomes across various countries, professions and trades. Recently, the Pareto distribution has been used to characterize the distribution of fault-trace lengths in controlling the deformation behaviour of rock masses [33], FTPDATA bursts in wide-area traffic [76] and Web traffic [7] [11] [15] [35] [36] [61]. The probability distribution function (PDF) for the Pareto distribution is given as

$$p(x) = \alpha k^{\alpha} x^{-\alpha-1}, \quad \alpha, k > 0, x \geq k \tag{4.2}$$

and its cumulative distribution function (CDF) is expressed as

$$F(x) = P[X \leq x] = 1 - (k/x)^{\alpha}, \quad \text{for all} \quad x \geq k \tag{4.3}$$

where α is known as the tail index or Pareto exponent and k represents the smallest possible value of the random variable. If $\alpha \leq 2$, then the distribution has infinite variance, and if $\alpha \leq 1$, the distribution has an infinite mean. Thus, as α decreases, the tail of the distribution becomes flatter and the portion of the probability mass present at the tail becomes larger. Similarly, the closer the value of α is to 2, the steeper the tail becomes and the smaller the proportion of probability mass present at the tail.

Figure 4.2 shows the presence of heavy-tailed property in the document and transfer size distributions. For instance, Figure 4.2(a) depicts the CDF of document references, unique document sizes, and transfer sizes, with document size on the horizontal axis and cumulative percentage on the vertical axis. The graph reveals the heavy-tailed property in its S-shape. For document size distribution, this shape means that a few large documents occupied a significant percentage of the server disk space. In the case of transfer size distribution, the shape implies that a few large documents were responsible for a significant percentage of the volume of data (in bytes) transferred by the server. Figure 4.2(a) illustrates this heavy-tailed property by indicating that 55.81% of all documents available at the server were less than 10 kB. However, 28% of the volume of bytes transferred by the server were generated by this category of documents. At the tail of the distribution, 5.75% of of all documents available at the server were over 40 kB in size. This category of a few large documents accounts for 21.86% of the volume of bytes transferred by the server.

In addition, documents less than 10 kB were responsible for 80.49% of all requests to the server, while documents over 10 kB were responsible for about 72.46% of the bytes transferred by the server. The closeness in percentages is due to 46.65% of total requests which were for documents less than 1 kB, while 49.60% of bytes transferred were for documents between 10-40 kB. From this analysis, it is likely

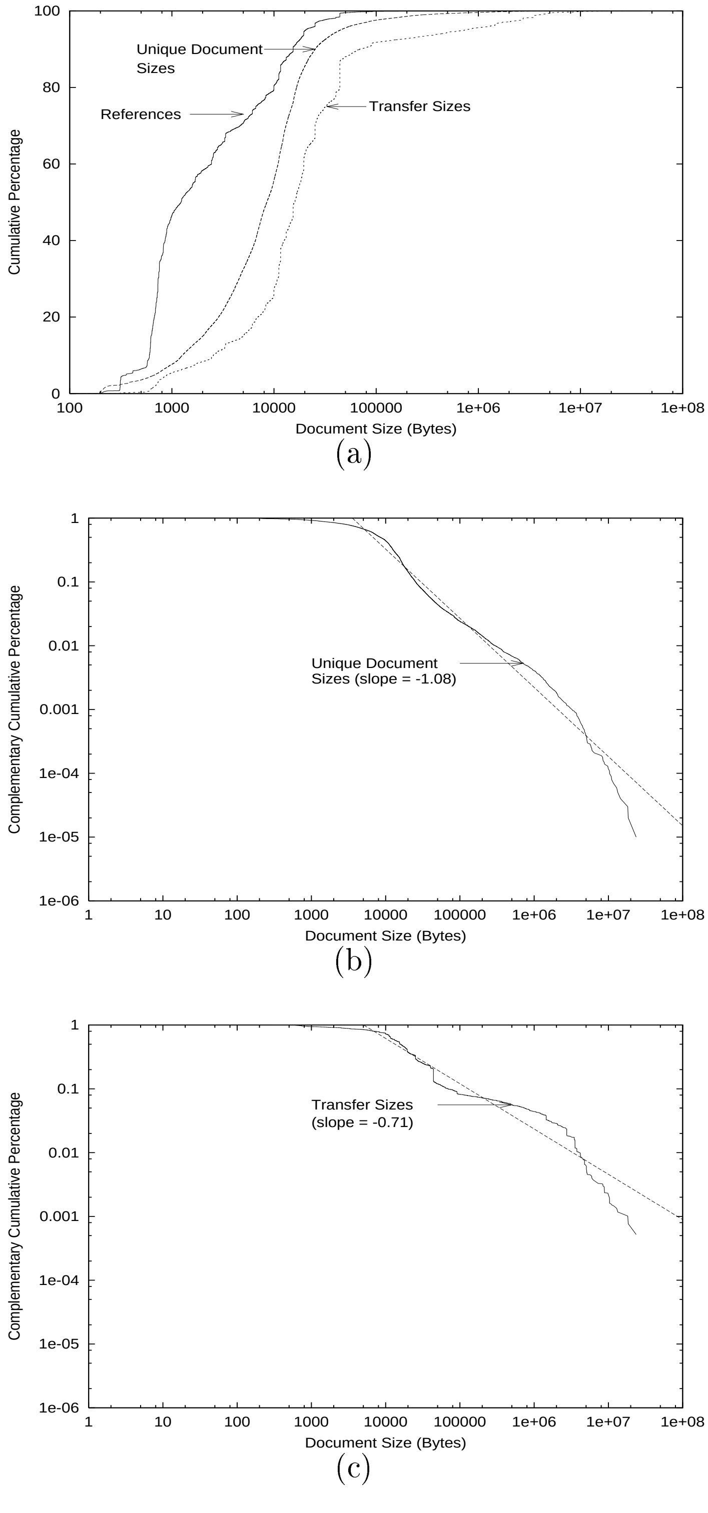


Figure 4.2: Document Size Distribution: (a) CDFs by Document Sizes; (b) LLCD Plot of Unique Documents; (c) LLCD Plot of Transfer Sizes

that for documents greater than 10 kB, document and transfer size distributions can be modeled reasonably well by the Pareto distribution.

In order to estimate the tail index α for the unique document and transfer size distributions, a log-log complementary distribution (LLCD) plot is applied [35] [65]. This plot is the complementary cumulative distribution of Equation (4.3), implying that $\bar{F}(x) = 1 - F(x) = P[X > x]$ on log-log axes. If plotted in this manner, a heavy-tailed distribution has the property that

$$\frac{log\bar{F}(x)}{logx} = -\alpha, \quad \text{for } x > k \tag{4.4}$$

for some k. Equation (4.4) suggests plotting $log(1 - F(x))$ against $logx$ and examining if a straight line results. The presence of a straight line suggests that the distribution is heavy-tailed and the α parameter can be estimated as the slope of the line. A nearly linear relationship indicates a hyperbolic upper tail, as observed in Figure 4.2(b)-(c). In order to quantify the degree of the heavy-tailed behaviour, equally spaced points from among the LLCD points larger than the k value were selected and least-squares estimation (LSE) was used to estimate the slope. Alternatively, maximum likelihood estimation (MLE)

$$\alpha = \sum_{j=1}^{n} \frac{n}{log(X_j/k)} \quad \text{for all } x_j \geq k \tag{4.5}$$

can also be used to estimate α. The resulting α and k estimates using LSE and MLE techniques for unique document and transfer size distribution are shown in Table 4.5. The coefficient of determination (R^2) is a measure of the variation in the document size that is explained by the analytical model. The closer the value of R^2 is to 1.0, the better the fit.

The results show that MLE estimates are affected by the clustering of document sizes at the upper tail. The value of the α estimate from MLE is significantly different from its LSE counterpart. This implies that the clustering effect is more pronounced in the transfer size distribution than in the unique document size distri-

Table 4.5: Estimation of Pareto Index for Size Distributions

Distribution		MLE	LSE	
	k (Bytes)	α	α	R^2
Unique Document Sizes	20,000	1.15	1.08	0.97
Transfer Sizes	10,000	1.82	0.71	0.92

bution. The goodness-of-fit measure R^2 also supports this evidence. In addition, the preponderance of fairly large image and text document accesses can be responsible for the difference.

4.2.2 Document Size Modeling

The distribution of document sizes at the Web server is characterized for the purpose of making comparison with an appropriate theoretical distribution. Barford *et al.* [15] [16] demonstrated that a hybrid lognormal-Pareto model can capture the body and tail of the document size distribution. To check for the validity of this model, the following procedure was developed.

1. Probability plots[5] for the entire document size distribution were carried out to narrow the set of candidate models for the distribution. Five candidate models were examined: normal, lognormal, exponential, uniform and Pareto[6].

2. Formal numerical techniques, such as Andersen Darling (A^2) and Kolmogorov statistics, were employed to verify the validity of the probability plot in capturing the hypothesized distribution including the tail with 95% confidence.

3. Least-squares estimation was used to estimate the slope and intercept of the probability plot. A good fit emerges if a straight line drawn using the LSE

[5]The concept of probability plotting is broadly used to transform the vertical scale of an empirical cumulative distribution function to a scale which will produce exactly a straight line, if the analytical and empirical distributions resemble each other.

[6]To model the empirical dataset as a Pareto distribution, the transformed variable $Y = log(X/k)$ (i.e, the denominator of Equation (4.5)) must satisfy an exponential distribution. Clark *et al.* [33] explain this further.

estimators coincides with the probability plot.

4. A goodness-of-fit line that matches the whole empirical distribution indicates that the analytical model captures the empirical distribution. The closeness of the goodness-of-fit line to either the body or the tail of the distribution likewise indicates the part of the empirical distribution which the analytical model fits well. The goodness-of-fit line of a probability plot was used to determine where to divide the body of the hypothesized distribution from the tail.

5. Parameter values were estimated for candidate models that closely fit the body or the tail.

6. Formal numerical techniques (as mentioned in step 2) were used to ensure the goodness-of-fit of the analytical model(s).

The procedure is similar to other techniques reported in [13] [15]. With the use of this procedure to model the unique document size distribution, Figure 4.3(a)-(b) depicts probability plots for the entire distribution using the lognormal model and Pareto distribution for the upper tail. Figure 4.3(a) shows that the lognormal distribution can be used to model the body of the distribution, but deviates significantly from the upper tail. The point of deviation is shown by the goodness-of-fit line that approximates the document size at which the upper tail of the dataset may start. The clumping of document sizes around 10,000 bytes to 40,000 bytes formed the bulk of documents at the upper tail.

Figure 4.3(b) shows the goodness-of-fit for the Pareto distribution with the smallest value k given as 10,000 bytes. However, the A^2 test showed no significance in terms of goodness-of-fit for the body and tail, respectively. This failure is due to the clumping of document sizes around the body and tail of the distribution as shown in Figure 4.3(a)-(b). A similar problem has been reported for document size distribution in [15]. Figure 4.3(c)-(d) demonstrates the overall goodness-of-fit of the hybrid model with the hypothesized distribution using CDF and LLCD for the tail.

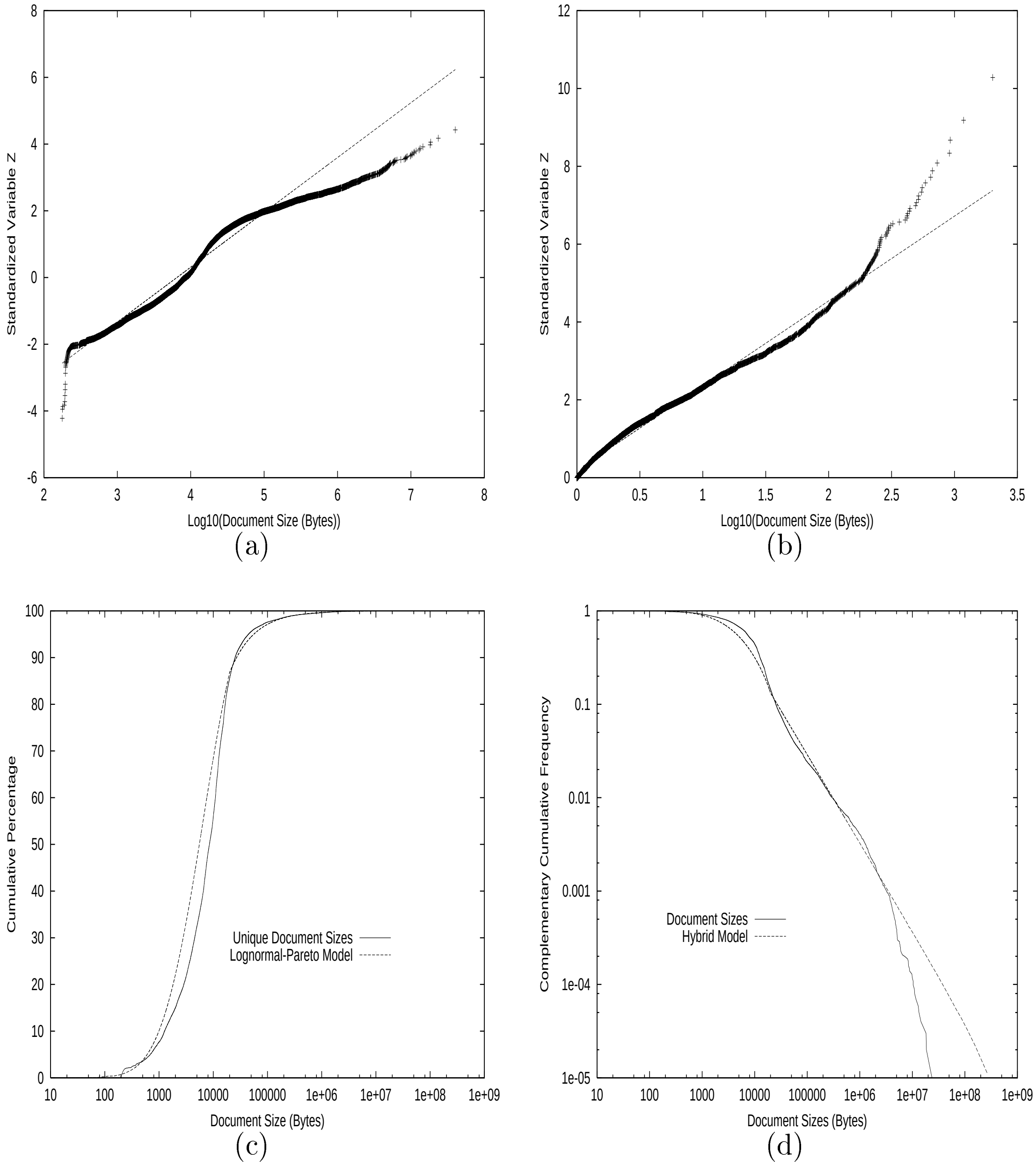


Figure 4.3: Modeling Size Distribution: (a) Lognormal Probability Plot for Unique Documents; (b) Pareto Probability Plot for Upper Tail of Unique Document Distribution; (c) CDF of Hybrid Lognormal-Pareto Model; (d) LLCD of Hybrid Lognormal-Pareto Model

Table 4.6: Hybrid-Model Parameters for Unique Document Sizes

Components	Model	Unique Documents		
		Slope	Intercept on Y axis	R^2
Body	Lognormal	$a = 1.64$	$b = -6.27$	0.93
Tail	Pareto	2.18	0.19	0.98
Percent Document in the Upper Tail	15%			

The results show that a hybrid lognormal-Pareto model approximates document size distribution at the server. Table 4.6 gives the estimated parameters for the hybrid Lognormal-Pareto model.

4.2.3 Summary

The transfer size of documents over the network and the size of unique documents accessed by the clients within the data collection period were characterized. These distributions are observed to be heavy-tailed. In addition, a novel procedure was developed to model the document size distribution using the probability plot. The result shows that a hybrid Lognormal-Pareto model can be used to closely capture document size distribution at the server.

4.3 Document Referencing Behaviour

The purpose of this section is to investigate whether or not the document referencing behaviour reported in [7] [9] [10] [11] [83] has changed over time. Important characteristics such as concentration of references, correlation of document references and data transferred with document sizes, temporal locality and inter-reference times are discussed.

4.3.1 Concentration of Document References

Many studies have found that Web documents exhibit a non-uniform referencing pattern [9] [10] [36] [61]. This means that a small percentage of Web documents are extremely popular to users, while a significant number of Web documents are rarely requested. To illustrate the presence of concentration in the server workload, the list of distinct documents is arranged in decreasing order (from the most popular to the least popular documents) based on how many times the unique documents were accessed against the fraction of total documents referenced.

Figure 4.4 shows that there is a concentration of references among a small subset of the unique documents at the server. The first 10% of unique documents (representing the most popular documents at the server) were responsible for 96.05% of document requests. About 30% and 50% of the most popular documents were responsible for 98.81% and 99.61% of total requests, respectively. Similar results are observed for the bytes transferred by the server. For example, Figure 4.4 shows that the first 10% of unique documents (representing the most heavy transfer documents at the server) were responsible for 93% of bytes transferred. The results show that the popularity profile of documents at the server is proportional to the degree of busyness experienced by the server. They also suggest that an extremely busy server extends the "10/90" rule for document concentration identified in [7] [10]. A similar observation was made in [11].

Another common approach for characterizing the frequency of reference to unique documents is to test for conformity with *Zipf's Law*. This law was first postulated by George Kingsley Zipf [91] to explain how humans process and store information in natural languages where a very small proportion of words are often used in texts while the remaining words are rarely used. Further empirical observation in social sciences revealed that the size distribution of cities in most countries follow Zipf's Law [45]. Peachey *et al.* [73] identified many phenomena in computer systems that can be modeled with Zipf's Law. Recently, this law has been applied in characterizations of Web workloads, in particular in the characterization of the document

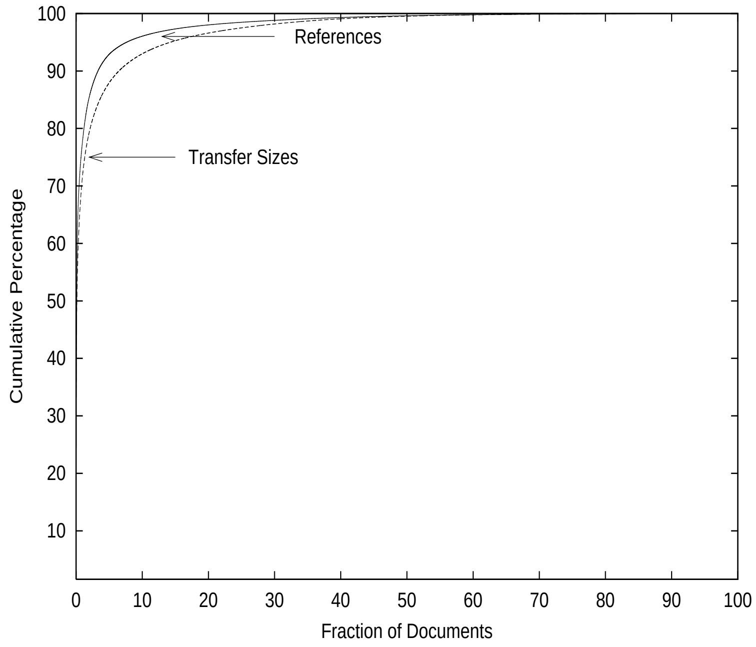


Figure 4.4: Concentration of References and Bytes Transferred

popularity profile of the client [15] [36], proxy [27] [28] [61] and server [5] [7] [9] [10] [11] [46] workloads. To visualize Zipf's law in document popularity profile, unique documents are sorted in decreasing order of popularity and a rank number is assigned to each unique document. For documents with the same frequency, the subsorting and ranking is arbitrary. The most frequently accessed document is assigned a rank of 1 while the least frequently accessed document is assigned a rank of N (which is the same as the total number of unique documents) and the popularity is plotted as a function of the rank. If the empirical distribution obeys Zipf's Law, the log-log transformation plot of the frequency of documents versus rank will approximate a straight line of slope -1.0. A *Zipf-like* distribution [27] [28] describes a popularity curve with a slope between -0.5 and -1.0.

The popularity profile of unique documents within the period of observation does not follow Zipf's Law because the slope of the popularity line is -1.59 ($R^2 = 0.99$) as shown in Figure 4.5. This popularity behaviour is due to the peculiar nature of this server workload in which a very small number of unique documents (about

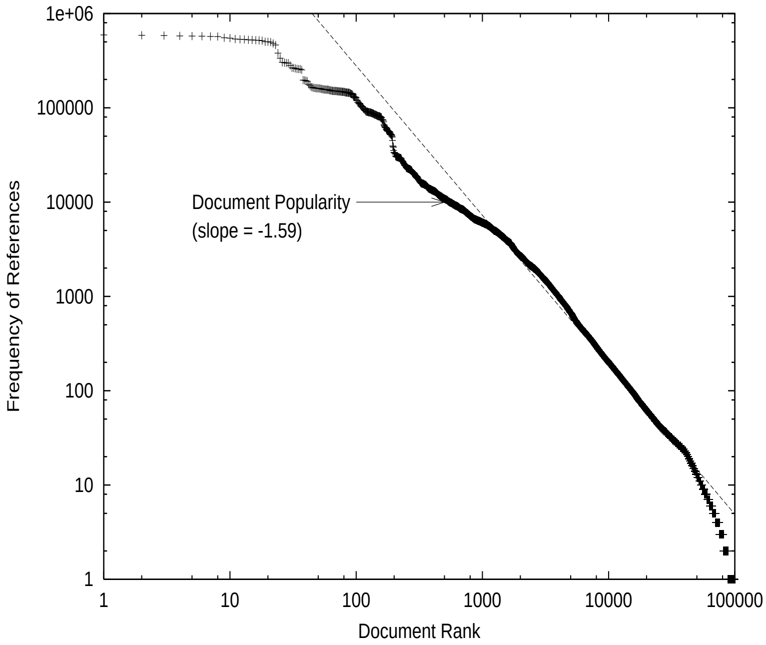


Figure 4.5: Zipf Distribution: Document Reference Count versus Rank

5%) are extremely popular (i.e., responsible for about 93% of all the document requests received by the server), while a large number of unique documents are unpopular. Despite the non-conformity with Zipf's Law, document caching is likely to be successful at the Web server, since the steeper the slope of the popularity curve, the stronger the frequency of reference to a smaller percentage of documents. The concentration of document referencing behaviour including the characteristic of the document popularity profile observed in this workload is similar to the observation reported for the 1998 World Cup (i.e., soccer competition) server workload [11].

4.3.2 Correlation Among Document Sizes, Frequency of References and Volume of Data Transferred

The results presented in Section 4.3.1 established that a small proportion of the documents at the server are frequently requested, while a large proportion are not. In this section, the measure of relationship between the size of document and the frequency of references to the document is investigated on one hand. On the other hand, the degree of relationship between the size of document and the volume of data transferred that accrued to the document is also investigated. The study of these two relationships is meant to improve the understanding of document referencing behaviour and resource management at the server.

While a substantial research activity has been directed at finding the correlation between the size of document and the frequency of reference to the document [7] [9] [15] [35], no statistical evidence to date shows the correlation between the size of document and volume of data transferred that accrued to the document. The determination of the two correlations are important for cache policies that optimize request throughput and data throughput, respectively.

Correlation is a statistical measure that attempts to quantify the degree of relationship that exists between two variables. Recently, Barford *et al.* [15] proved that the correlation measure is related to the difference in performance between the document hit ratio[7] and the byte hit ratio[8]. This mathematical relationship is represented as

$$H_\infty - B_\infty = \frac{-Cov(R_i, S_i)}{\mu_r \mu_b} \tag{4.6}$$

where $Cov(R_i, S_i)$ is the sample covariance between the frequency of reference R and the document size S. μ_r represents the mean number of requests per document and μ_b is the mean document size. Equation (4.6) means if there is no correlation between the document size and its likelihood of being accessed, then there will be no difference between document hit ratio H and byte hit ratio B. In addition, a negative covariance indicates a preference for smaller documents (resulting in $H_\infty > B_\infty$), while a positive covariance indicates a preference for larger documents (resulting in $H_\infty < B_\infty$).

Another statistical measure is the correlation coefficient between two variables.

[7]The ratio of the number of document requests satisfied by an infinite-sized cache to the total number of requests received by the server.

[8]The ratio of the volume of data in bytes satisfied by an infinite-sized cache to the total volume of data transferred by the server.

Table 4.7: Weak Preference Indicator for Smaller Documents

Measurement	Value
Cov(R_i, S_i)	-12,059,050.75
Mean Requests per Document (μ_r)	556
Mean Document Size (μ_s)	27,918
Mean Bytes Transferred per Document (μ_b)	3,471,891
$H_\infty - B_\infty$	-0.00624
Correlation Coefficient	-0.00427

Arlitt [7] and Mahanti [61] applied this measure to investigate the relationship between the size of document and the frequency of references to the document. The measure is given as

$$Corr(X, Y) = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\left[\sum_{i=1}^{n}(x_i - \bar{x})^2(y_i - \bar{y})^2\right]^{1/2}} \tag{4.7}$$

Equation (4.7) gives a value between -1 and +1 to quantify the preference, and its magnitude is not affected by any scaling of variables. A negative sign implies a preference for smaller documents, a positive sign is for larger documents, and magnitude zero is for indifference. The result from this measure can also be related to a measure of difference in document hit ratio and byte hit ratio of an infinite-sized cache as stated in Equation (4.6).

Table 4.7 summarizes the results obtained from using Equations (4.6) and (4.7). The covariance value is negative, which implies that $H_\infty > B_\infty$. Because the normalized covariance is very close to zero (i.e., -0.00624), any preference for small documents over large documents is at most very weak. Using Equation (4.7), the correlation coefficient is -0.00427, which implies the same thing as normalized covariance. The slight difference between the results obtained using Equations (4.6) and (4.7) can be tied to the unrealistic assumption of infinite cache or the sensitivity of covariance magnitude to the scaling of variables.

The correlation obtained in using both statistical measures is similar to other reports in [7] [9] [61] that correlation between the document size and the frequency of reference is extremely close to zero, but concluded that such observation does not imply absolute indifference for preferring to reference small documents over large documents. Similarly, the correlation between the size of document and the volume of data transferred that accrued to the document using Equation (4.7) is +0.0521. The positive correlation implies that $B_\infty > H_\infty$, which implies that the larger the documents accessed by clients, the heavier the volume of data transferred over the network. Because the correlation is close to zero, the influence of large documents on the volume of data transferred over the network is considered weak.

In order to understand further the relationship between two variables, a graphical method is used by plotting the document size on the x-axis (in 1 kB bins) and the probability percentage on the y-axis. Figure 4.6(a) shows that there is high probability of referencing small documents that are less than 10 kB, while there is almost zero probability of referencing large documents greater than 100 kB. This graph shows that clients prefer to reference small documents that are less than 10 kB. Figure 4.6(b), however, shows documents between 1-100 kB are responsible for most bytes transferred over the network. Large documents beyond 100 kB contribute less, but significant, volume of bytes transferred over the network. This is because the probability of documents greater than 100 kB is non-zero.

Comparing areas covered by document size with high frequency of references and volume of bytes transferred, Figure 4.6(c) suggests that documents smaller than 100 kB, if cached, can improve both the request throughput and data throughput. The reason is that caching documents with sizes more than 100 kB may not lead to any significant performance improvement. With careful inspection of the plots in Figure 4.6, it is observed that documents less than 40 kB were most referenced at the server and were responsible for most bytes transferred over the network. This scenario is likely due to a change in the referencing behaviour of Web users as a result of lack of patience to download larger documents due to poor response times. A size-based replacement policy that preserves documents in this range could optimize both the request throughput and data throughput of a busy server.

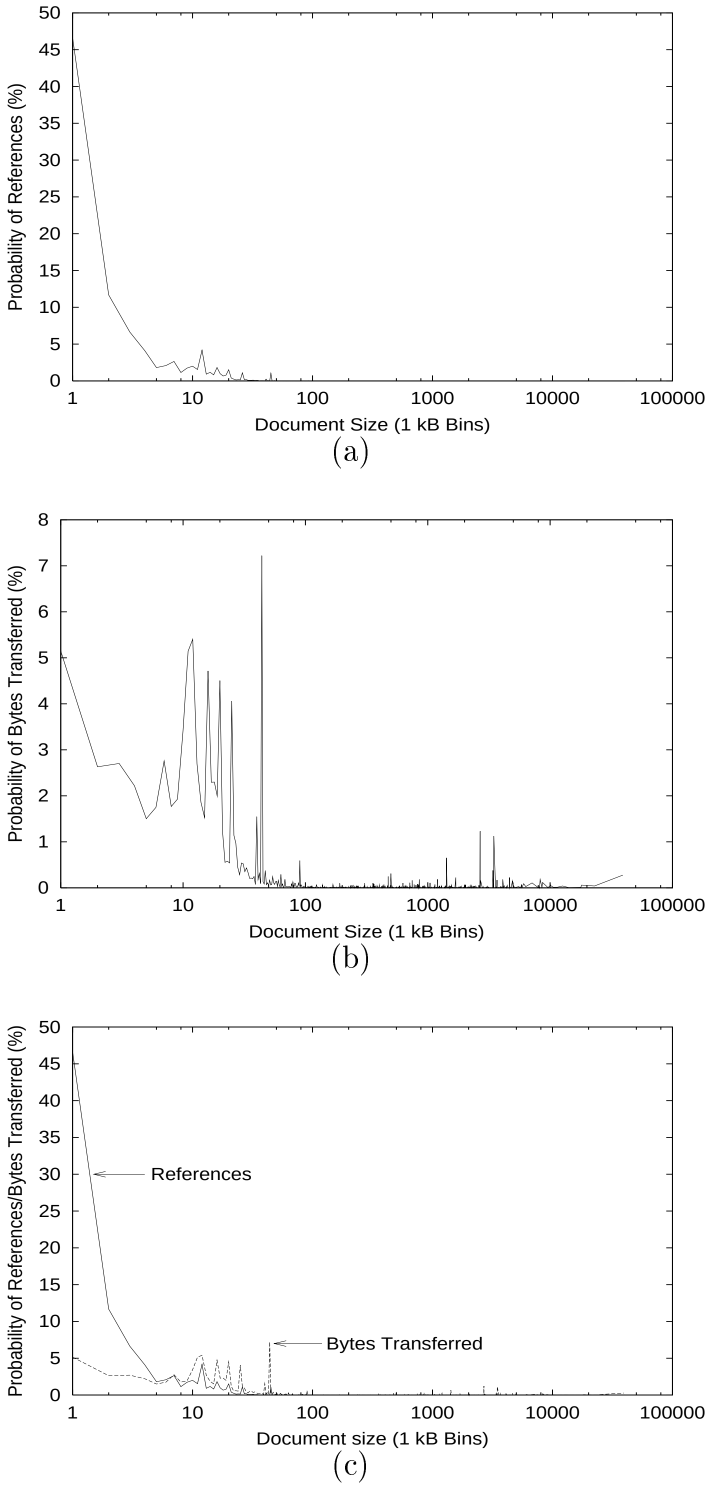


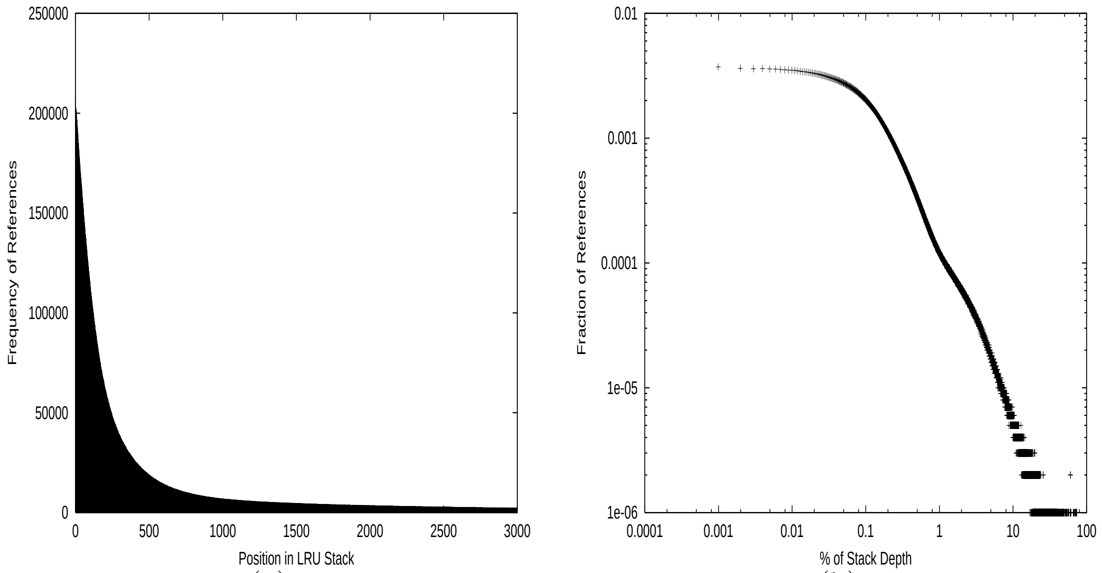
Figure 4.6: Probability Density Function of Document Sizes: (a) Number of Requests; (b) Volume of Bytes Transferred; (c) Number of Requests and Volume of Bytes Transferred

4.3.3 Temporal Locality

The fundamental properties that characterize the phenomenon of temporal locality were first described by Denning and Schwartz [38] and Denning [39]. Temporal locality is recognized not only as essential to the achievement of acceptable cache performance, but also as a property which directly influences the choice of cache management policy [43]. The presence of temporal locality in Web document referencing behaviour, though in lesser degree as compared to memory and file systems, has been reported in [7] [35]. Temporal locality means that a document that was recently requested is likely to be requested again in the near future. The Least Recently Used Stack Reference Model (LRUSRM) [81] [84] [62] is often used to measure temporal locality. The LRUSRM reflects the tendency for references to certain documents to persist over time, i.e. the possibility for clients to frequently re-reference documents that have been requested recently in the past.

Figure 4.7(a) shows the LRU stack distance histogram plot. The x-axis represents the first 3000 stack positions ordered by recency of reference (where 1 is the top of the stack for the most recently referenced document and 3000 is the stack position for the least recently referenced document). The y-axis represents the frequency of reference (or number of hits) to the document occupying each of the stack positions. The hyperbolic curvature of the LRU stack distance histogram in Figure 4.7(a) demonstrates the presence of temporal locality in the workload. Most document hits occurred at stack positions that are very close to the top of the stack. A log-log plot of the stack distance probability density function against the percentage of stack positions relative to stack depth is shown in Figure 4.7(b). This plot also demonstrates the presence of temporal locality because the frequency of reference diminishes rapidly towards the stack bottom. The plot also indicates that most document hits occur at stack positions that are less than 10% of the stack depth.

The results of an alternative statistical measure presented in [15] for quantifying the degree of temporal locality using stack distances are shown in Table 4.8. The results show that mean and median stack distances are 2262 and 275, respectively.



(a) (b)

Figure 4.7: Temporal Locality Characteristics: (a) LRU Stack Depth Histogram for 3000 Stack Positions; (b) PDF of the Overall LRU Stack Distances

Since the mean is 4 times greater than the median, it indicates some document hits near the stack bottom, thus skewing the mean stack distance. By normalizing the mean and median stack distances[9], document hits occur within 0.022% from the top of the stack, while about 50% of document hits occur within 0.003% from the top of the stack. The coefficient of variation and standard deviation of stack distances are 2.98 and 6,748, respectively. These statistics suggest some widely spread variation in stack distances relative to the size of the mean stack distances. However, there is evidence of stronger temporal locality in this workload as compared to other similar workloads characterized in [7] [11]. This suggests that the impact of upstream caching on the temporal locality in aggregate server workload might be less significant than first assumed.

One reason for this might be that the "aggregation effect", which means the

[9]The normalized mean and median stack distances are a relative measure of temporal locality suggested by Barford *et al.* [15]. The measures are calculated by dividing the mean and median stack distances by the total number of documents requested more than once.

Table 4.8: Statistics Measured for the LRU Stack Distances

Description	Estimated Value
Trace Length	56,065,061
Number of Unique Documents	101,007
Number of Re-Referenced documents	88,650
Mean Stack Distance	2,262
Median Stack Distance	275
Normalized Mean Stack Distance	0.0224
Normalized Median Stack Distance	0.0031
Std. Dev. of Stack Distance	6,748
Coefficient of Variation	2.98

multiplexing of individual request streams from multiple clients, is stronger than the "filtration effect". It may also be that most document requests by-pass upstream caches because the clients fail to configure their browser caches. As a result, a least recently used (LRU) cache management policy may manage server resources effectively.

4.3.4 Inter-Reference Times

Jin and Bestavros [53] use the probability density function (PDF) of inter-reference times to characterize temporal locality in document referencing. The intuition for this approach stems from a view that (i) the higher the probability for smaller inter-reference times, the stronger the temporal locality, and (ii) the lesser the probability for smaller inter-reference times, the weaker the temporal locality. The inter-reference times for individual unique documents in the workload were computed and combined to form the aggregate inter-reference time. A probability distribution plot of the aggregate inter-reference time of documents requested more than once is shown in Figure 4.8.

The graph reveals that the probability of referencing a document decreases rapidly with time. This is because highly popular documents tend to be requested frequently, and thus will exhibit shorter inter-reference times, while the less popular documents will exhibit longer inter-reference times. There is however, a break in the

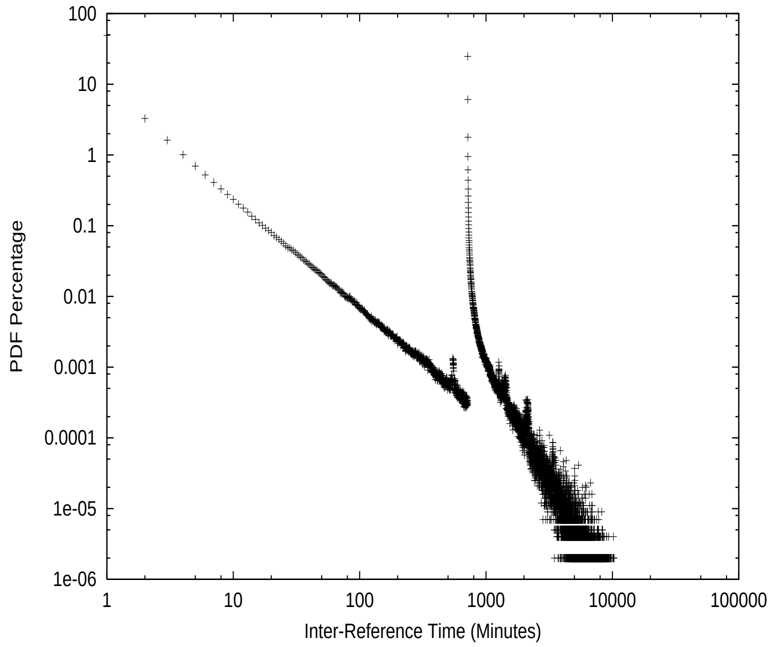


Figure 4.8: Inter-Reference Times Probability Density Function Plot

distribution of inter-reference times around 24-hour (1440-minute) intervals. This is likely due to Web user referencing patterns or crawlers operating on a daily (24-hour) basis. However, the inter-reference times of proxy workloads characterized in [27] [53] [61] did not show this feature probably due to the specific properties of proxies[10]. The observation of constant inter-reference times as shown by a spike around the 24-hour (1440-minute) mark suggests the presence of non-human referencing behaviour.

Further analysis indicates that the non-human clients are crawlers that visit the server around midnight with a huge number of requests that correspond to the same number of unique documents accessed. The continuous visitation of such non-human clients to previously accessed unique documents leads to the spike around the 1440-minute mark. Moreover, a larger proportion of the inter-reference time falls at the tail of the distribution. The presence of this non-human referencing behaviour

[10] A proxy receives requests from multiple clients and caches documents from multiple servers across the Web.

implies that the use of inter-reference times as a measure of temporal locality as suggested in [53] may not be applicable at the server.

In order to understand the intrinsic behaviour of inter-reference times, apart from using this as a measure of persistence, a separate analysis was conducted to investigate whether or not individual popular documents have inter-reference times that are independent and exponentially distributed. In a nutshell, the analysis is meant to find out if the document request arrival process at the server is Poisson. Since a fixed-rate Poisson test cannot be carried out on the aggregate workload because of the time-of-day effect [7] [10] [61] [77], the aggregate document inter-reference time was excluded from the analysis. Using the method postulated in [77] but adapted to Web traffic analysis by Arlitt and Williamson [10], Figure 4.9 indicates that popular document[11] inter-reference times can be modeled effectively by homogeneous Poisson processes at ten-minute and one-hour intervals. Along the x-axis is the percentage of tested intervals that passed the A^2 goodness-of-fit test for exponential distribution at a 95% confidence level, and along the y-axis is the percentage of intervals that passed the autocorrelation test for independence. The broken lines correspond to a 95% pass-rate. Each plotted point on the graph represents individual popular documents. If the request arrival process for each of the popular documents considered was truly Poisson, a stronger concentration of plotted points would appear at or above the intersection of the broken lines.

Figure 4.9 indicates that the request arrival process for popular documents is closely modeled as Poisson over ten-minute and one-hour intervals, even when the 95% pass-rate on average for independence and exponential distribution tests was not statistically satisfied. The number of popular documents that satisfied the condition at ten-minute intervals is 248 with 95% and 88% pass-rates on average are observed for independence and exponential distribution tests, respectively. For one-hour intervals, 1045 popular documents satisfied the condition. The average pass-

[11] A popular document is defined as a document that is referenced at least 50 times in a time interval, in at least 25 different (non-overlapping, but not necessarily contiguous) time intervals. Arlitt and Williamson suggested this definition in [10] for busy documents.

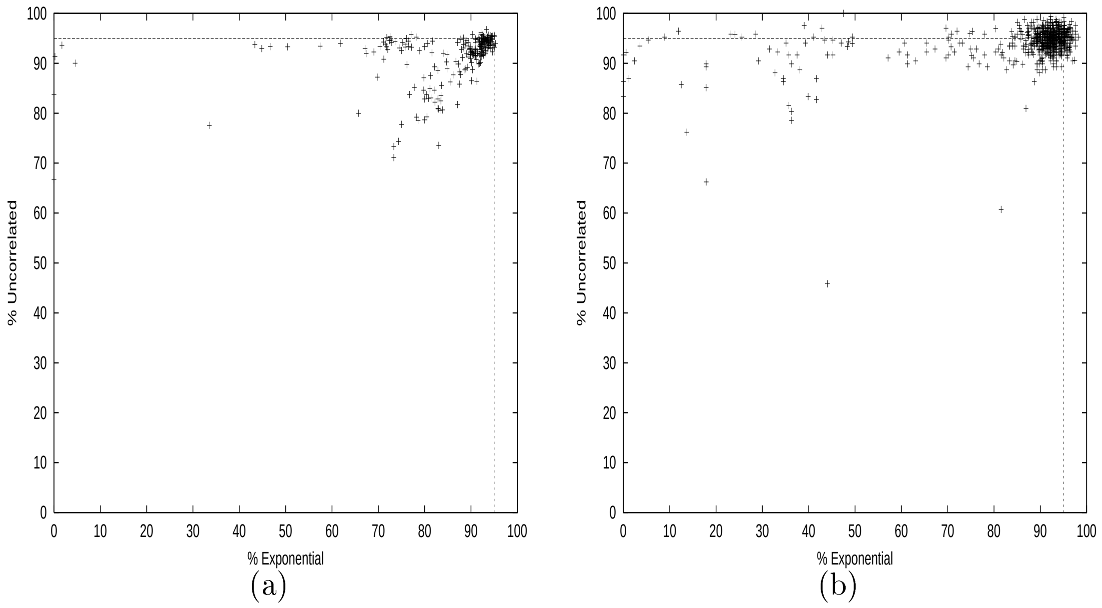
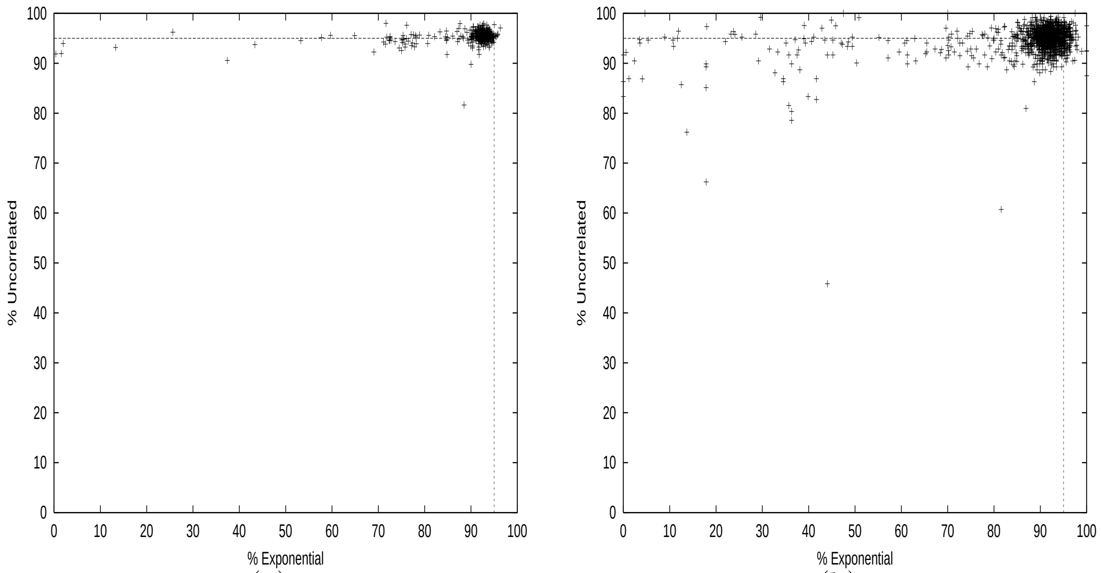


Figure 4.9: Poisson Inter-Reference Time Tests For Documents Referenced at Least 50 Times in a Time Interval, in at Least 25 Non-Overlapping Time Intervals: (a) Ten-Minute Intervals Plot ; (b) One-Hour Intervals Plot

rate for the independence test is 87%, while the average pass-rate for the exponential distribution test is 94%. The fall below 95% in both cases of average pass-rates is due to a negligible number of popular documents that have interval pass-rates that are far from 95%. Since a stronger cluster of plotted points appears at the upper right-hand corner of Figure 4.9, document reference processes are closely modeled as Poisson at ten-minute and one-hour intervals.

An attempt was also made to re-define popular documents either by reducing or increasing one or both parameters (i.e., the number of times a popular document is referenced in a time interval, and the number of non-overlapping sessions) to see if the results change. For instance, the threshold value for document references per time interval was reduced to half (25 times), in at least 25 different (non-overlapping, but not necessarily contiguous) time intervals. The number of popular documents that satisfied the condition for ten-minute intervals is 425, while 1,416 popular documents satisfied the one-hour intervals. More closely clustered points appear at the



(a) (b)

Figure 4.10: Poisson Inter-Reference Time Tests for Documents Referenced at Least 25 Times in a Time Interval, in at Least 25 Non-Overlapping Time Intervals: (a) Ten-Minute Intervals Plot; (b) One-hour Intervals Plot

upper right-hand corner as shown in Figure 4.10, revealing clearly that the document reference process is indeed Poisson for both time intervals.

The results show that individual document references at the server can be closely modeled by a Poisson arrival model at ten-minute and one-hour intervals. A ten-minute Poisson arrival model indicates that the arrival rate can change six times each hour rather than remaining constant throughout the hour. The possible reason why the arrival times to popular documents on a busy server are independent and exponentially distributed can be associated with multiple requests from different clients. Because repeated references of documents from single clients would have passed through upstream caches, timestamps of documents accessed at the server mostly come from distinct clients. The behaviour has been observed in [7] [10] to be responsible for the uncorrelated behaviour of document referencing behaviour during sessions. Similar reasons have been conjectured for TELNET connection arrivals and FTP session arrivals [77]. These observations differ from [10] because inter-reference times for popular documents can be modeled at one-hour and ten-minute intervals.

4.3.5 Summary

The salient properties of document referencing behaviour for a busy server were investigated. The results show that an extremely small percentage of documents are responsible for most references at the server. This behaviour influences the popularity profile of documents not to conform to Zipf's law. The correlation measure shows little preference for small or large documents, while large documents weakly influence the volume of data transferred over the network. The probability density function of document sizes for this workload shows a preference for documents between 1-10 kB while documents between 1-40 kB are responsible for most volumes of data transferred over the network. The request arrival process for popular documents is observed to follow a Poisson model at ten-minute and one-hour intervals.

4.4 Performance Implications of Aggregate Web Traffic Characteristics

The characteristics of aggregate Web server traffic indicate that caching strategies are most suitable to manage the server resources effectively. There are two foremost goals of improving the performance at the Web server. The first goal is to improve the request throughput—the number of requests that can be served in a given unit of time, while the second goal is to improve the data throughput—the amount of data (in bytes) that can be transferred in a given unit of time. Arlitt and Williamson [10] reported that optimizing the first goal in terms of cache policies at the server does not necessarily optimize the other. Thus, the choice between cache policies that improve either request throughput or data throughput depends on which resource at the server is the bottleneck: CPU cycles or disk I/Os. In the case of this workload, however, the results show that the choice does not necessarily depend on which resource is the bottleneck. In fact, the choice hinges on the size of document cache placed at the Web server.

In order to explain the disappearance of the tradeoff between request throughput and data throughput, a statistical measure proposed by Barford *et al.* [15] is employed. This measure shows the existence of a relationship between the performance of an infinite-sized cache and the correlation between two variables as expressed in Equation (4.6). Another approach is the graphical method used in [10] to illustrate the tradeoff between the choice of cache policies that reduce the number of requests presented to the server and cache policies that reduce the network traffic.

The first approach shows that the correlation between the size of document and the frequency of references to the document is very close to zero. This implies that caching of small and large documents is essential to optimize request throughput. Any attempt to sacrifice small documents over large documents may not optimize request throughput at a busy server. Similarly, the correlation between the size of a document and the volume of bytes transferred that accrued to the document is also very close to zero. This also implies that caching large documents over small documents may not necessarily optimize data throuphput. Therefore, both large and small documents must be cached in order to optimize request throughput or data throughput at a busy server.

For the second approach, the graphical approach uses the CDF to illustrate the relationships between document sizes, and frequency of document references, actual and weighted[12] data transferred for each referenced document. The document size is on the horizontal axis and the cumulative percentage on the vertical axis. Using this approach, Figure 4.11 can be used to answer whether cache policies should preserve documents which are frequently requested or documents that produce the most bytes transferred.

Figure 4.11(a) reveals that 55.81% of all documents requested from the server were less than 10 kB. About 81% of all the requests to the server were for documents

[12]Weighted data transferred by each unique document is calculated by multiplying the number of references to this document by the estimated document size. This is done to estimate the volume of data transferred from the server backing store.

in this category. Thus, caching a large number of small documents (that are less than 10 kB) may allow the server to optimize request throughput. However, 28% of the total volume of bytes transferred over the network were generated by documents less than 10 kB. But if the cache size is large enough to preserve documents between 10-40 kB, which accounted for 38.54% of all documents available at the server with 17.58% of total requests presented to the server and 49.60% of the volume of bytes transferred over the network, then data throughput would be optimized as well. This also implies that it is not necessary to trade request throughput for data throughput, except if the cache is very small.

In addition, Figure 4.11(a) shows that documents over 100 kB may be prevented from entering the cache since there is a wide gap between the actual bytes transferred and the weighted bytes transferred. This implies that documents which fall into this category were modified or were not fully downloaded before the requests were aborted by the Web users. Upstream caching of these documents would lead to a gross underutilization of the cache space since the server would still be required to service incomplete documents from the disk system. Because of these problems associated with documents greater than 100 kB, cache policies that seek to cache large documents may be ineffective at the server. The implication for Web page designers is to keep document sizes at the server small, and no greater than 100 kB.

Figure 4.11(b) shows the results of cache policies that seek to optimize request throughput at the server. In this graph, the top line represents the percentage of requests serviced from the cache and the next two lines represent the percentage of potential bytes transferred (in terms of transfer sizes and document sizes) as a result of requests being serviced from the cache. The last straight line represents the percentage of the maximum-needed-cache size available at the server. Using these cache policies, for example, 10% of the most frequently accessed documents occupied 4.56% (129 MB) of the maximum-needed-cache size and accounted for 96.05% of the total requests satisfied from the cache. This proportion of total requests contributed 84.35% of the total bytes transferred by the server. Moreover, a linear growth in cache size is required for every additional most frequently requested document to

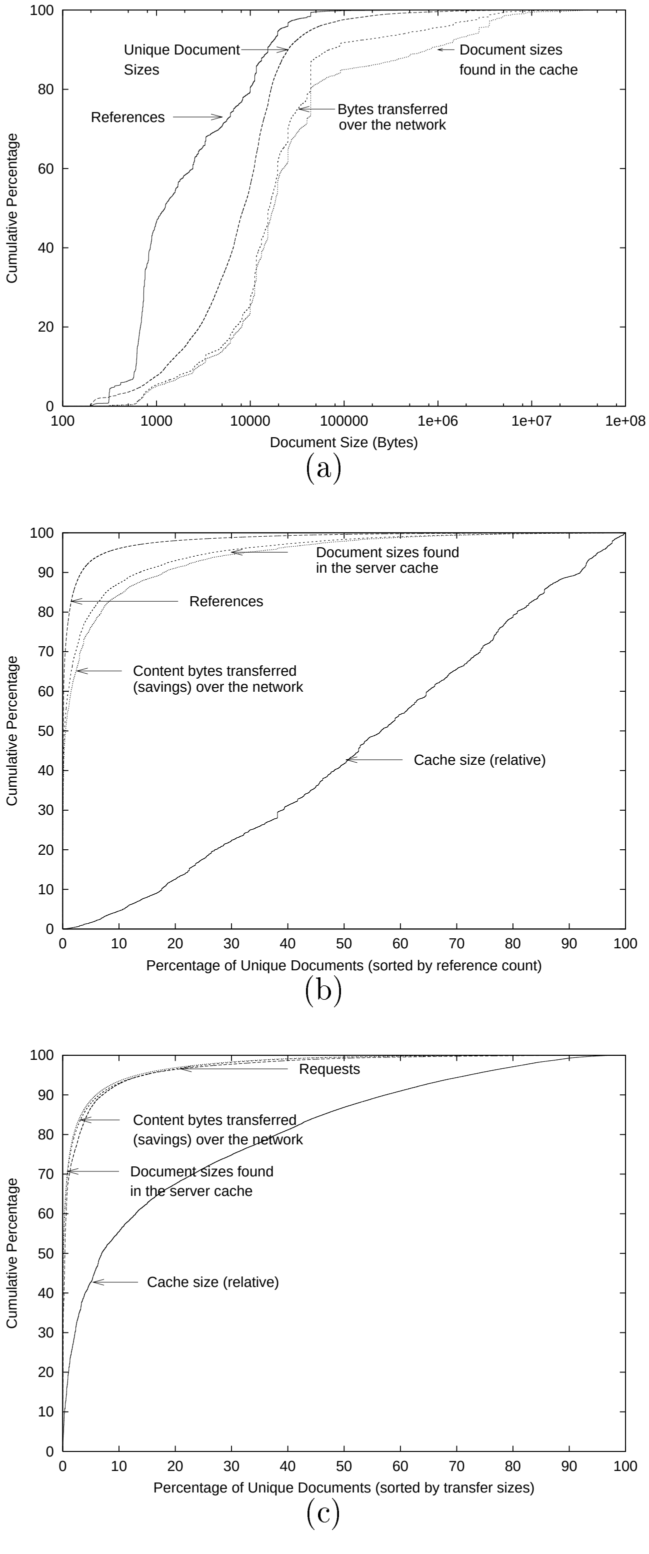


Figure 4.11: Caching Implications for Web Server Resource Management

be preserved in the cache.

On the other hand, Figure 4.11(c) shows the results of cache policies that seek to optimize data throughput at the server. The labeling of this graph is similar to Figure 4.11(b). Comparing the performance results in Figure 4.11(c) with those in Figure 4.11(b) suggests that caching documents on the basis of most bytes transferred will yield the best performance in terms of both request throughput and data throughput. For example, 10% of the documents that were responsible for most bytes transferred occupied 55.49% (1,564 MB) of the maximum-needed-cache size and accounted for 93% of the total requests satisfied from the cache. This proportion of total requests contributed 93% of the total bytes transferred by the server. However, a geometric growth in cache size is required for every additional most bytes transferred document to be preserved in the cache.

The results show that cache policies that seek to optimize request throughput use about 5-22% of the maximum-needed-cache size while those that optimize data throughput use about 55% of the maximum-needed-cache size. This clearly indicates the existence of a tradeoff in cache size between policies. In addition, policies that optimize data throughout may not need to sacrifice "cache hits" for many (small) document requests if a large cache is placed at the server. The relevance of these results to resource management of a busy server is that if a large cache (about 50% of the maximum-needed-cache size) is managed by policies that optimize data throughput, CPU cycles and disk I/Os can be managed effectively at the same time. It also implies that these cache policies can be used in a situation where the resource that causes the bottleneck cannot be ascertained. But the performance of a small cache placed at a busy server hinges on the determination of which resource is the bottleneck: CPU cycles or disk I/Os. Cache policies that optimize request throughput are still the most effective for managing the resources at a busy server because they require a small document cache.

Table 4.9: Summary of Web Server Workload Characteristics

Characteristic	
Success Rate	About 80% of requests are successful, and these account for 98% of the volume of bytes transferred.
File Types	HTML and Image documents account for 96% of valid requests.
Mean Transfer Size	$\approx$ 28 kB
Median Transfer Size	$\approx$ 9 kB
One-Timer Client	$\approx$ 5.36% of total clients at the server. Accesses account for 0.10% of the bytes transferred.
One-Timer Document	About 12% of documents are referenced once. This accounts for 9% of total document disk space and $\approx$ 0.02% of total document requests to the server.
Size Distribution	Document and transfer size distributions are heavy-tailed.
Concentration of References	10% of documents accessed account for 96% of of valid requests and 93% of the bytes transferred.
Inter-Reference Times	Web crawlers can invalidate the measure of temporal locality using inter-reference times. Document inter-reference times are independent and exponentially distributed at 10-minute and 1-hour intervals.

4.5 Summary

This chapter has presented the characteristics of the aggregate server workload from the perspective of multiple clients. The results show that some of the characteristics of Web server workloads have changed over time, while others have not. Table 4.9 highlights some of the observed characteristics. Examining the performance implication of these characteristics, a tradeoff exists in cache size between cache policies that optimize request throughput and cache policies that optimize data throughput. The performance results also show that it does not matter which resource is the bottleneck at a busy server if a large server cache is managed by policies that optimize data throughput. For a small cache, the determination of which resource is the bottleneck is required: CPU cycles or disk I/Os. Cache policies that optimize request throughput, however, are cost effective (in terms of cache size) for resource management of a busy server. This is because such policies yield the best cache size-to-performance ratio at relatively small cache size. Moreover, Web page designers should keep document sizes at the server small, and no greater than 100 kB to improve performance.

Chapter 5

Web Client Workload Characterization

In Chapter 4, the aggregate traffic from multiple clients was characterized as the first level of the Web server access hierarchy. This chapter examines the characterization of traffic from individual clients (i.e., the second level of the Web server access hierarchy) that accessed the Web server over the period studied. The key issue is to identify the behavioural characteristics of clients that can be useful for managing the server resources. Section 5.1 deals with the client referencing behaviour at the server. Section 5.2 studies in detail the behavioural characteristics of four selected client workloads, while Section 5.3 investigates the document referencing behaviour by analyzing sessions within selected clients. Section 5.4 gives a summary of the chapter.

5.1 Client Referencing Behaviour

This section deals with the nature of document requests and the volume of data transferred by all the clients that accessed the server. For understanding the behaviour of request streams emanating from individual clients at the server, Section 5.1.1 characterizes the distribution of the number of requests, and the volume of data requested by each client; Section 5.1.2 studies the concentration of clients; Section 5.1.3 discusses document sharing behaviour among the clients that accessed the server and Section 5.1.4 investigates the temporal locality of client requests at the server. Section 5.1.5 gives the summary.

5.1.1 Distributions of Requests and Data Generated by Clients

A small percentage of the clients[1] that accessed this server were responsible for most of the document requests and data (in bytes) transferred as shown in Figure 5.1. In particular, the figure illustrates that most clients generated fewer than 100 requests within the duration of the period studied. For instance, the CDF of the number of requests generated by each client shows that more than 99% of the clients accessed the server fewer than 1000 times, while fewer than 1% of the clients accessed the server more than 10,000 times. This distribution is similar to that reported by Mogul [66]. A similar observation can be made for the CDF of the data transferred by individual clients at the server. For example, Figure 5.1(c) shows that about 25% and 99% of clients were responsible for less than 300 kB and 10 MB of data transferred, respectively.

The distributions of document requests and the volume of data transferred by individual clients are subjected to the heavy-tailed test using the LLCD plot as explained in Section 4.2.1. The linear relationship observed in Figures 5.1(b) and 5.1(d) suggests that the distributions are heavy-tailed. Linear regression estimates of the upper tail index are $\alpha = 1.44$ ($R^2 = 0.99$) for the requests generated and $\alpha = 1.24$ ($R^2 = 0.96$) for the volume of data transferred by individual clients. This implies that the Pareto distribution can be used to characterize the number of document requests and the volume of data transferred by all clients at the server. Since the distribution of data transferred by clients is heavier-tailed than the distribution of document requests, there is a strong indication that managing server resources using individual client usage patterns may cause the server disk I/Os to be more of a bottleneck than the CPU cycles.

---

[1] A client is identified as a distinct host address in the server access log. All requests from the same host address in the server access log form the client traffic.

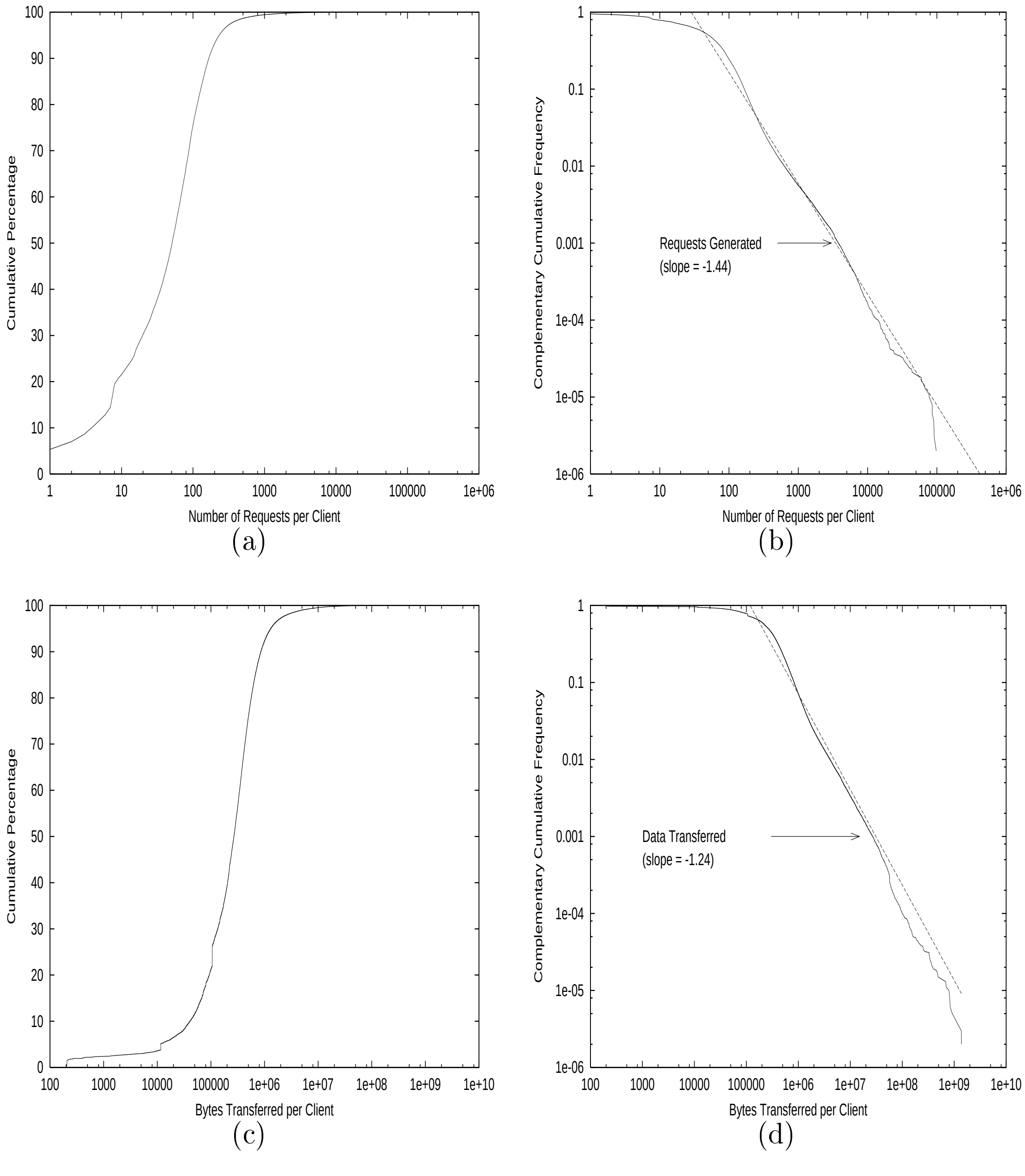


Figure 5.1: Client-Based Characterization: (a) CDF of the Number of Requests Per Client; (b) LLCD of the Number of Requests per Client; (c) CDF of the Volume of Bytes Transferred per Client; (d) LLCD of the Volume of Bytes Transferred per Client

5.1.2 Concentration of Clients

Concentration of clients is another important characteristic in determining the proportion of clients that are responsible for most requests and data transferred at the server. This characterization is quite different from the concentration of documents. Earlier studies of server [46] and proxy [3] workloads have characterized the concentration behaviour of clients. This characterization was done by sorting all the clients in the server workload individually in decreasing order, first based on the number of document requests generated, and second based on the volume of data transferred. The cumulative frequency of each sorted dataset was then plotted against the fraction of the total number of clients observed in the server workload. The results are shown in Figure 5.2. About 10% of the clients accounted for 45% of all document requests and 55% of the data transferred by the server. This result indicates that a few clients were responsible for most of the document requests received and the volume of data transferred by the server. Using Equation (4.7), a correlation of +0.90 is obtained between the number of requests generated and the volume of data transferred by each client. Since a few clients generate most of the requests and volume of data transferred, it is most likely that these clients are either Web proxies and firewalls that access the server on behalf of the clients, or Web crawlers that initiate requests on behalf of the Web users. Similar observations have been reported in [3] [66].

Another approach to understanding the distribution of clients at the server is to plot the log-log transform of requests generated by individual clients on the y-axis against the rank number of clients on the x-axis. If the slope of this plot is -1.0, then the pattern is in conformity with Zipf's Law. But if the slope is between -0.5 and -1.0, the distribution is Zipf-like. Section 4.3.1 elaborates on how to identify the Zipf distribution. Figure 5.3 shows that the slope of the best-fit line through the popularity plot of clients is -0.6, which implies that the popularity of clients at the server can be characterized as a Zipf-like distribution. This suggests that a few clients account for most accesses at the server, which by implication also means

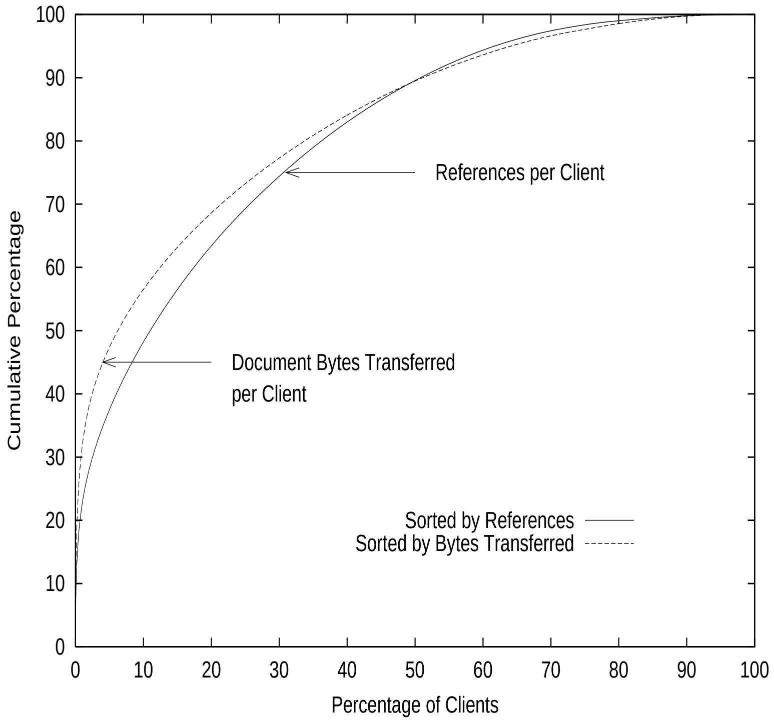


Figure 5.2: Concentration of Client Requests and Bytes Transferred

that a few clients account for most of the data transferred. The implication is that managing server resources based on this few client popularity[2] can be useful. The rationale for this is that the request stream emanating from a few clients is the source of the bottleneck at the Web server. In addition, a client-based strategy would be more effective to achieve Pareto-efficient management[3] of the server resources when requests are serviced based on individual client identifications.

5.1.3 Document Sharing Among Clients

In the previous section, the management of server resources based on the request stream that emanates from individual clients was suggested. The focus of this section

[2]Note that this implies the server would have to manage server resources based on the frequency of the request stream emanating from individual clients (i.e., the second level of Web server access hierarchy).

[3]This is the management of server resources such that no client accessing the server is worse off, but at least one client is made to experience better document response time.

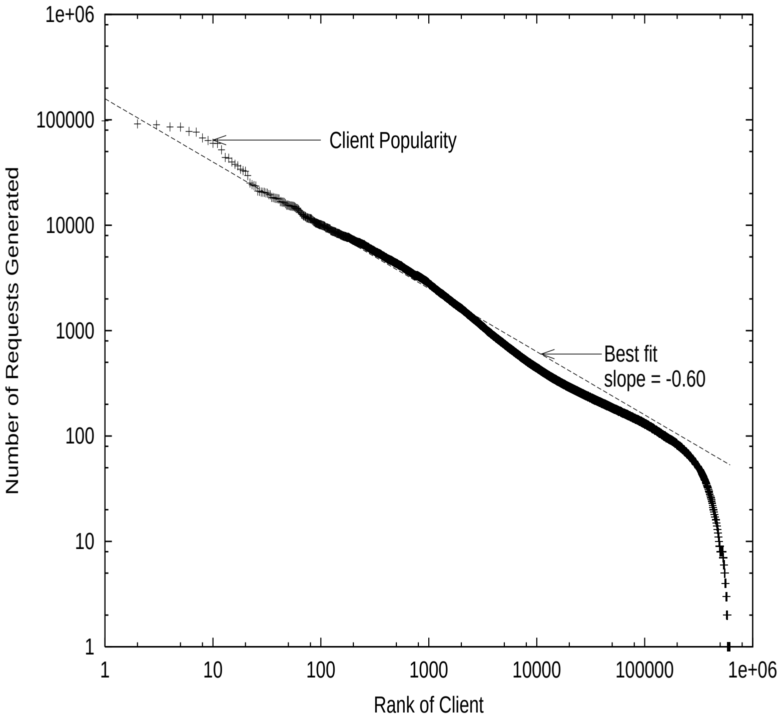


Figure 5.3: Frequency of Client References versus Client Rank

is to understand the impact of document sharing behaviour among those clients that accessed the server. Using Equation (4.7), a correlation of +0.87 is obtained as a measure of the relationship between the frequency of references to each unique document and the number of clients that accessed each unique document. This result implies strong document sharing among those clients that are accessing the server. The popularity of unique documents at the server is observed to be a result of many distinct client accesses.

Figure 5.4 depicts the CDF of the number of clients that accessed each unique document, and the CDF of the frequency of reference to each unique document at the server. The figure shows that most documents at the server were referenced by at least 10 clients and 95.48% of the overall requests received by the server are for 9% of the unique documents, which are individually accessed by more than 200 clients over the period of data collection. This observation shows that strong sharing of documents is due to a few popular documents (about 9% of the entire

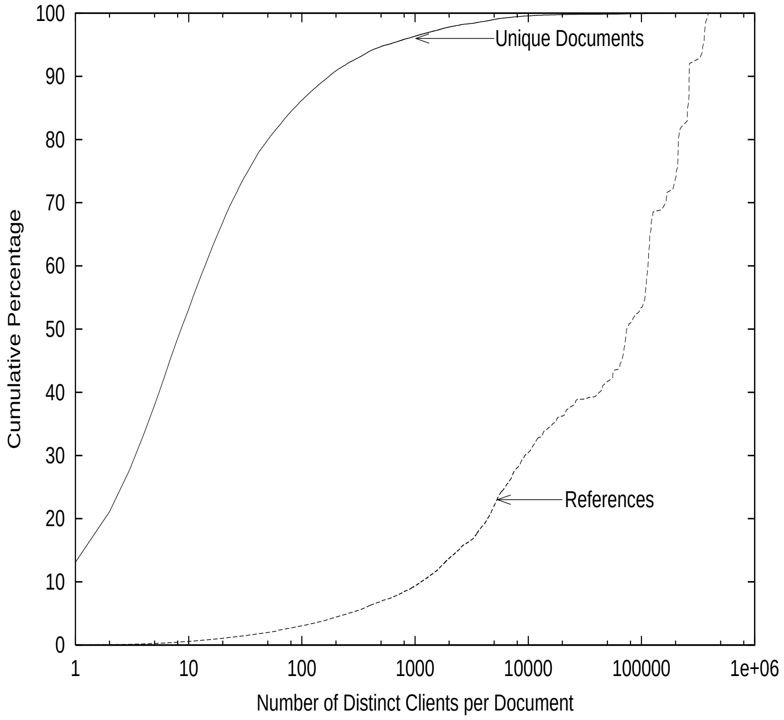


Figure 5.4: CDFs of Unique Documents and Frequency of References by Number of Clients per Document

unique documents) at the server. The LLCD plot of the number of clients that accessed individual unique documents was plotted (as discussed in Section 4.2.1 for the heavy-tailed test) and its upper tail index α is estimated to be 0.78. The result supports Figure 5.4, in that a few documents experience a strong concentration of document sharing among clients while the concentration of document sharing among clients for many documents at the server is insignificant. This behaviour is believed to be caused by the impact of upstream caching on individual client reference patterns at the server, as discussed in Section 2.2. That is, most repeated references made by clients to unique documents have been intercepted at the client and in the network as a result of browser caching and caching proxies, respectively.

A careful observation of the distribution of clients that accessed each unique document reveals the existence of a dichotomy in the behaviour of accessed documents—a few popular documents exhibit a significant document sharing property and many unpopular documents exhibit an insignificant document sharing property. The popularity behaviour of documents, however, gives a clue to what an effective resource management strategy at the server may need to harness for performance improvement.

5.1.4 Temporal Locality

The consequences of by-passing the document sharing behaviour of clients in order to manage the resources at the server using the request stream from individual clients are examined. These consequences can be analyzed in two phases. First, the persistence of the requests from individual clients and second, the persistence of document requests in the reference pattern of "typical" heavy-access clients are investigated. However, only the first phase is discussed, while the second phase is deferred to Section 5.2.

Temporal locality in client referencing behaviour is a measure of persistence, which means a client that was recently seen by the server is likely to be seen again in the near future. Since client referencing behaviour is different from document referencing behaviour, understanding the concept is important for the management of the server using the request stream from single clients. Table 5.1 shows the results of the quantitative measurement of temporal locality using the LRUSRM. Out of 585,959 re-appearing clients, the mean and median stack distances are 639 and 38, respectively. This means that the mean stack distance is 0.11% from the top of the stack and median stack distance is 0.01% from the top of the stack. The closeness of the mean and median stack distances to the top of the stack implies that the client reference pattern has strong temporal locality. The high coefficient of variation (16.51) suggests that the sequence of stack distances obtained using LRUSRM were more widely spread relative to the mean stack distance. These statistical values imply that many client references, though with fewer requests, are not persistent in generating requests; while a few clients, though with many requests, are persistent in generating requests.

Table 5.1: Quantitative Measurement of Temporal Locality

Description	Estimated Value
Trace Length	56,166,068
Number of Distinct Clients	618,935
Number of Re-Appearing Clients	585,959
Mean Stack Distance	639
Median Stack Distance	38
Normalized Mean Stack Distance	0.00108
Normalized Median Stack Distance	0.00006
Std. Dev. of Stack Distance	10,545
Coefficient of Variation	16.51

The result also shows that temporal locality of client referencing behaviour at the server is stronger than temporal locality of document referencing behaviour (as shown in Section 4.3.3). This implies that the individual request streams of the few clients that are most active can be exploited for managing the server resources effectively. However, the presence of this property can be useful only if there is a corresponding strong temporal locality in each of the request streams that emanates from a few heavy-access clients. It is hypothesized that the greater (lesser) the impact of upstream caching on each of the client reference patterns (as explained in Section 5.1.3), the lesser (greater) the effectiveness of any server resource management strategy based on individual reference patterns of the clients at the server.

5.1.5 Summary

The nature of request streams from all clients that accessed the server within the period studied was characterized. Important observations are the Zipf-like distribution of clients, the heavy-tailed property of the distributions of client accesses and bytes transferred, the strong temporal locality of client requests, and the strong document sharing behaviour among clients. Since a few heavy-access clients dominate the overall requests and the volume of data transferred at the server, the characterization of the reference pattern of those heavy-access clients is studied next.

5.2 Selection and Analysis of Single Clients

The second level of access traffic under the Web server access hierarchy scenario comprises requests that emanate from individual clients in the server workload. This section presents the characteristics of four heavy-access clients. The individual clients selected for further study have reference streams that range from thousands of requests over seven days, to hundreds of thousands of requests. This group of clients was selected based on the premise that the request stream from a few heavy-access clients is the source of bottleneck at the server. Since 1% of clients generated more than 10,000 requests, the four clients selected are considered to represent "typical" heavy-access clients at the server. For each of the four clients selected, Section 5.2.1 discusses the reference pattern; Section 5.2.2 analyzes the request stream; Section 5.2.3 characterizes the document size distribution; Section 5.2.4 looks for document concentration in the reference pattern; Section 5.2.5 studies the temporal locality in the reference pattern and Section 5.2.6 characterizes the distribution of client inter-arrival times. Section 5.2.7 gives a summary report.

5.2.1 Reference Patterns of Sample Clients

The reference patterns of the four selected clients are displayed in Figure 5.5. These clients are labeled according to their transcribed IP (host) address number in the server access log collected. The vertical axis represents a sequence of numbers assigned to each unique document in the client reference pattern as the document identity. This means that the maximum number plotted on the y-axis as a document identity corresponds to the number of unique documents in the client reference pattern. The horizontal axis represents the time since the start of the workload collection. In explaining the reference pattern, the terms "mostly-working" client and "mostly-surfing" client are adopted from [34] [36]. A mostly-working client means that repeated references to previously visited documents form its reference pattern, while a mostly-surfing client rarely has repeated references to previously visited documents in its reference pattern.

Client 2597 is an example of a mostly-surfing client. It wakes up at fixed intervals of time to traverse the same number of unique documents each time it visits the server. According to Figure 5.5, the client reference pattern reveals that the client visits the server nine times within the seven-day period. This pattern of references to unique documents each time the client visits the server is indicated by the parallel reference pattern. The parallel reference pattern implies that the client accessed only the same unique documents each time it visits the server. Client 2597 seems to have an entirely different pattern from those user patterns reported in [36]. It is speculated that this reference pattern is due to a non-human client such as a Web crawler (or search agent) [32].

Client 485 is an example of a mostly-working client. The densely populated document reference pattern confirms its working behaviour. The client reference pattern shows that there are more document accesses during the weekday than during the weekend. The rate at which new documents were accessed by this client decreases as each day passes. This indicates that some element of locality is present in the reference pattern. Client 485 is speculated to be a firewall or relay that accesses the server on behalf of a group of users [66].

Client 374 is another example of a mostly-working client. The presence of faintly scattered points implies a low level of working behaviour on the URLs requested. Some locality may also be found in its reference pattern. Client 374 is conjectured to be a time-sharing system or caching proxy. This is because of the sparse reference pattern that suggests the impact of upstream caching (either at the browser or in the network) on its request stream. A similar observation has been made in [66].

Client 22 is another example of a mostly-surfing client. Its reference pattern is, however, different from that of Client 2597. Client 22 seems to have spent most of its time searching for new documents on the first day. When the document was found, the client began to make repeated requests to Web pages within the site as it viewed and worked for a while. Catledge and Pitkow [30] described this behaviour as *searching* and Cunha *et al.* [36] referred to it as *mostly-surfing* when viewed from the client side. Client 22 is therefore speculated to be a caching proxy because no

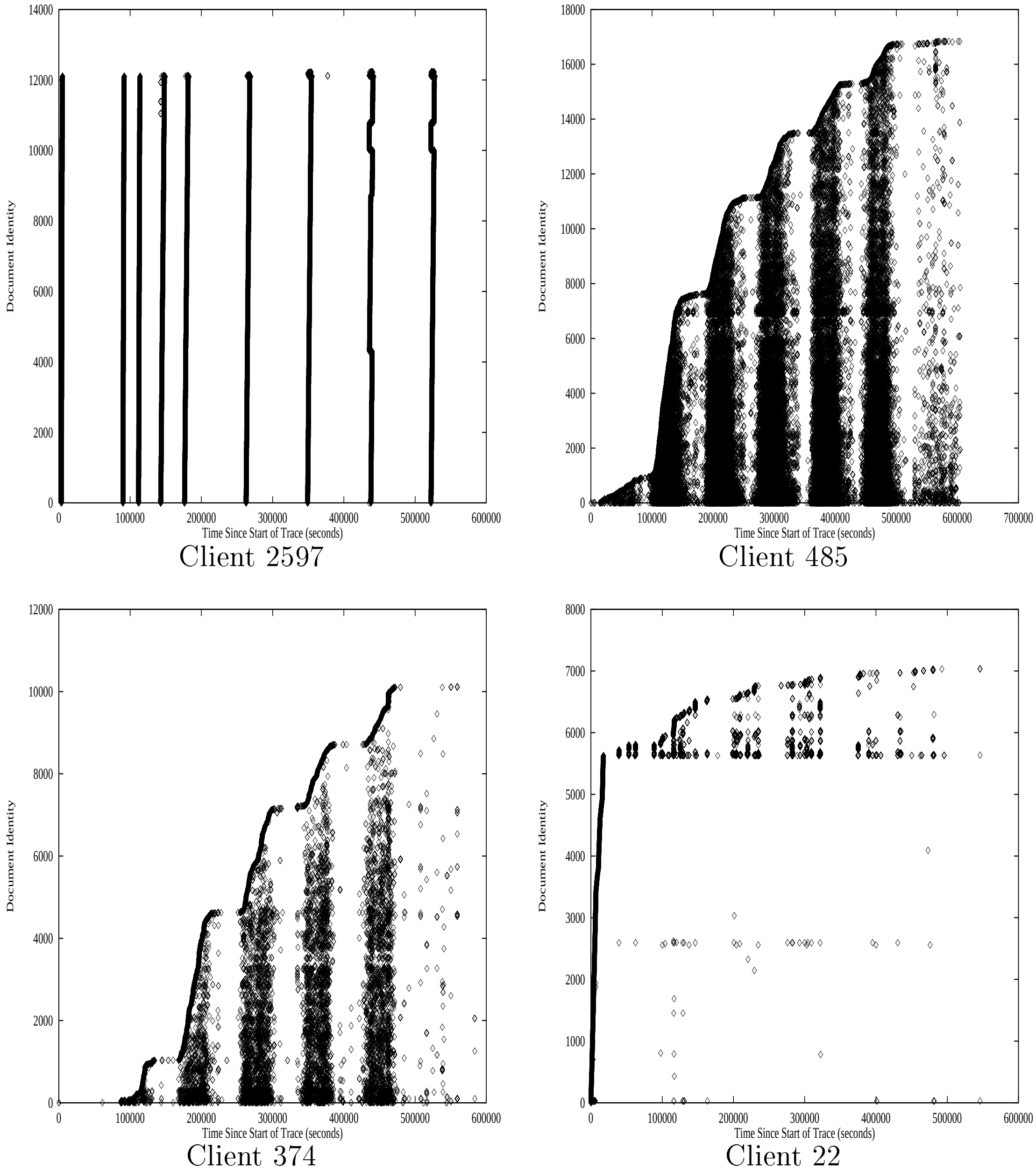


Figure 5.5: Reference Patterns of Four Selected Clients

reasonable user can access 7,000 documents within a single day. Since a caching proxy is usually empty on the setup day (i.e. first day), document misses due to cold cache (start-up period) must be serviced by the server. Subsequent requests will, however, be serviced from the caching proxy with few documents re-referenced at the server afterwards. This may be the cause of 7,000 request misses in the first day. The scanty document re-references in the client reference pattern after the first day can be due to capacity and coherence misses[4] at the client.

A careful observation of the four selected clients shows that there are different classes of reference patterns exhibited by heavy-access clients: human and non-human reference patterns.

5.2.2 Workload Analysis of Sample Clients

The reduced access log of the four selected clients is summarized in Table 5.2. Client 2597 has the highest mean document requests per day, while Client 22 has the lowest. For the median document requests per day, Client 485 has the highest, while Client 22 also has the lowest. Clients 485 and 374 have median document requests per day greater than mean document requests per day. This observation was reported for the aggregate server workload and it suggests that these clients were affected by the weekend effect—an indication that both clients are likely to access the server on behalf of human users. The reason is because time-of-day and weekend effects have been reported to be correlated to the low affinity of Web users to access the Web outside working-day hours [2] [29] [61]. Figure 4.1 reveals the presence of this weekend effect in the aggregate server workload. The other two clients (Clients 2597 and 22) did not seem to be influenced by the weekend effect. The mean and median transfer sizes per day also show that Client 485 and 374 have the weekend effect. Client 485 has the highest number of unique documents in its workload, followed by clients 2597, 374 and 22, respectively. This means that the coefficient of variation of the documents accessed by heavy-access clients differs from one another, as shown

[4]Capacity misses occur due to a finite-sized cache at the client and coherence misses occur due to stale documents in the cache.

Table 5.2: Summary of Server Client Access Log Statistics (Reduced Data)

Item Description	Client Number			
	2597	485	374	22
Access Log Duration	7 Days			
Total Requests	109,385	97,859	21,122	9,594
Average Requests/Day	15,626	13,980	3,017	1,371
Median Requests/Day	12,139	19,140	4,671	677
Unique Documents	12,235	16,834	10,116	7035
Total Storage Size (MB)	196	333	173	146
Mean Document Size (Bytes)	15,977	19,765	17,061	20,691
Median Document Size (Bytes)	14,314	5,739	5,496	11,025
Coefficient of Variation	0.71	8.80	6.60	3.27
Total Bytes Transferred (MB)	1,750	919	272	158
Average Bytes/Day (MB)	250	131	38	23
Median Bytes/Day (MB)	194	171	47	4
Mean Transfer Size (Bytes)	15,997	9,393	12,883	16,499
Median Transfer Size (Bytes)	14,339	1,967	3,089	9,691
Coefficient of Variation	0.73	9.33	7.35	3.55

in Table 5.2.

Table 5.2 reveals that the storage size requirement for unique documents accessed by Client 485 is almost twice that of the other clients. This implies that there are large documents in the client reference pattern. The mean document size of Client 485 supports this claim. However, since the mean document size is greater than the median document size across the clients, a few large documents might have skewed the mean document size. Client 2597 has the highest volume of data transferred, while Client 22 has the least volume of data transferred. The mean transfer size of the four clients ranges from 9-16 kB, while the median transfer size is in the range of 2-14 kB. Client 2597 accessed fairly consistent document sizes, while Clients 485 and 374 accessed widely spread document sizes, as shown by the coefficient of variation. Similar observations can be reported for the transfer sizes of these clients. These results are all consistent with the hypothesis that clients 485 and 374 are human clients.

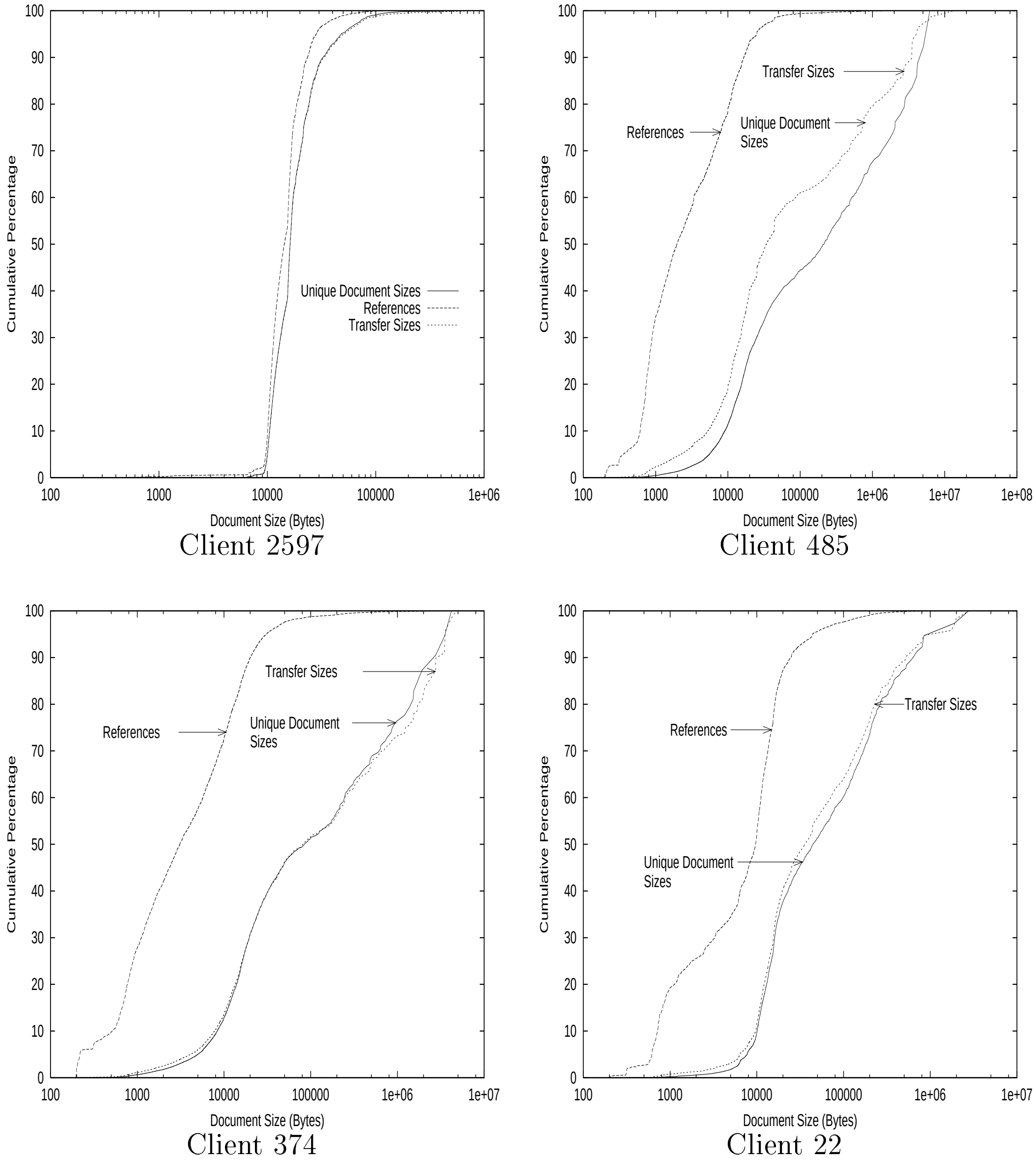


Figure 5.6: Document Sizes for Selected Clients

5.2.3 Document and Transfer Size Distributions

The question of whether or not there is a preference for smaller documents in client reference pattern is examined. Section 4.3.2 reports the observations of a similar investigation conducted for the aggregate server workload. The absence of this behaviour in client reference patterns can be used to differentiate between human and non-human clients. Previous workload characterizations at the client side have shown that human users prefer to access small documents over large documents [15] [36]. For Client 2597, references are distributed evenly across unique documents as shown in Figure 5.6. This makes the percentage of document references close to the percentage of data transferred and unique documents accessed. For instance, 0.01% of the unique documents less than 1 kB were responsible for 0.22% of the total requests and 0.01% of the data transferred. 92% of unique documents less than 40 kB accounted for 98.12% of the requests and 92.86% of the data transferred. The statistics reveal that Client 2597 does not prefer smaller documents over larger documents. This observation is quite different from the Web user's behaviour reported in [15], in which there is a weak preference for smaller documents. Therefore, Client 2597 is believed to be a non-human client.

Client 485 shows a preference for smaller documents in its reference pattern. For instance, 0.47% of the unique documents less than 1 kB received 34.43% of the requests and were responsible for 2.32% of the data transferred. Similarly, 11% of the unique documents less than 10 kB were responsible for 79.04% of the requests and 19.61% of the data transferred. The same conclusion is drawn for Client 374 and Client 22. For Client 374, about 12.84% of the unique documents less than 10 kB received 79.04% of requests and were responsible for 13.74% of the volume of data transferred. Client 22 has 9.72% of its unique documents less than 10 kB responsible for 52.65% of the document requests and 11.55% of the volume of data transferred.

These results indicate that the reference patterns of Client 485, Client 374 and Client 22 show a preference for small documents over large documents, while Client

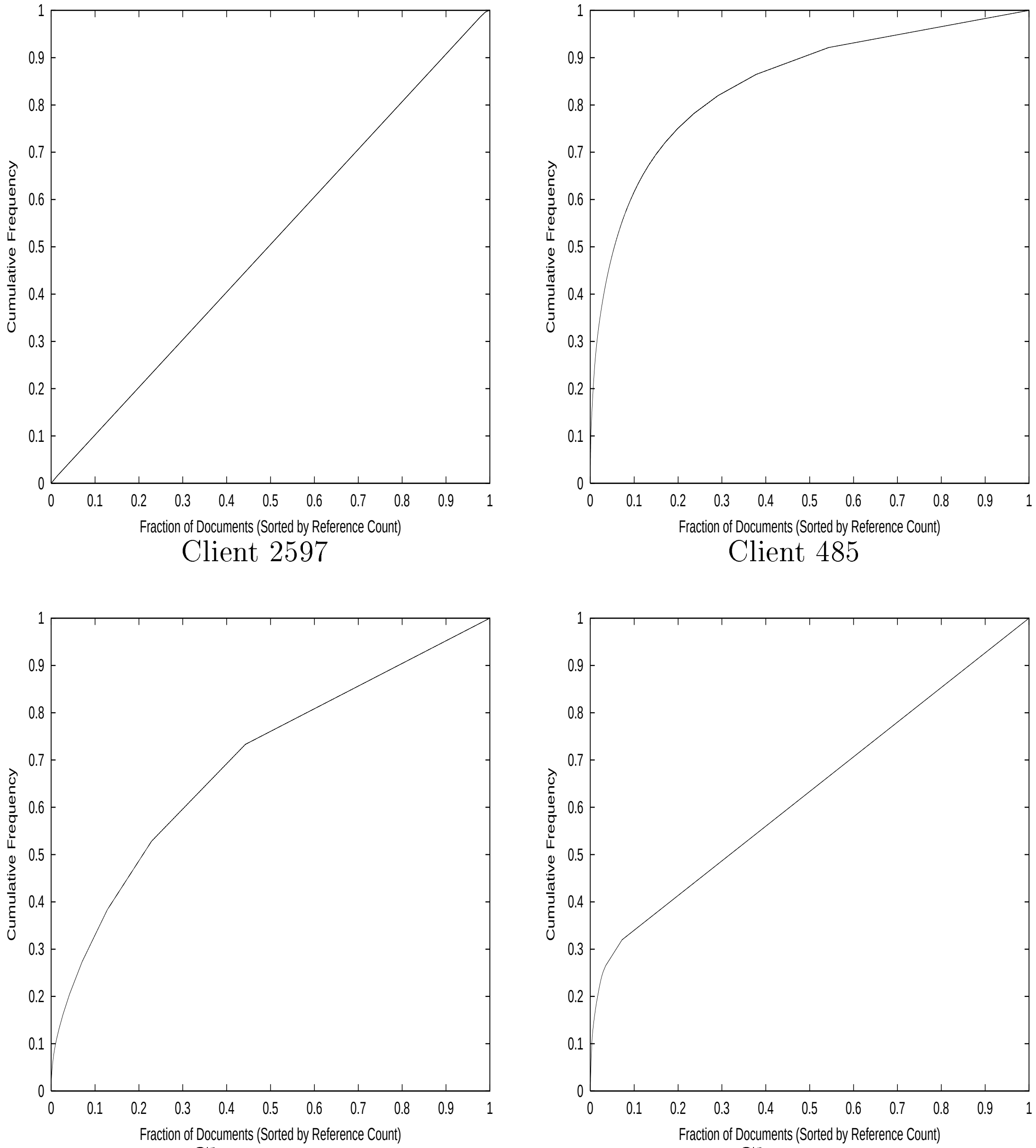
2597 does not. The presence of this behaviour in the aggregated server workload can be due to the proliferation of human clients at the server. However, the future server workload can change if the problems of locating information on the Web [49] and the growing interest in electronic commerce [19] continue to be solved by introducing non-human technology such as software agents[5] that mediate between human users and Web servers.

5.2.4 Concentration of Document References

The presence of non-uniform referencing behaviour in client reference patterns can be shown by arranging the list of unique documents in descending order based on how many times each document was accessed. A graph is then drawn by plotting the cumulative frequency of document accesses (from the most popular to the least popular documents) against the fraction of total unique documents accessed. The resulting graphs for the four selected clients are shown in Figure 5.7.

Client 2597 shows a uniform document referencing behaviour. This indicates the absence of document concentration. Client 485 has a non-uniform pattern: 10% of the unique documents account for 65% of the requests. Similarly, 10% of the unique documents accessed by Client 374 and Client 22 account for 35% of the requests. However, Client 374 and Client 22 have weak document concentrations. Figure 5.8 characterizes the popularity of documents in the four client workloads based on conformity with Zipf's law. Only Client 2597 deviates completely from Zipf's law, while the others display Zipf-like behaviour. These results are consistent with observations reported in Section 5.2.3. In particular, the document concentration behaviour of Clients 22, 374 and 485 is consistent with the hypothesis that these clients are likely to be human clients.

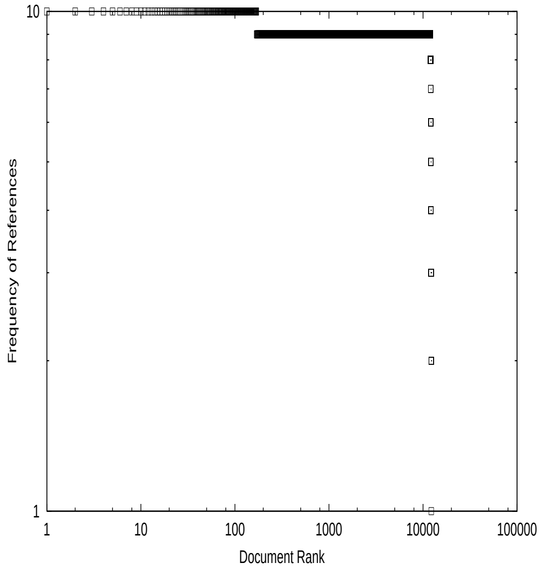
[5]A software agent is a piece of software which performs a given task using information gleaned from its environment to act in a suitable manner so as to complete a task.



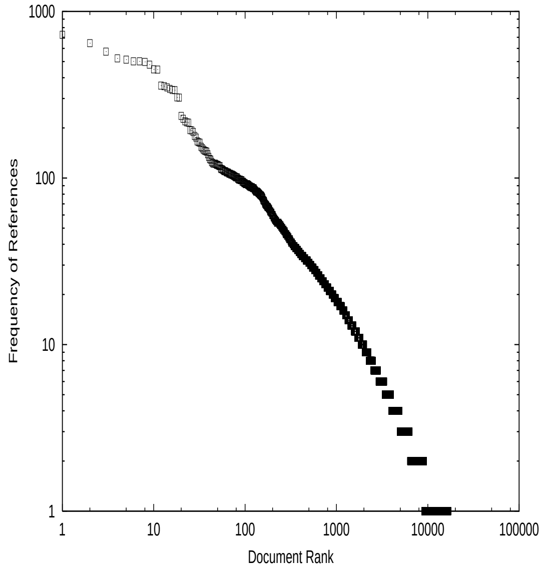
Client 2597 Client 485

Client 374 Client 22

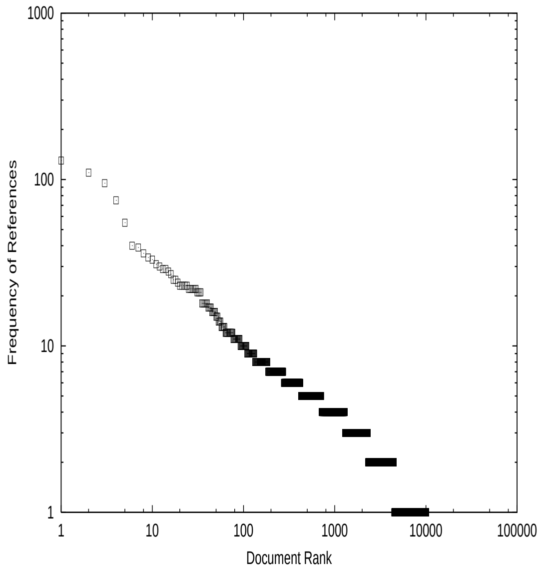
Figure 5.7: Concentration of Document Referencing



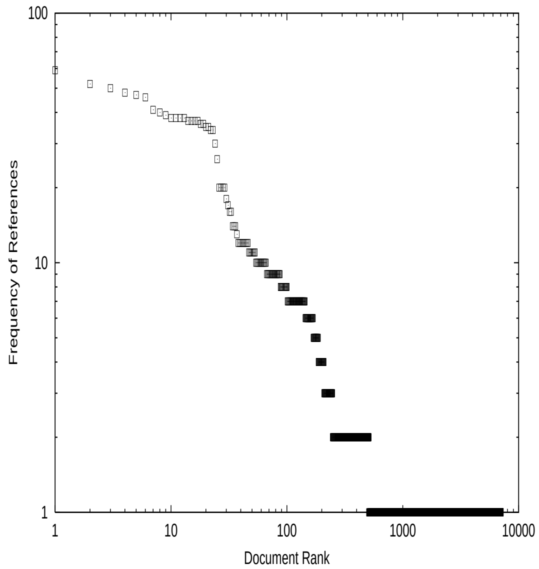
Client 2597



Client 485



Client 374



Client 22

Figure 5.8: Test for Zipf Distribution

5.2.5 Temporal Locality

The reference patterns of the selected heavy-access clients are analyzed to investigate temporal locality. The presence of this property in document referencing behaviour of clients is very crucial to the performance of any client-based resource management policy. A visual method of inspecting temporal locality in document reference pattern using the histogram plot was first used. The more (less) densely populated the distribution of stack distances towards the top of the stack, the stronger (weaker) is the temporal locality of document referencing behaviour in the client reference pattern. Figure 5.9 shows the histogram plots of stack distances obtained for each client using the LRUSRM. The plot shows that Clients 22, 374 and 485 have many references close to the top of the stack. Since Mahanti [61] suggests a maximum stack length of 1000 as a satisfactory measure of temporal locality, Figure 5.9 depicts the histogram plots of Clients 22, 374 and 485 using the maximum stack length of 1000. But Client 2597 uses an infinite stack length because most of its references are close to the bottom of the stack. Therefore, Client 2597 is judged to have poor temporal locality. On the other hand, Clients 22, 374 and 485 are speculated to have some degree of temporal locality, even though the clients have long-tailed stack distance distributions (meaning that many references are made to each of the 1000 stack positions).

In order to complement visual observations, Table 5.3 reveals the results of a quantitative measurement of temporal locality using the Barford *et al.* [15] approach. This quantitative approach assists in the relative measurement of temporal locality in the reference patterns of Client 22, 374 and 485. Temporal locality in the client reference pattern using this approach implies that mean and median stack distances are close to the top of the stack. The closer (farther)[6] these statistical measures are, the stronger (weaker) is the temporal locality. Client 2597 has a mean stack distance of 11,645 out of 12,234 re-referenced documents. This implies that the mean of stack distances is roughly 95% away from the the top of the stack. The median of stack

[6] The normalized mean and median of stack distances in Table 5.3 are used to measure the closeness of mean and median stack distances to the top of the stack.

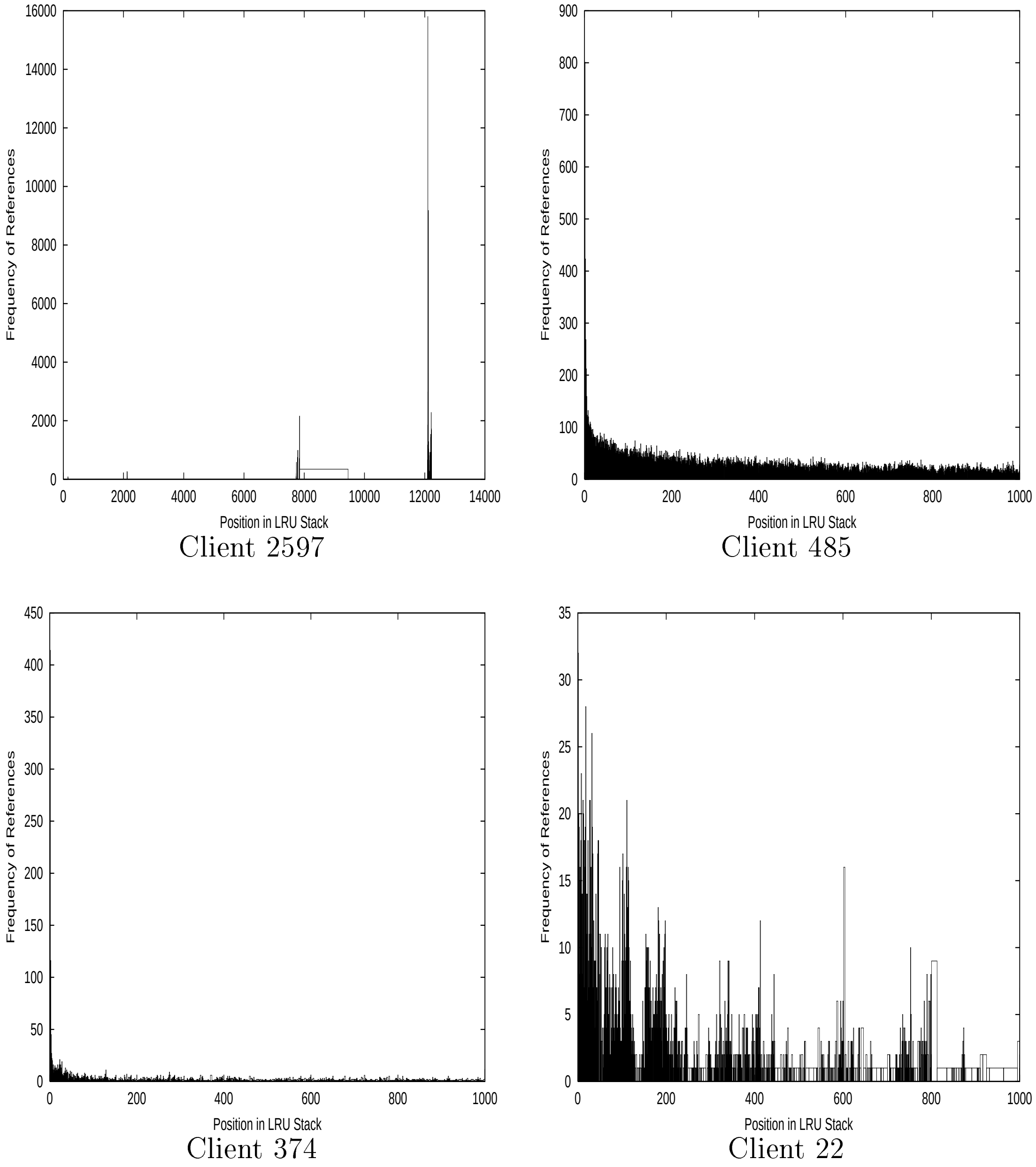


Figure 5.9: Distribution of LRU Stack Depths

Table 5.3: Information on Clients and Quantitative Measurement of Temporal Locality

Description	Client Number			
	2597	485	374	22
Trace Length	109,385	97,859	21,122	9,594
Total Stack Hits	97,150	81,025	11,006	2,559
Number of Unique Documents	12,235	16,834	10,116	7035
Document Referenced More than Once	12,234	9,143	4,478	509
Number of One-Timer Documents	1	7,691	5,638	6,526
Stack Distance Mean	11,645	2,536	3,466	266
Stack Distance Median	12,120	1,396	3,393	153
Stack Distance Standard Deviation	1,590	2,948	2,557	412
Stack Distance Coefficient of Variation	0.14	1.16	0.74	1.55
Normalized Mean of Stack Distances	0.95	0.28	0.77	0.52
Normalized Median of Stack Distances	0.99	0.15	0.76	0.30
% of One-Timer Documents	0.00	45.69	55.73	92.76

distances shows that 50% of document re-references occur at a stack distance that is 99% away from the top of the stack. Since temporal locality indicates the closeness of mean and median stack distances to the top of the stack, Client 2597 is judged to have poor temporal locality. This result is consistent with Figure 5.9. Similarly, Client 485 has a mean stack distance of 2,536, 28% away from the top of the stack; Client 374 has a mean stack distance of 3,466, 77% away from the top of the stack; and Client 22 has a mean stack distance of 266, 52% away from the top of the stack. The median stack distances for Clients 485, 374 and 22 are 1,396 (15% away from the top of the stack), 3,393 (76% away from the top of the stack) and 153 (30% away from the top of the stack), respectively. These observations imply that Clients 485, 374 and 22 have temporal locality. Based on the mean and median stack distances, Client 485 has the strongest temporal locality, followed by Client 22 and Client 374, while Client 2597 has the weakest temporal locality. The results are consistent with the visual histogram plots in Figure 5.9.

Table 5.3 shows that Client 22 has the highest variability of stack distances, while Client 2597 has the lowest variability of stack distances. Low variability of stack distances means that most re-referencing of documents occurs at almost the same stack positions, revealing the cyclic traversing behaviour of a fixed set of documents by the client. This document referencing behaviour has been identified for non-human clients as *crawling* [32]. A crawling pattern has been reported to have a negative impact on caching strategies [10] [11] with regards to the aggregate traffic from multiple clients received by the server. The reason is attributed to the need for the cache to preserve some space ahead of time, or continuously free some space, to accommodate the document working set of a Web crawler. However, separating the reference stream of a Web crawler client from the aggregate reference stream of multiple clients at the server in order to improve the performance of caching strategies may be difficult. On the other hand, the high variability of stack distances implies that most re-referencing of documents occurs at widely spread stack positions, suggesting that Clients 22, 374 and 485 do not reflect the non-human crawling behaviour.

With the assumption of perfect upstream caching for the request stream of clients, the impact of upstream caching is quantified and the possibility of caching heavy-access request streams is studied. Table 5.3 shows that Client 2597 would have made 12,235 requests (11.2% of its actual requests) if perfect upstream caching is assumed. This total number represents the number of unique documents in Client 2597 reference pattern. In addition, the proportion of one-timer documents to the overall number of unique documents is zero for Client 2597. Since a high (low) proportion of one-timer documents relative to the total number of unique documents in the client reference pattern is an indication of strong (weak) upstream caching effect, Client 2597 surprisingly appears not to be affected by this upstream caching effect. Similarly, Client 485 would have made 16,834 requests (17.20% of its actual requests) to the server, Client 374 would have made 10,116 requests (47.89% of its actual requests), and Client 22 would have made 7,035 requests (73.33% of its actual requests), respectively. These observations show that the reference pattern of Client 22 is not suitable for caching, while the reference patterns of Clients 374 and 485 do not clearly show this.

The ratio of one-timer documents to the overall number of unique documents is used to analyze the reference patterns of Clients 485, 374 and 22. The results show that Client 374 has 56% of its unique documents as one-timer documents, Client 485 has 46% of its unique documents as one-timer documents, while Client 22 has 93% of its unique documents as one-timer documents. Since proxy workloads have a high percentage of one-timer documents up to 70% [61] [62], the implication of these observations is that the reference pattern of Client 22 is not suited to caching, while the reference patterns of Clients 485 and 374 are. However, the absence of document concentration and temporal locality in Client 2597 (as discussed in Sections 5.2.4 and 5.2.5) indicates that its reference pattern is not amenable to caching, despite having the lowest percentage of one-timer documents. Web proxy caching strategies, on the other hand, may be applied to the caching of human clients. The conclusion, however, contradicts the observation of Mogul [66] that the reference stream of heavy-access clients at a busy server is not suitable for caching because of the high percentage of one-timer documents in the reference pattern of clients at the server.

Almeida *et al.* [5] characterized the request sequences arriving at Web servers and showed that temporal locality in the stack distance stream is well modeled by the Lognormal distribution. The procedure in Section 4.2.2 is used to match the stack distance distribution of each selected client with analytical models such as Gaussian Normal, Lognormal, Uniform and Exponential. The results obtained show that the Lognormal distribution does not match all the distribution of stack distance streams for the clients considered. Figures 5.10 and 5.11 show examples of the comparison between the CDF of client stack distances and the analytical model that gives the best goodness-of-fit using the probability plot. The coefficient of determination R^2 measures the variation in the stack distance explained by the analytical model and is shown in Table 5.4. The distribution of stack distances for Client 2597 closely matches Normal, Lognormal and Uniform models. For Client 374, the stack distance distribution matches Normal and Uniform models, while the distributions of stack distances for Clients 485 and 22 match Normal and Exponential analytical models. Conversely, the Andersen-Darling (A^2) goodness-of-fit

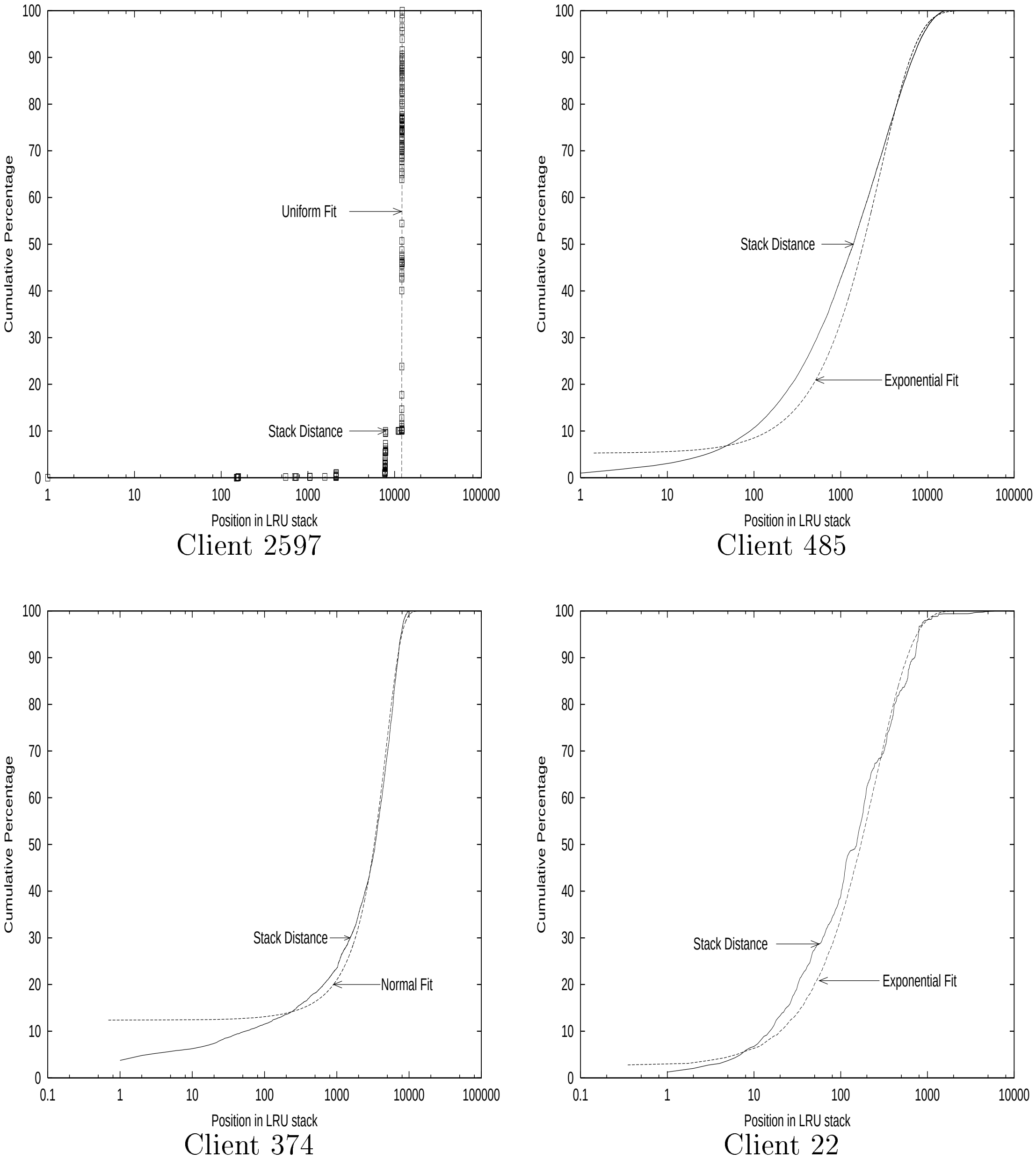


Figure 5.10: Distributions of Stack Distance in Client Traces

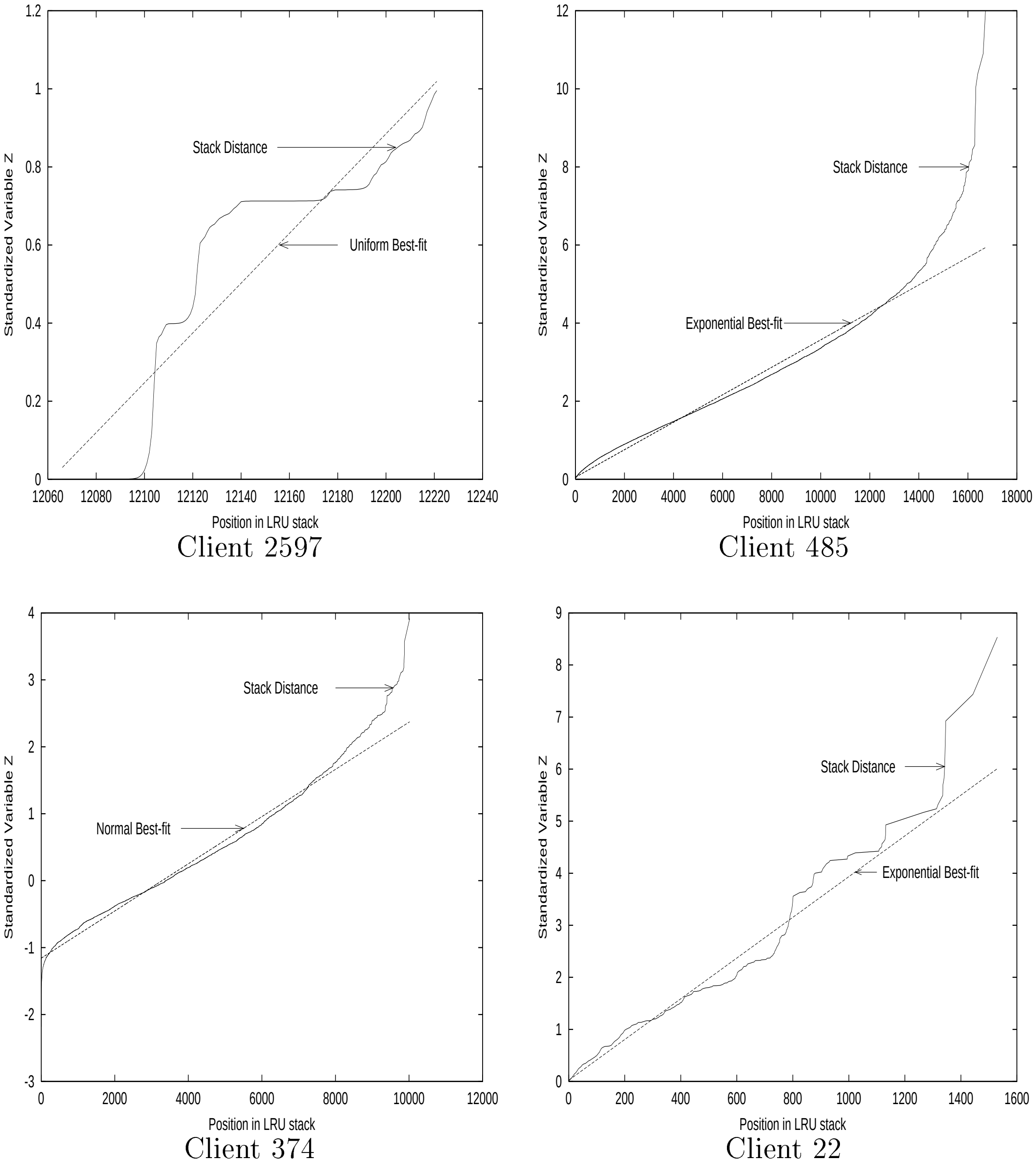


Figure 5.11: Probability Plots for Stack Distance in Client Traces

Table 5.4: Goodness-of-fit Measure (R^2) of Stack Distance Traces

Analytical Model	Client Number			
	2597	485	374	22
Normal	0.75	0.94	0.98	0.89
Lognormal	0.75	0.57	0.12	0.43
Uniform	0.77	0.74	0.99	0.74
Exponential	0.57	0.96	0.81	0.88

measure shows that Clients 485 and 22 are most closely modeled by the Lognormal model, while Client 374 closely fits the Normal model and Client 2597 remains modeled by the Uniform model. The reason for the difference between the A^2 and R^2 goodness-of-fit measures is that the A^2 technique is good for capturing the upper tail of the empirical distribution. The results show that Clients 485 and 22 exhibit strong temporal locality, Client 374 has weak temporal locality, while Client 2597 has very poor temporal locality. The results are also consistent with the visual inspection of temporal locality as shown in Figure 5.9.

5.2.6 Inter-Arrival Times

The distribution of inter-arrival time, time intervals between successive requests from single clients, is characterized. It is hypothesized that the higher the probability of smaller (larger) inter-arrival times, the stronger (weaker) the frequency of requests at the server. Since persistence due to frequency of requests without the corresponding smaller inter-arrival times may worsen the utilization of HTTP persistent connections, this analysis provides evidence to refute or support its use for "heavy-access" clients at the server.

Figure 5.12 shows the cumulative frequency distribution of inter-arrival times for the four selected clients. The first observation from the plot is that Client 2597 generates almost every additional request in its request stream at time intervals of less than one second. That is, more than 95% of the inter-arrival times are less than one second. For the other clients (Clients 22, 374 and 485), over 80% of the inter-

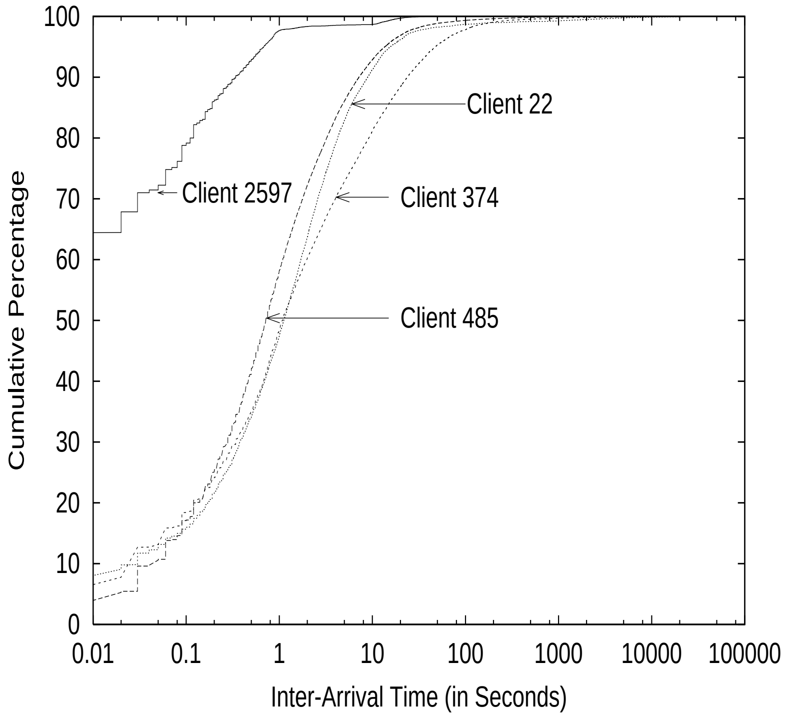


Figure 5.12: Cumulative Frequency Distribution of Inter-Arrival Times for the Four Selected Clients

arrival times are less than 10 seconds, while almost 100% of the entire inter-arrival times are less than 100 seconds. The results imply that a threshold inter-arrival time between 10-100 seconds would provide an effective timeout value for HTTP persistent connections for all the selected clients. A similar report was made in [8].

A few inter-arrival times have large inter-arrival times at the upper tail of the distributions. These large inter-arrival times are believed to be due to the period within which the client is not active in generating requests. Such periods represent the large difference in time between the last request before the client becomes non-active and the first request when the client becomes active again. These observations give an indication that HTTP persistent connections could be beneficial to all the selected clients. However, the inter-arrival time distribution of Client 2597 is quite different from the other clients in the sense that almost all its requests are being generated in less than one second. The implication is that Client 2597 is likely to retrieve most of the documents requested at reduced response times with persistent connections at the server. It is speculated that only non-human clients could have generated those requests within a constant interval of time that is less than one second. Therefore, these observations are consistent with the hypothesis that Client 2597 is a non-human client, while Clients 22, 374, 485 are all human clients.

5.2.7 Performance Implications of Web Client Characteristics

The characteristics of Web clients at the server show that caching strategies are also suitable for managing their request streams. Since few clients generate most requests at the server, a client-based management strategy that manages the request stream of clients is possible. However, the revelation of two major types of clients known as human and non-human clients as part of these few clients requires policies that detect such clients and at the same time adjust to their referencing behaviour. The use of single resource management policies based on aggregate traffic characteristics of clients is incapable of maximizing performance at the server. The reason is because these two types of clients exhibit a dramatic difference in their behaviour characteristics.

While the request stream from non-human clients is difficult to cache, the request streams of human clients can be cached if they are properly managed based on their reference patterns. This implies that the use of client information is essential in view of adjusting to the referencing behaviour of individual clients for effective management of the server resources. This approach will make possible an allocation of server resources in such a way that most human clients can experience better response times at the server at the expense of non-human clients. Client information and cache management policies that are amenable to the characteristics of human clients are considered as the key to effective server resource management.

5.2.8 Summary

Various characteristics of the reference patterns of heavy-access clients have been illustrated. Two major groups of heavy-access clients are identified at the server: human and non-human. The non-human client (Client 2597) is found to exhibit *crawling* behaviour, while the human clients (Clients 22, 374, 485) exhibit mixtures of *working*, *surfing* and *transition* behaviours. Therefore, referencing behaviour of individual "heavy-access" clients are quite unlike the aggregate referencing behaviour discussed in Chapter 4. The results of the workload characterization of clients show that there is a dramatic difference in the behavioural characteristics of heavy-access clients. In particular, non-human client traffic differs from human client traffic.

5.3 Characterizing Sessions within Web Client Traffic

The third level of the Web server access hierarchy relates to the traffic emanating from sessions within individual clients, as determined by a threshold time interval. The characterization of client sessions is meant to achieve two objectives: (i) to examine the effectiveness of HTTP persistent connections at the server; and (ii) to examine phase-transition behaviour in client reference patterns. First, the reference patterns of sessions within individual client request streams are subjected to empirical analysis using different threshold time intervals to understand the utilization of persistent connections at the server. Second, a study of phase-transition behaviour in the document reference pattern of sessions within client traffic is conducted. Phase-transition behaviour, if present, would imply the use of adaptive (or prefetching) policies that can detect the transient period in client reference pattern.

5.3.1 Definition of a Client Session

For the purpose of analyzing sessions within client traffic, a client session is defined as the stream of requests from a single client with an inter-reference time less than a given threshold window time. The threshold window time is also referred to as the *timeout value*. This definition is quite different from the session definition given in Section 4.4.3. This means that if request r_{i+1} from a given client arrives at the server Δt seconds after request r_i from the same client, and $\Delta t \leq T$ (where T is the timeout value in seconds), then requests r_i and r_{i+1} are both considered to be part of the same client session. If $\Delta t > T$, then request r_i is the last request in the client session and indicates the end of a client session, while request r_{i+1} initiates the next client session. This definition has been used in [2] [8] [11] to study Web users and clients within the proxy workload.

5.3.2 Analysis of Sessions Within Client Traffic

The impact of a range of fixed timeout values on the distribution of the number of sessions within individual client request streams is investigated. The results obtained from this empirical analysis would assist in understanding the utilization of HTTP persistent connections at the server. Figure 5.13 shows the distribution of the number of sessions within individual client request streams during the period of observation. A range of timeout values between 1 second and 3600 seconds is used. As the timeout value increases, the number of sessions per client drops rapidly. For a one-second timeout value, about 10% of the clients have a single session, while approximately 50% of the clients have at least 10 sessions. With a timeout value of 100 seconds, only 42% of the clients have a single session, while about 80% of the clients have a single session with timeout values of 1800 and 3600 seconds.

The observations from Figure 5.13 show that increasing the timeout value of a session beyond 100 seconds has little impact on the distribution of the number of sessions per client. The significant increase in the percentage of clients having a single session as the timeout value increases is because most clients generate few requests, and a few clients generate most requests at the server. Figure 5.1 also supports this observation. The high variation in the distribution of the number of sessions per client at relatively short timeout values would suggest that persistent

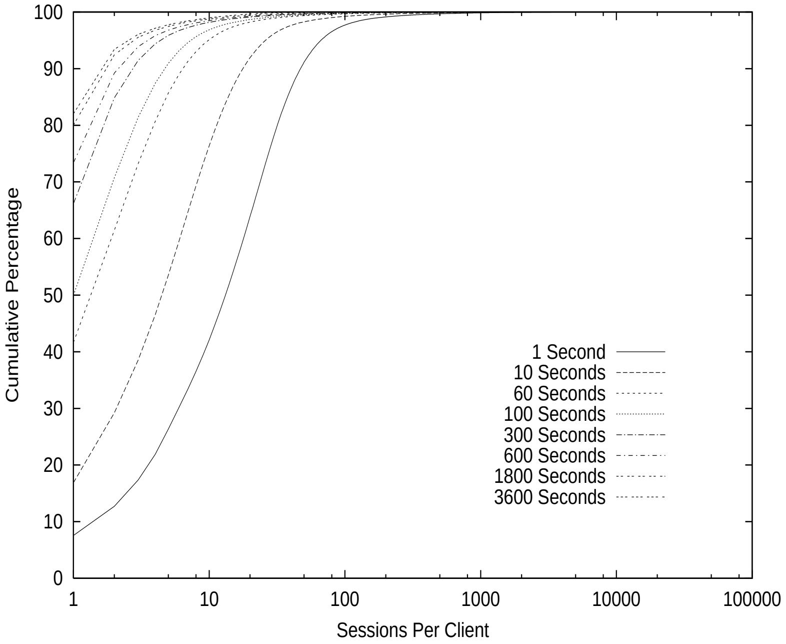


Figure 5.13: Distribution of the Number of Sessions per Client for a Range of Timeout Values

connections may be ineffective at short fixed-timeout values at the server. However, further empirical studies of the variation of document requests, and volume of data transferred within sessions of selected client traffic, will be used to verify this observation.

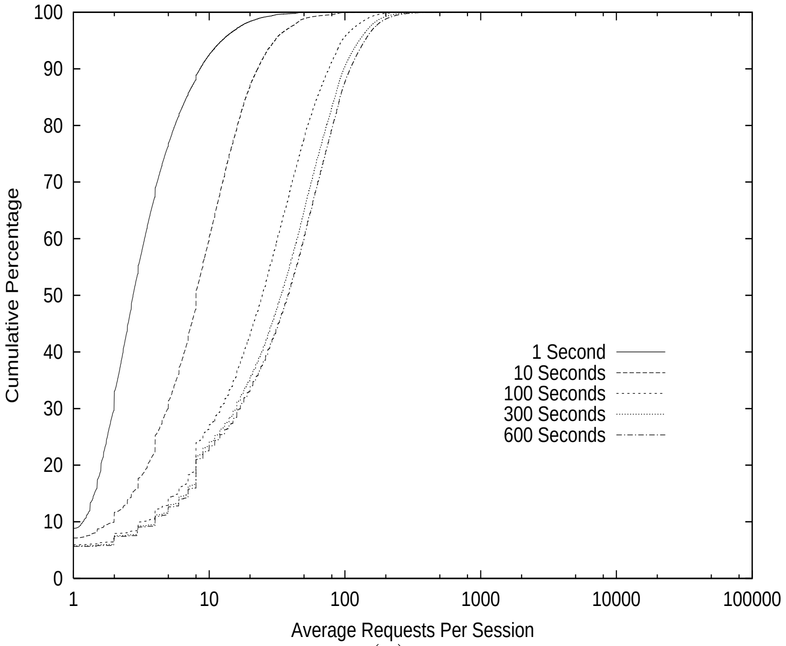
5.3.3 Analysis of Requests and Data Transferred within Client Sessions

The analysis examines the effectiveness of HTTP persistent connections using a range of fixed timeout values for all clients that accessed the server over the period of observation. Since there can be significant variations in total requests and volume of data transferred within sessions of individual clients, the coefficient of variation for total requests and volume of data transferred over sequences of client sessions are used to complement average requests and bytes transferred per client session. Figure 5.14(a) shows the distribution of average requests over sequences of sessions within client request streams. Fewer than 10% of the clients generated an average of one request per client session for all the timeout values considered. For timeout values between 1-100 seconds, the percentage of clients that belong to one-request group remained at 6% for timeout values between 1-100 seconds. This is likely due to the presence of one-timer clients as shown in Table 4.1. The implication of this result is that persistent connections are likely to be beneficial at the server.

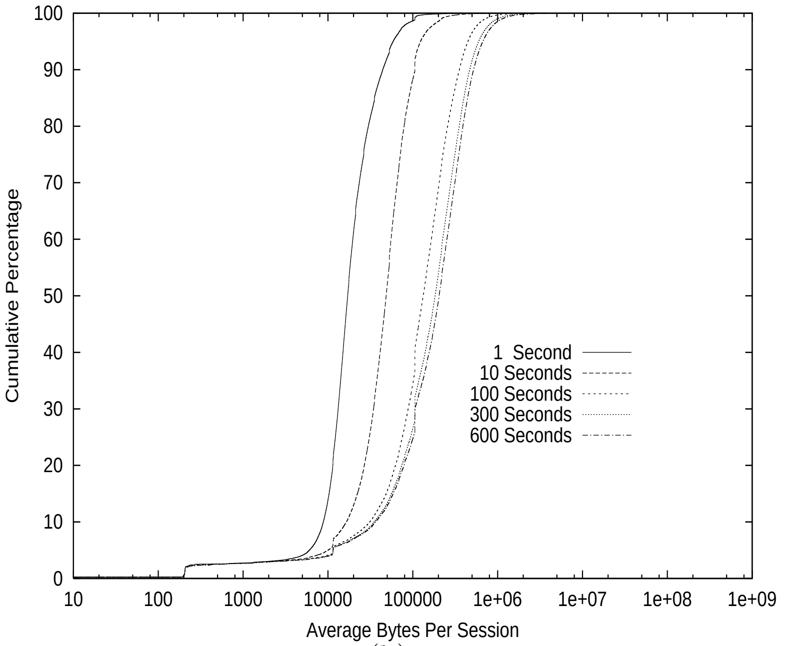
With timeout values between 1-100 seconds, the distribution of average requests per client session differs from one another. As the timeout value increases beyond 100 seconds, the distribution remains unchange. Similarly, Figure 5.14 (b) shows the distribution of average bytes per client session changes for timeout values between 1-100 seconds. For example, about 50% of all sessions are responsible for transferring between 100-1000 kB of data when the timeout value is 100 seconds. When the timeout value exceeds 100 seconds, there is little impact on the distribution of bytes transferred per client session. This implies that a timeout value of 100 seconds or more is closer to the best utilization of HTTP persistent connections[7].

However, the skewness of the total requests and volume of byte transferred per client session can lead to a spurious conclusion on the effectiveness of persistent connections, if an average measure is used. Therefore, the coefficient of variation is further used to characterize individual client sessions. Figure 15(a)-(b) shows the cumulative distribution of coefficients of variation per client session. The figure indicates that almost 100% of the coefficients of variation are less than 1.0 for total requests and data transferred over sequences of individual client sessions. For example, when a timeout value between 100-600 seconds is tested, most clients have a coefficient of variation that is less than 0.1. This small value of the coefficient of variation for total requests and bytes transferred per client session do not vary widely when a timeout value begins from 100 seconds and above. The changes in the distribution of coefficients of variation is likely due to a few clients which generate

[7]This implies that the number of TCP connections required for client-server communications on the Web has been reduced substantially by allowing a few active clients to maintain state of open TCP connections at the server.

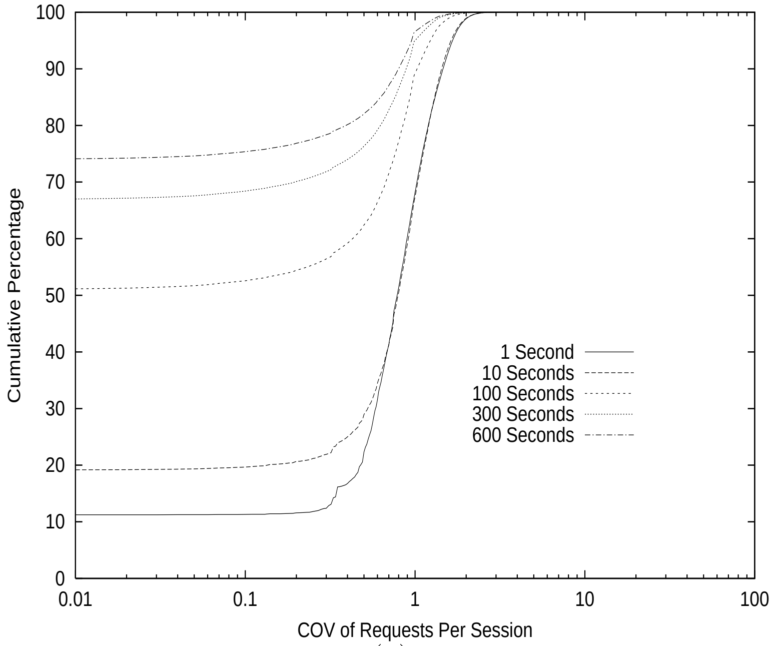


(a)

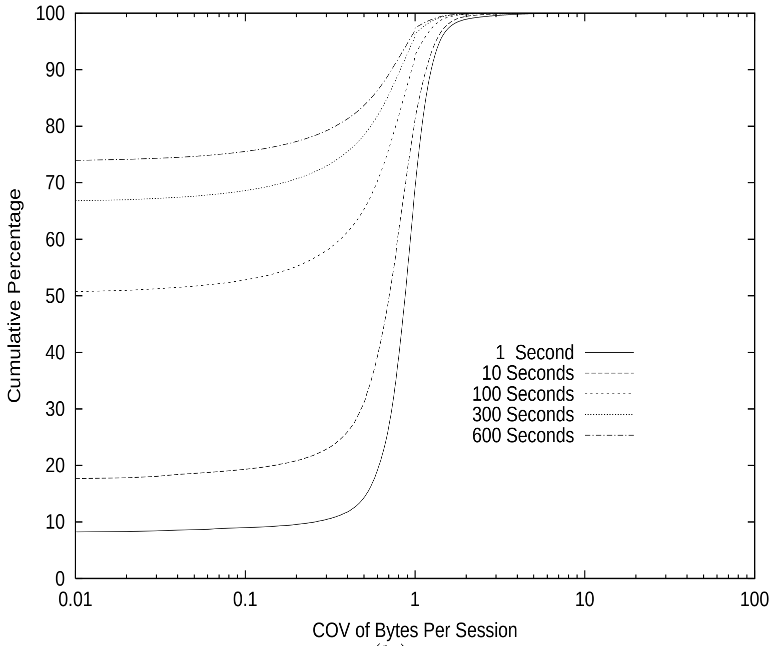


(b)

Figure 5.14: Analysis of Client Session Activity: (a) Average Requests per Session; (b) Average Bytes Transferred per Session



(a)



(b)

Figure 5.15: Distribution of Coefficients of variation: (a) Total Requests per Client Session; (b) Bytes Transferred per Client Session

most of the requests at the server.

For a timeout value between 1-10 seconds, there is no impact on the upper tail distribution of the coefficients of variation for requests per client session while there is little impact in the case of the distribution of bytes transferred per client session. At the lower tail, a significant impact on the distribution of the coefficients of variation for both requests and bytes transferred per client session are observed. However, Figure 5.15 shows that more than 70% of clients have coefficient of variation that is close to 1.0 when timeout values between 1-10 seconds are used. This implies that the distributions of total requests and bytes transferred per client session can be modeled by an exponential distribution. It also means that the total requests and bytes transferred per client session vary widely at short timeout values. This means that HTTP persistent connections are grossly underutilized at short fixed timeout values.

The finding of this empirical analysis supports the opinion expressed in [8] [63] [65] that fixed timeout values may be beneficial for improving the performance of HTTP persistent connections at the server, but at higher timeout values of at least 100 seconds. However, the presence of many clients with few requests (as a result of the upstream caching effect on repeated document references of Web users) might be responsible for the failure of a short fixed timeout value to achieve the best utilization of HTTP persistent connections at the server.

5.3.4 Analysis of Sessions Within Selected Client Traffic

The focus of this analysis is to understand how supposedly non-human and human clients might utilize persistent connections at the server. The request streams of four selected clients as characterized in Section 5.2 are used. A range of timeout values from 1 to 600 seconds is used. Figures 5.16 and 5.17 show the results of different timeout values for the four selected clients discussed in Section 5.2.

Client 2597 shows the potential for better utilization of persistent connections because individual distributions of total requests and volume of data transferred

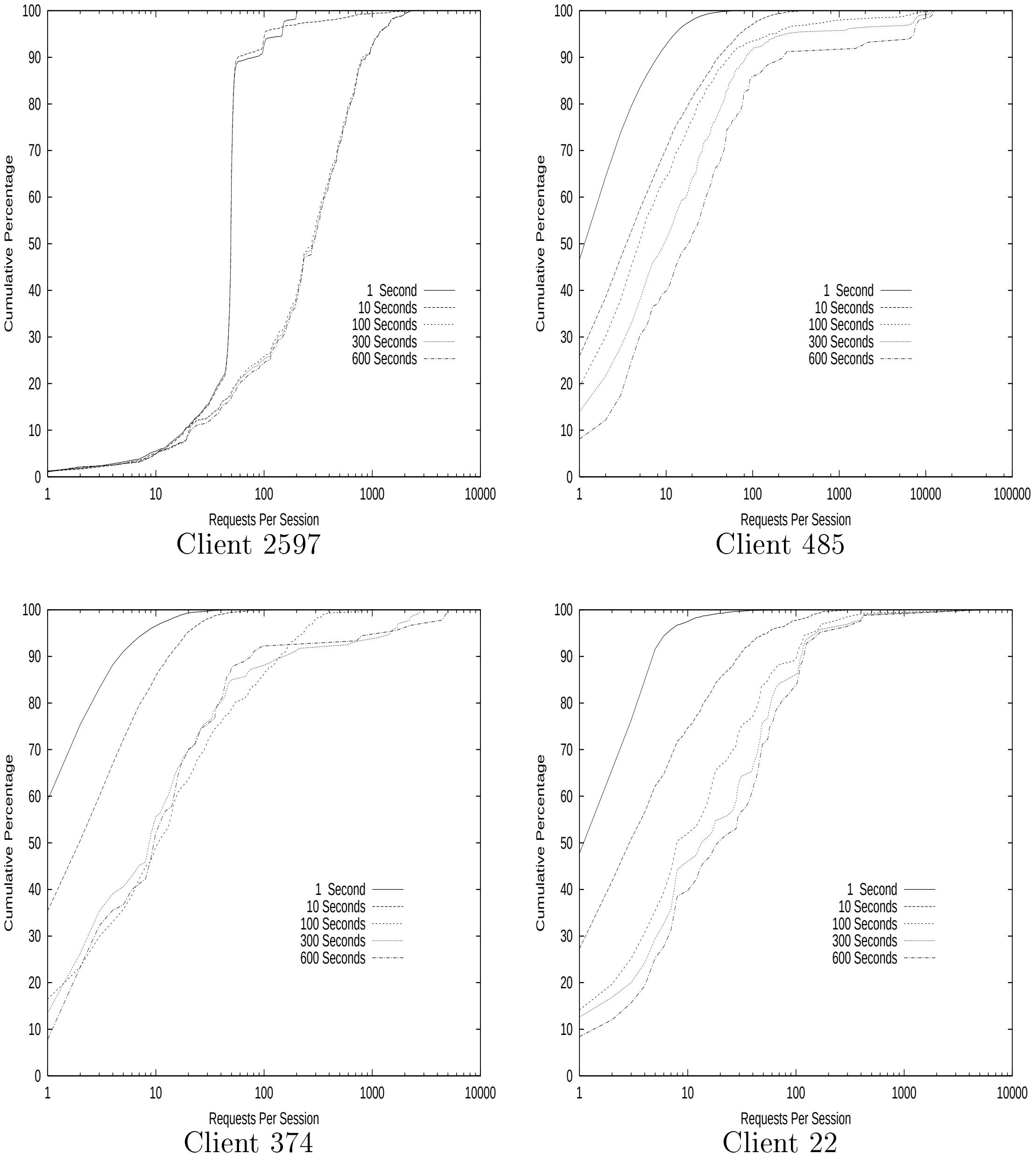


Figure 5.16: Distribution of Requests Within Sessions of Client Traffic

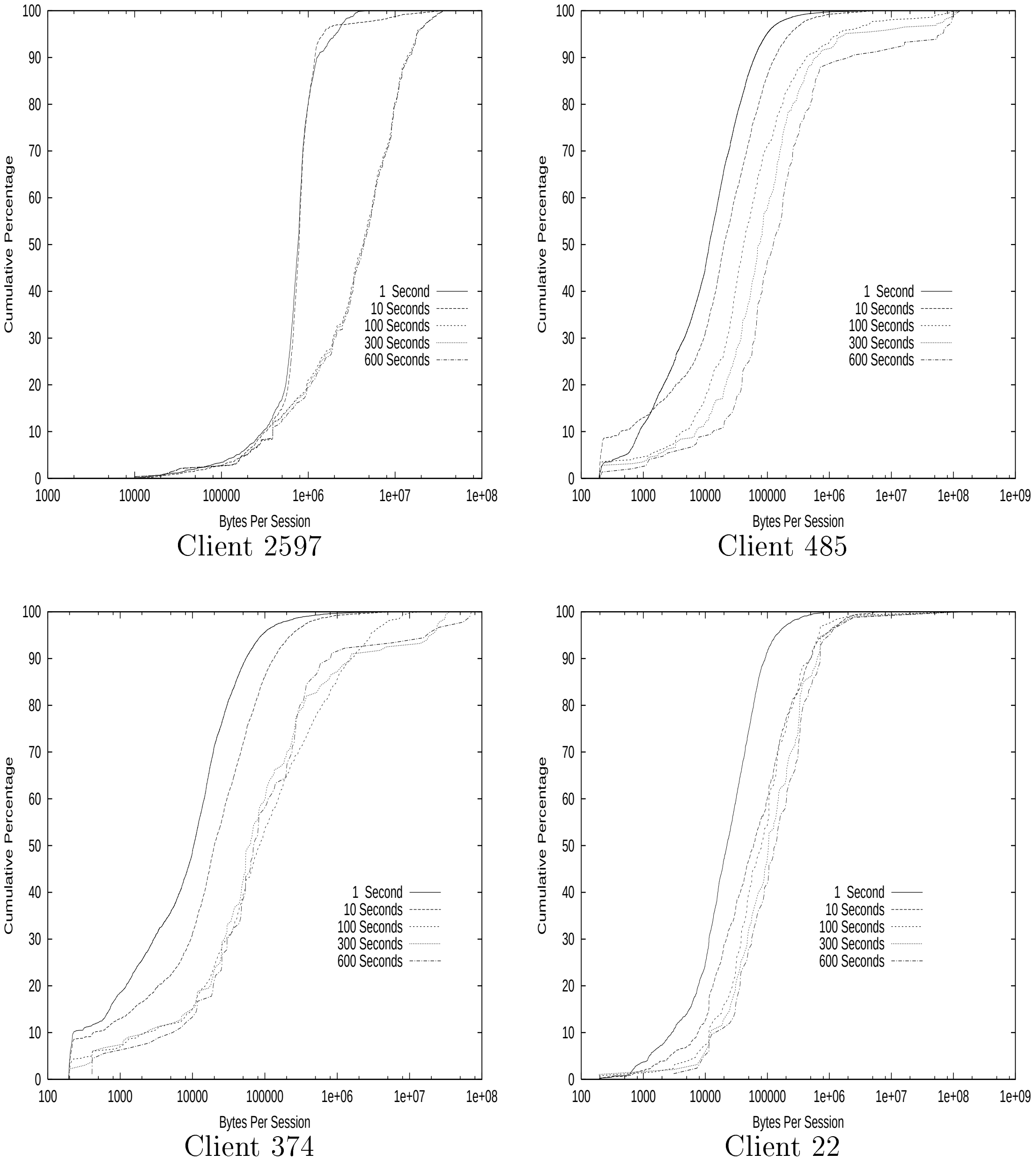


Figure 5.17: Distribution of Bytes Transferred Within Sessions of Client Traffic

across client sessions are the same for small timeout values between 1-10 seconds. For a timeout value of 100 seconds or more, the distribution of total requests and bytes transferred per client session is quite different at the upper tail when compared to the distribution with short timeout values between 1-10 seconds. But at the lower tail, there is no difference in the distributions. For instance, for the timeout values considered, over 95% of the client sessions have at least 10 requests which are responsible for over 100 kB of data transferred from the server. This means that a short fixed timeout value between 1-10 seconds yields the best utilization of persistent connections for Client 2597.

Client 485 shows that for all the timeout values considered, the distribution of total requests and bytes transferred per session is quite different. This implies that none of the timeout values is closer to the best utilization of persistent connections. With a timeout of 100 seconds, about 40% of the client sessions have at least 10 requests which are responsible for over 10 kB of data transferred from the server. In the case of Clients 374 and 22, increasing the timeout values beyond 100 seconds has little effect on the distribution of requests and bytes transferred per session. For these clients, a range of timeout values from 100 to 600 seconds has at least 10 requests which are responsible for over 10 kB of data transferred in most sessions. This implies that a timeout value of 100 seconds is close to the best utilization of persistent connections for Clients 485, 374 and 22.

The results indicate that a supposedly non-human client (such as Client 2597) would utilize HTTP persistent connections effectively at short timeout values, while supposedly human clients (such as Clients 22, 374 and 485) would do likewise at high timeout values. This means that a dynamic timeout policy that treats human and non-client differently is required for optimizing HTTP persistent connections at the server. Since a timeout value of 100 seconds has been observed to yield the best utilization of persistent connections for all clients at the server, it is likely that the dominance of human clients over non-human clients is responsible for this timeout value.

5.3.5 Phase-Transition Behaviour in Client Reference Patterns

An important consideration for this analysis is the study of phase-transition behaviour in document referencing patterns of clients. Each of the four selected heavy-access clients (i.e., Clients 2597, 485, 374 and 22) is characterized to investigate this property. Fundamental to this study is the document working set, which is defined as the number of unique documents accessed within a client session.

Phase-transition behaviour means a significant change in behavioural pattern occurs between consecutive document working sets of sessions within the request streams of clients. For example, phase-transition behaviour would imply that most past document working sets of a client session may not be relevant to the document working set of the session at hand. Therefore, a large change in document working set implies that the entire working set can be discarded. Prefetching may provide a better resource management strategy at this level of the access hierarchy.

In order to characterize the change in document working set over sequences of client sessions, the range of timeout values that provides the best utilization of persistent connections for selected clients is used. Using these timeout values, the percentage changes (in the number of unique documents referenced) between consecutive document working sets over sequences of sessions within a client request stream are first recorded. The complementary cumulative frequency distribution of the recorded percentage changes is plotted to visualize the phase-transition behaviour. Kienzle *et al.* [56] used a similar approach. Since the document working set of the current client session can be larger than that of the previous session, it is expected that the percentage change in document working set between consecutive sessions can be greater than 100%.

Figure 5.18 shows the variation of the document working sets over sequences of sessions within the request stream of individual selected clients. The x-axis of this figure represents the percentage change in the document working set of two consecutive client sessions. The y-axis represents the percentage of consecutive client

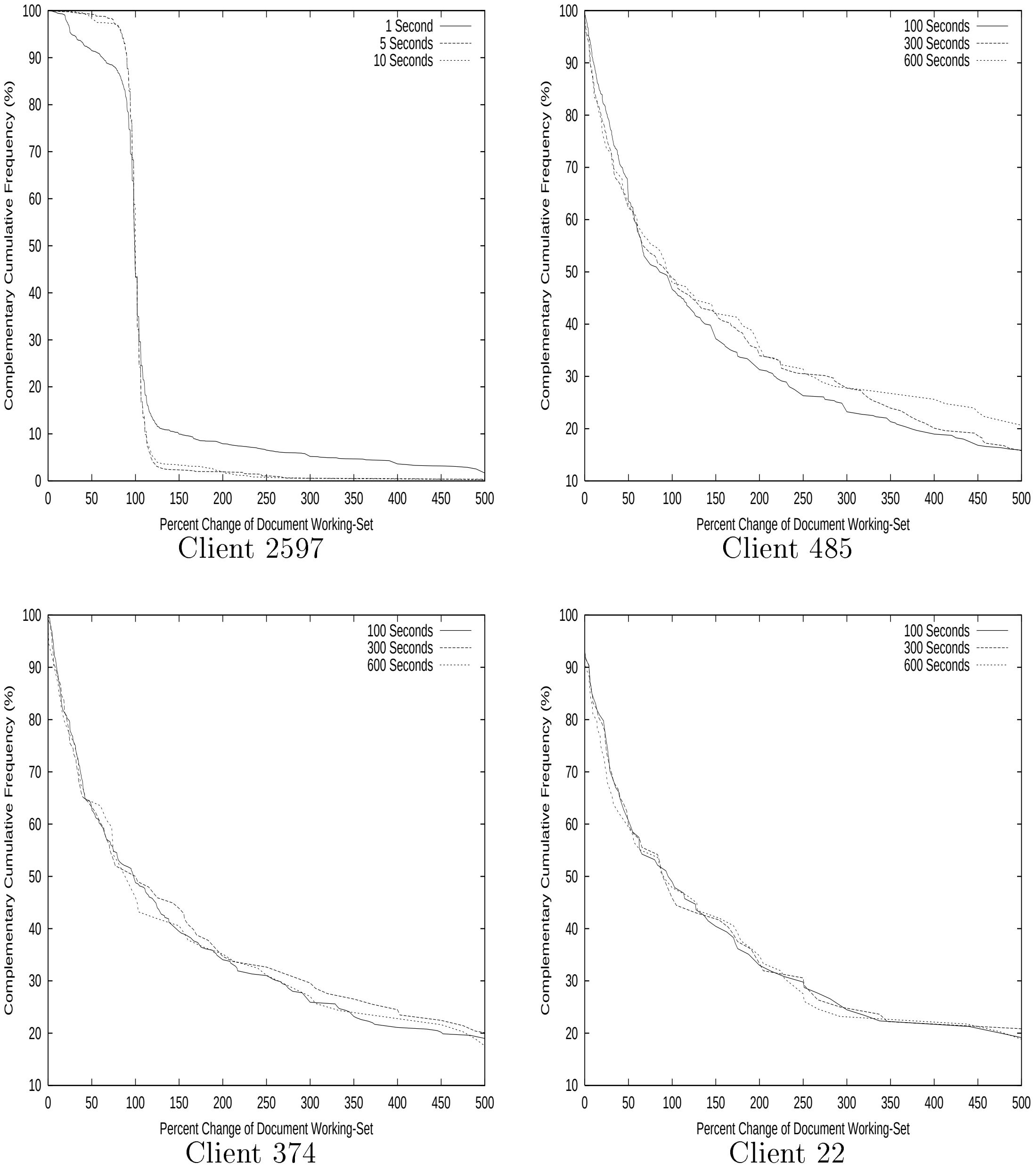


Figure 5.18: Distributions of Percentage Changes Between Consecutive Document Working Sets of Client Sessions

sessions that exceeds the percentage change mark on the x-axis. If most consecutive client sessions (as represented on the y-axis) have a percentage change mark (on the x-axis) greater than 100%, phase-transition behaviour is considered to be present. For this behaviour to be clearly visualized, the tails of the distributions having a few plotted points starting from a percentage change mark greater than 500% on the x-axis are cut off as shown in Figure 5.18. This will allow the percentage of consecutive client sessions that is less than 100% to be visually inspected.

For Client 2597, short timeout values between 1-10 seconds are used. The results show that over 80% of consecutive client sessions have a percentage change that is greater than 100%. This means that the document working set of the previous session is completely different from that of the current session. However, when the 600-second timeout value is used, the document working sets over sequences of sessions are the same for Client 2597. This implies that it takes about 600 seconds for the non-human client to generate all its requests each time it visits the server and any timeout value less than 600 seconds would result in large differences between consecutive client sessions. Phase-transition behaviour is present in the document reference pattern of the client when a timeout value less than or equal to 300 seconds is used.

The variation of document working sets over sequences of sessions is characterized for Clients 22, 374 and 485 using a timeout value between 100 and 600 seconds. This range of timeout values has been observed to be closer to the best utilization of persistent connections for this type of client. Clients 22, 374 and 485 seem to have reference patterns that are more predictable to some extent because individual change distributions drop off sharply below the 100%-change mark on the x-axis for all the timeout values. Client 485 has about 50% of consecutive sessions within its request stream exceeding the 75%-change mark on the x-axis. Similarly, Clients 22 and 374 have about 50% of consecutive sessions within their respective request streams exceeding the 100%-change mark on the x-axis. After these percentage change marks on the x-axis for Clients 485, 374 and 22, a slowly declining behaviour is observed for each of these clients. This behaviour implies that the distributions have longer tails, showing considerable difference in document working sets between consecutive client sessions larger than 100%. These large changes mean longer transitions in document reference pattern of client sessions, which suggests a shift in client document referencing behaviour at the server.

The results show that the variation of the document working sets over sequences of sessions for Client 2597 (non-human) is quite different from other selected clients (human clients). A complete change of consecutive document working sets across sessions of non-human client traffic is observed when the timeout value is less than the threshold time it takes the client to frequently generate requests each time it visits the server. However, in the case of human clients (Clients 22, 374 and 485), phase-transition behaviour is observed in their request streams, but the document working sets between consecutive client sessions are predictable to some extent because of the sharp drop in their distributions.

5.3.6 Performance Implications of Web Client Sessions

Several characteristics of Web client sessions have been observed. In this section, an attempt is made to describe how these characteristics can be used to make resource management at the server effective. Since HTTP persistent connections require that the server must reserve resources, then performance implications are first discussed in the light of this.

Persistent connections exist in HTTP/1.1 [17]. The feature allows active clients to maintain state of open TCP connections at the server within a fixed timeout value. The request stream (from which the total requests made and volume of bytes transferred by a client are estimated) before closing the TCP connections contains the activity of the client within a session. In view of this, the activity of clients during persistent connections can easily affect performance since the server must reserve memory resources to implement this feature.

For instance, if the memory is the bottleneck, there is limitation to the number of active clients that can maintain state of open TCP connections at the server depending on the timeout policy and duration. If the timeout policy is fixed for a client session and the timeout duration is high, the server maintains the state of open TCP connections for a longer time and this can make the allocation of memory resources to newly active clients impossible at times. This situation can impair the performance of the server. On the other hand, if a fixed timeout policy is used and the duration of timeout duration is short, the scarce memory resource can be used effectively if characteristics of sessions within client traffic indicate that most open TCP connections at the server are not idle. In the case of this server workload, such short and fixed timeout policies have been shown to underutilize HTTP persistent connections. This affects the server performance negatively because there is no substantial reduction in the number of TCP connections for short timeout values that are fixed. But high and fixed timeout policies are also appropriate when only the server memory is bottleneck free. Another implication of the memory being the bottleneck is that the nature of sessions within the request streams of human clients will affect the server negatively while that of non-human clients may not.

However, if the client information is transparent to the server, a fixed timeout policy can be used for non-human clients, while an adaptive timeout policy as suggested in [8] [11] [65] can be used for human clients. This is because non-human clients optimize persistent connections at short timeout values between 1-10 seconds, while human clients do likewise at a high timeout value of 100 seconds or more. Since the pattern of requests and bytes transeferred over sequences of sessions within human clients are observed to be exponentially distributed[8] at short timeout values, the use of adaptive and short timeout policies that are amenable to this type of client pattern should be evaluated. Such policies will cause the server memory that is the bottleneck to be managed effectively for HTTP persistent connections.

The implication of phase-transition behaviour in client referencing patterns is that an adaptive strategy based on a prefetching or phase-transition model is sug-

[8]The choice of exponential model is because over 70% of clients that accessed the server have coefficients of variation for requests per client session and bytes transferred per client session that are very close to 1.0 when short timeout values between 1-10 seconds are used. Figure 5.15 reveals this observation.

gested for the management of server resources from the perspective of individual client reference patterns.

5.3.7 Summary

The third level of the Web server access hierarchy has been characterized since the management of server resources and the effectiveness of HTTP persistent connections may depend on the ability to recognize the characteristics of sessions within individual clients. The results show that a fixed timeout value of 100 seconds can offer the best utilization of HTTP persistent connections at the server. In particular, human clients are observed to make good use of persistent connections at a high timeout value of 100 seconds while the non-human clients do so at short timeout values between 1-10 seconds. The behaviour of the patterns of service demand of the non-human client within sessions is quite different from that of human clients. Phase-transition behaviour is present in client reference patterns, but easy to predict to some extent in human clients and difficult to predict in the non-human client.

5.4 Summary

In this chapter, the workload of individual client request streams was characterized. The results show that there are both human and non-human reference clients. These two types of clients have different behavioural characteristics which make a single resource management policy incapable of maximizing the performance of the server. The characterization of sessions within selected client request streams also indicates that human clients need a high timeout value to optimize persistent connections and the non-human clients require short timeout values. This implies that a fixed timeout policy is better for optimizing non-human clients while an adaptive timeout policy is suggested for human clients. The need for client information at the server is seen as a key factor for using this approach. Phase-transition behaviour is also observed as a dominant document referencing pattern in request streams of selected clients.

Chapter 6

Trace-Driven Simulation for Web Server Performance Study

This chapter discusses the results of the performance evaluation of different caching strategies, using some of the salient characteristics of the Web server access hierarchy reported in Chapters 4 and 5. The performance study uses trace-driven simulation to assess how the various approaches to cache management perform at the server. Section 6.1 articulates the objective of the simulation experiments. Section 6.2 describes the design of the simulation experiments. Section 6.3 reports the results of the simulation experiments. Section 6.4 summarizes the chapter.

6.1 Experimental Objectives

The primary objective of the simulation experiments is to demonstrate how the characteristics of the Web server access hierarchy can be used to manage Web server resources effectively. Cache management policies that are capable of making informed decisions based on the document referencing behaviour of Web clients at each level of the server access hierarchy are essential to achieve this objective. In view of this, an assessment of the performance of cache replacement policies that are frequency-based and recency-based are investigated using a trace-driven simulation approach. The secondary objective of the simulation is to evaluate the performance of cache management approaches based on the use of client information.

6.2 Design of the Simulation Experiments

Many aspects of the simulation are explained in this section, including a description of the various factors considered and parameters required for the experiments. Section 6.2.1 gives a description of the system simulated. Section 6.2.2 presents the metrics used for measuring the performance of cache replacement policies with various cache strategies. Section 6.2.3 discusses the factors and parameters considered for the simulation experiments.

6.2.1 Description of the System

Trace-driven simulation has been widely used to evaluate the performance of Web servers [7] [10] [25] [26] [29] [58]. For this performance study, the focus is the document cache located at the server. The simulated system operates by receiving time-ordered document requests from clients. If the document is found in the server's cache, a *document cache hit* has occurred and the server returns a copy of the document to the client immediately. Otherwise, a *document cache miss* has occurred, and the server retrieves a copy of the document from its backing store (where all documents on the server are kept permanently). Upon returning the requested document to the client, a copy of the document is kept in the cache so that repeated references to the same document would be serviced from the cache rather than from the backing store. In the event that the server's cache is full, new cache space is provided by replacing some cached document(s) with the new incoming document according to the replacement policy in effect.

Requests for documents of zero-size are exempted from the performance study. Zero-document sizes are attributable to proxy clients, which sometimes verify if documents found on their backing store are the same as copies at the origin server. If these document copies have been modified at the origin server, then the proxy clients would download these modified documents from the origin server, and update the document copies on their backing store, before forwarding the requested documents to their respective end-users. Otherwise, a zero-document size transfer occurs at the origin server. This approach is part of the solution to the *cache consistency* problem at the upstream cache. Document requests which are aborted by clients can also have zero-document sizes at the server. Since a zero-document-size request is not being serviced from the server disk system, such requests are not included in the simulation experiments.

6.2.2 Performance Metrics

There are three commonly used performance metrics to evaluate cache replacement policies in the Web: document hit ratio, byte hit ratio and response time. The document hit ratio (DHR) is defined as the ratio of the number of document requests satisfied by the document cache to the total number of document requests received by the server. Byte hit ratio (BHR) (or weighted hit ratio [86]) means the ratio of the volume of data (in bytes) that corresponds to the number of requests satisfied by the document cache, to the total volume of data (in bytes) that corresponds to the total number of document requests received by server. The higher the document hit ratio and byte hit ratio, the better the performance of the cache replacement policies. A third possible metric is response time, which measures the time required for the server to transfer a copy of the document requested by a client, before serving the next document request on its queue. Since there is no information relevant to the usage of this metric in the server workload, it is impossible to use response time as a measure of performance. Implicitly, it is assumed that a higher document hit ratio implies a shorter average response time for servicing a document at the server because the need for backing store access is reduced.

6.2.3 Factors and Parameters

In order to carry out a performance study of the Web server, certain factors are considered. These factors are the size of the document cache at the server, the cache replacement policy, the threshold of document sizes for admission to the cache and the configuration of the server cache. The simulation parameters include the length of the "warm-up" period, the set of client requests and the estimated size of the unique documents available at the server.

6.2.3.1 Cache Sizes

The first factor for the simulation experiments is the size of the document cache at the server. Cache sizes range from 1 MB to 4096 MB, increasing in a geometric progression with a common ratio of 2 at every interval. Table 4.1 shows that the total size of unique documents in the server workload is 2,819 MB, which implies that using a cache size of 4096 MB can be regarded as an infinite-sized cache. Testing the performance of cache replacement policies within this range of cache sizes shows the relationship between cache replacement policy and cache size in terms of document hit ratio and byte hit ratio.

6.2.3.2 Cache Replacement Policies

The second factor for the simulation experiments is the cache replacement policy. A good cache replacement policy should be able to make use of properties of the request stream to evict documents that are not of immediate use from the cache, so that documents more likely to be accessed in the near future are retained in the cache. In addition, a good policy should be computationally efficient and perform a minimal amount of book-keeping. Five cache replacement policies are evaluated in this study: Least Recently Used (LRU), Least Frequently Used (LFU), Least Frequently Used with Aging (LFU-Aging), Frequency-based Largest Document Removed First (SIZE-F) and Recency-based Largest Document Removed First (SIZE-R).

The decision to evaluate the performance of these particular replacement policies for managing the server cache is based largely upon the observed behavioural characteristics at the first level of the server access hierarchy namely: (i) the frequency of small documents compared to large documents, (ii) the persistence of small documents compared to large documents, (iii) the strong temporal locality; and (iv) the high concentration of references. Both *frequency-based* and *recency-based* policies are considered. LRU is one of the most popular replacement policies. This policy exploits the principle of temporal locality by evicting document(s) least recently accessed from the cache. The assumption of the policy is that such document(s) would not be accessed by clients in the near future. LRU is simple to implement for small caches, but can become computationally expensive for large caches. Because the aggregate reference pattern exhibits temporal locality, LRU is expected to yield good cache performance.

LFU uses the frequency of document accesses to preserve documents that are likely to be referenced in the cache. This policy operates by allowing the cache to maintain a frequency-of-access count for all its documents. When replacement is required, LFU selects the document with the smallest frequency of access count. Unfortunately, once popular documents with high frequency counts, which are no longer accessed by clients, could remain entrenched in the cache, thereby preventing newly popular documents from acquiring sufficient reference counts to stay in the cache. This phenomenon has been described as *cache pollution* [54], and negatively affects cache performance.

To address this problem, LFU-Aging attempts to gradually reduce the reference counts of cached documents which are no longer requested by clients, thus making them potential candidates for replacement. The maximum reference count that any cache document can attain (M_{refs}) and the mean reference count per cache document (A_{max}) are the two required parameters to implement LFU-Aging. Cached documents that reach M_{refs} will no longer have their reference counts increased, and whenever the mean reference count per cached document reaches A_{max}, the reference count of each document in the cache is reduced by a factor of 2. In this study, the first parameter is not used (i.e. $M_{refs} = \infty$), while the second parameter (A_{max}) is assigned different values between 1 and 200. The choice of A_{max} values are based on results obtained in [7]. The SIZE policy uses the document size to predict which documents are likely to be requested in the near future. When replacement is required, SIZE chooses the document with the largest size. Two approaches are used to resolve ties: SIZE-R uses LRU and SIZE-F uses LFU-Aging.

6.2.3.3 Cache Configuration

The third factor for the simulation experiments is the cache configuration. Two possible configurations are considered: global caching and partitioned caching. With global caching, there is a single large cache at the server, and all requests compete for it. Partitioned caching on the other hand, deals with the division of a single large cache into partitions so that request streams reaching the server can be separately managed. In computer architecture for instance, partitioned caching has been used to increase bandwidth between the memory hierarchy and the CPU. At the Web server, Murta *et al.* [67] studied the performance impact of dividing the cache into three partitions based on document sizes. In contrast, Arlitt [7] studied the effect of partitioning on the basis of document types on Web server performance. The rationale for partitioned caching is to improve the hit ratio in each partition by preserving document reference locality. This means that documents stored in one partition cannot be replaced by documents in other partitions.

Various placement schemes for documents are possible in partitioned caches. One scheme is to place documents in partitions based on empirical observations of document referencing behaviour. In this particular workload, the empirical observations in Chapter 4 show that documents smaller than 1 kB receive the highest number of document requests (46.65%), documents within 10-40 kB contribute the highest volume of data transferred (49.60%), and most documents are between 1-40 kB. Using these empirical observations as a means of allocating request streams reaching the server, partitioning on document size can be studied. This scheme is referred to as empirical placement policy. Murta *et al.* suggested placing documents in partitioned caches based on median and mean of document sizes. This scheme is referred to as the statistical placement policy. In this study, both statistical measures and empirical observations of document referencing behaviour are used to determine the document size threshold(s) for partitioning the server cache.

Another scheme is per-client placement policy, in which documents are placed in partitions based on a round-robin allocation of individual client Internet Protocol

(IP) addresses. This implies that documents which are requested by individual clients during the period of observation are placed in partitions being allocated to these clients the first time they visit the server. For this type of placement scheme, a client information table would be kept in the server's memory for directing subsequent individual client's request streams to allocated partitions for service. However, how this client information table would be managed is not considered in this research. The per-client placement scheme is useful for understanding the potential value of client information for resource management at the server.

For this performance study, partitioned caching will be studied first by partitioning a single large cache at the server into two equally-sized caches and managing each of the partitioned caches with policies for placement and replacement of documents. The second part will partition a single large cache at the server into three equally-sized caches, each of which is managed with policies for placement and replacement of documents. The use of empirical, statistical and per-client placement policies are all tested for partitioned caches, while two frequency-based policies (LFU-Aging and SIZE-F) and two recency-based policies (LRU and SIZE-R) are used as replacement policies. These experiments are meant to explore the possible performance gains from partitioned caches.

6.2.3.4 Threshold Sizes

The fourth factor for the simulation experiments is the threshold that is used to constrain the size of documents allowed in the cache. The choice of the document size threshold is based on the workload characterization in Chapter 4. For example, documents less than 100 kB were found to be responsible for most requests at the server, most data transferred over the network and most unique documents available at the server. Of these documents, those with sizes less than 0.5 kB were found to be frequently requested. Figure 4.6 justifies the threshold sizes by showing that caching documents that are larger than 100 kB at the server would result in a negligible increase in performance. Therefore, a lower threshold of 0.5 kB and an upper threshold of 100 kB are used to assess the performance of threshold-based policies. Threshold sizes are applied only to documents entering a single large cache at the server.

6.2.3.5 Workload Parameters

The workload parameters for the experiments are the sizes of the documents at the server and the access log which contains the request stream from clients. The workload used is described in detail in Chapter 4. Because the transfer size for individual documents in the access log varies, the estimated size of each unique document is set to the maximum transfer size observed for that document in the access log.

6.2.3.6 Warm-Up Period

An important issue in any simulation is the length of the warm-up period within which no performance measurements are taken. This issue is considered to avoid the impact of first-timer document misses on the cache performance. In order to find the length of the warm-up period, a preliminary experiment was conducted, using an infinite-sized cache so that document replacement does not occur. Therefore, all misses in the cache are due to first-timer document requests. The workload was divided into intervals of 10,000 requests. At the end of each interval, document hit ratio and byte hit ratio were collected. The performance values taken for each interval were plotted and visually inspected to determine the point at which there is no further change in the mean document hit ratio. This point is known as the *cache steady-state*, after which performance measurements can be taken. Arlitt [7] applied this approach to Web server workloads in order to determine the length of the warm-up period. Figure 6.1 shows the plot that was used to determine the length of the warm-up period for this study, in terms of number of client requests. The vertical line indicates 10 million requests as the length of the warm-up period. This represents about one-fifth of the total number of requests in the workload.

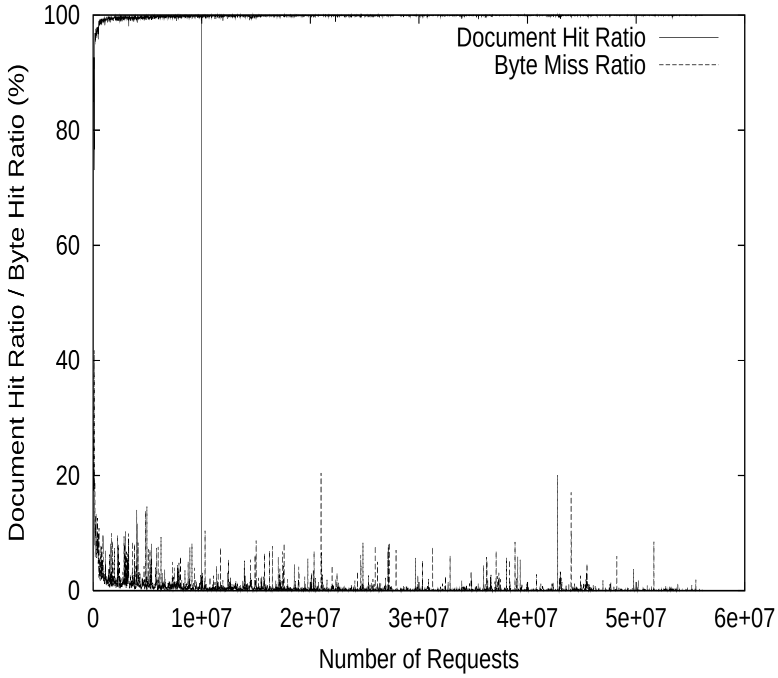


Figure 6.1: Identification of Cache Steady-State

6.3 Evaluation of Cache Replacement Policies

The evaluations of the performance of cache replacement policies at the server are presented in three different ways. First, global caching on the aggregate traffic is considered. Second, partitioning is considered on the basis of document size and on a per-client basis. Since information about clients is not available in the access log collected, a round-robin distribution of individual client's request streams to partitioned caches based on client IP addresses is used for partitioning on a per-client basis. Third, management on the basis of client type is considered in an attempt to assess how the behavioural characteristics of individual clients (human and non-human) affect performance under different replacement policies.

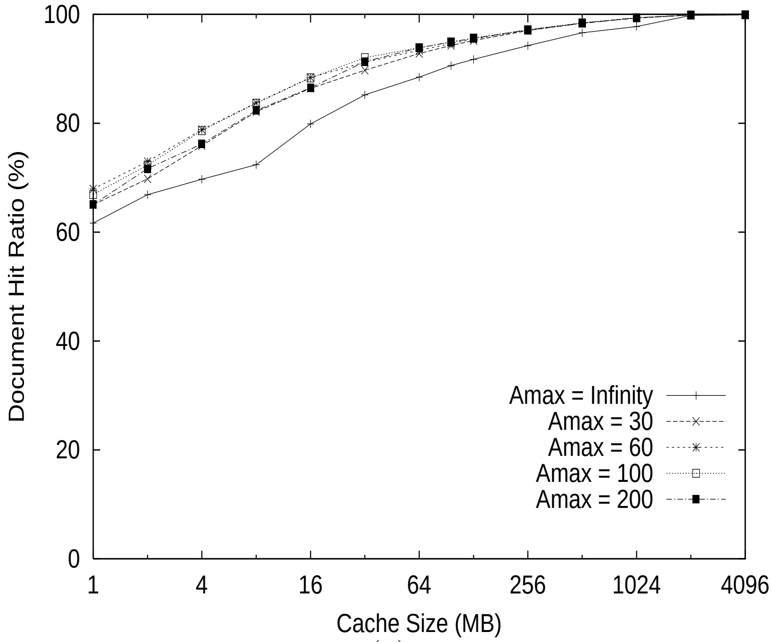
6.3.1 Global Caching

This section addresses the performance of cache management policies at the Web server using the aggregate request stream from multiple clients and a single large cache. The performance of cache replacement policies at the server is assessed. In addition, the effect on performance of admitting documents on the basis of threshold sizes is evaluated.

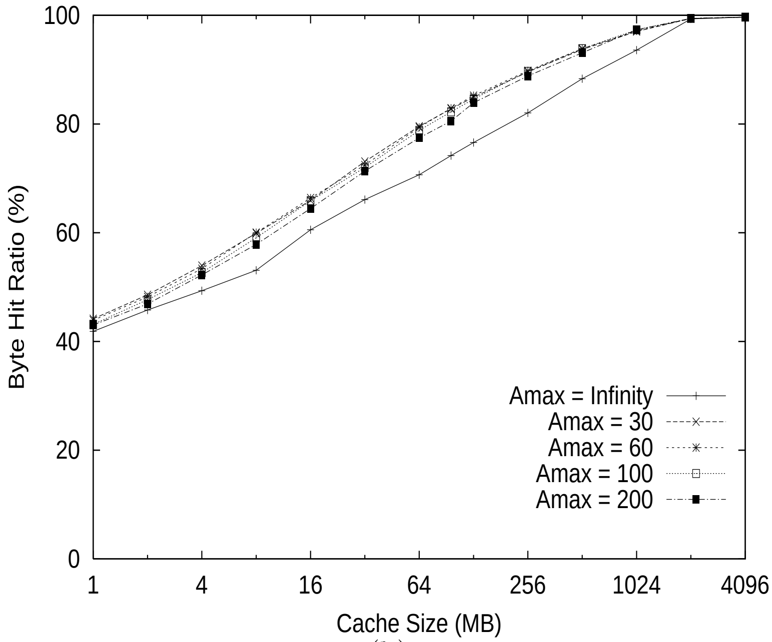
6.3.1.1 Unrestricted Caching

The performance of the four replacement policies (LFU-Aging, LRU, SIZE-F, SIZE-R) for managing a single large cache at the server is investigated. Since LFU-Aging and SIZE-F require an aging policy to avoid cache pollution, the best aging parameters for these two policies are determined as suggested in [7] [61]. Recall from Section 6.2.3.3 that the frequency-based aging policy uses two parameters: M_{refs} and A_{max}. By setting $M_{refs} = \infty$, the frequency-based aging policy (LFU-Aging) can be reduced to LFU when $A_{max} = \infty$. The reduction of A_{max} parameter to various fixed values is tested to study the performance improvement of LFU-Aging over what LFU would provides. To choose the best A_{max} value for LFU-Aging, the performance of LFU-Aging is evaluated for $M_{refs} = \infty$, when A_{max} values of 30, 60, 100 and 200 are tested. The A_{max} value that yields the best performance for LFU-Aging will be used for all subsequent experiments to evaluate the performance of frequency-based policies in comparison to recency-based policies. Figure 6.2 shows the performance of LFU-Aging for the different values of A_{max} considered, when $M_{refs} = \infty$. The figure reveals that LFU-Aging performs better than LFU (with no aging policy) and that A_{max} has a significant impact on LFU-Aging performance when $M_{refs} = \infty$. This result is different from what was observed in [7]. There is no clear difference in the performance of LFU-Aging when the A_{max} value is either 60 or 100, but for all subsequent experiments LFU-Aging and SIZE-F use aging parameters that are set to $A_{max} = 60$ and $\mathrm{M}_{refs} = \infty$.

The performance of the various cache replacement policies with a single large cache at the server is summarized in Figure 6.3. The results indicate that there are substantial differences in performance of cache replacement policies when the server cache is small. However, when the server cache is large, the performance differences of the cache replacement policies become insignificant. At small cache sizes (below

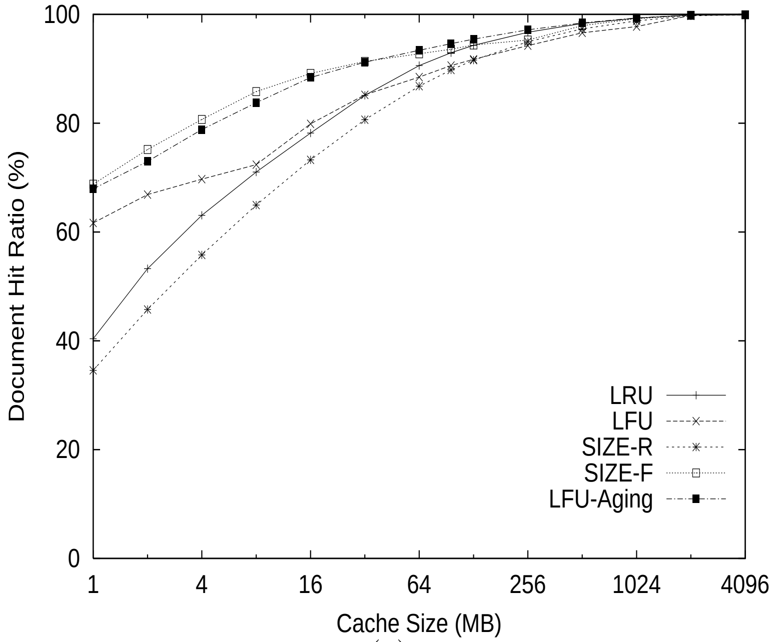


(a)

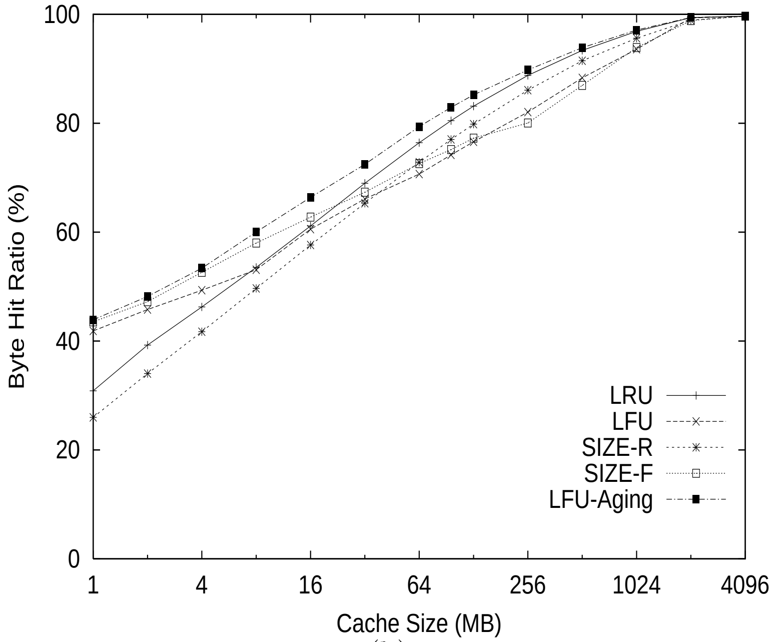


(b)

Figure 6.2: Effects of the A_{max} Parameter on the Performance of the LFU-Aging Policy: (a) Document Hit Ratio; (b) Byte Hit Ratio



(a)



(b)

Figure 6.3: Effect of Cache Replacement Policies on Performance (Global Cache): (a) Document Hit Ratio; (b) Byte Hit Ratio

256 MB), all the frequency-based replacement policies perform better than all the recency-based replacement policies considered in this study. In particular, SIZE-F achieves the best document hit ratio, while LFU-Aging provides the best byte hit ratio. The performance of SIZE-F, however, becomes worse when the cache size is greater than 64 MB, while LFU-Aging provides the best document hit ratio and byte hit ratio. This implies that SIZE-F is only better than LFU-Aging at small cache sizes. The reason for this can be attributed to the peculiar nature of the document reference pattern in this workload, in which about 5% of the unique documents (which were mostly small documents) account for about 93% of the total document requests received. Since a size-based policy is known to preserve small documents by evicting large documents, such a policy maximizes document hit ratio while sacrificing byte hit ratio.

The relative advantage of LFU-Aging over SIZE-F as the cache size becomes larger is caused by the inability of SIZE-F to evict small documents which are no longer frequently requested by clients. In the case of LFU-Aging, small documents can easily become potential candidates for replacement as opposed to large documents that were frequently requested because the size of documents is ignored. Similarly, LFU-Aging has better performance than SIZE-F in terms of byte hit ratio because the policy does not discriminate against large documents. If the mean document hit ratio and mean byte hit ratio are calculated for every size of the server cache considered in this experiment, LFU-Aging performs best. However, for relative small cache size (i.e. 32 MB), SIZE-F is regarded as the best in terms of mean document hit ratio and byte hit ratio.

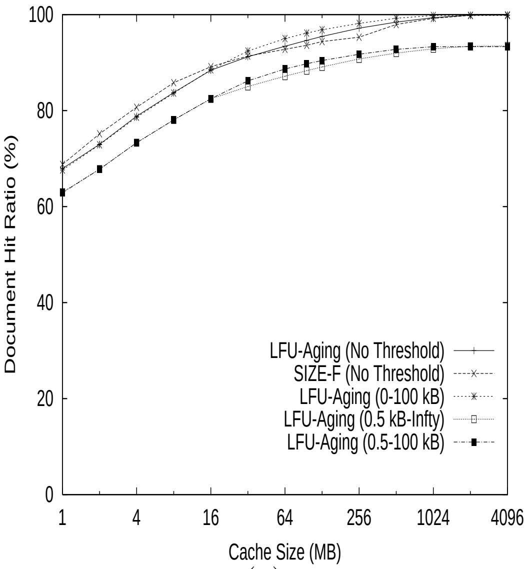
6.3.1.2 Caching with Document Size Thresholds

The performance impact of document size thresholds on cache replacement policies for managing a large global cache at the server is studied. The study is important because document size thresholds could not only improve the performance of cache replacement policies, but could also require the cache replacement policies to perform a minimal amount of book-keeping. In this section, four different types of threshold-based policy are considered: (i) no restriction on the size of documents admitted to the cache; (ii) a restriction on the maximum allowable document size; (ii) a restriction on the minimum allowable document size; and (iv) a restriction on both the minimum and maximum allowable document size.

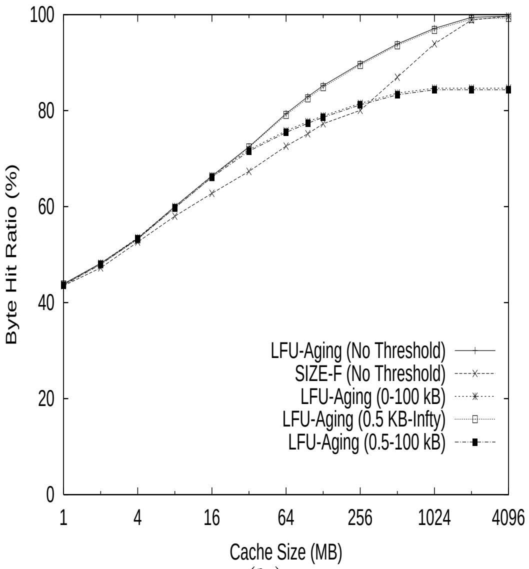
The results of the performance study in Section 6.3.1.1 are considered to be the results of a threshold-based policy with no restriction on the size of documents admitted to the global cache. These performance results are meant to provide a basis for comparison with other threshold-based policies. The three threshold-based policies are combined with individual cache replacement policies to determine if there is any performance improvement at the server. For these experiments, a document size of 100 kB is chosen as the upper limit threshold, while a document size of 0.5 kB is chosen as the lower limit threshold. The introduction of these particular document size thresholds is based on the observation (in Chapter 4) that documents less than 100 kB account for 97% of documents accessed by clients in the workload.

Additionally, this group of documents is responsible for most of the document requests and volume of data transferred in this workload. Documents less than 0.5 kB in size account for only 4% of unique documents accessed by clients and 7% of total document requests at the server. Arlitt [7] selected this lower limit threshold size for assessing the effect of threshold-based policies at the server. Thus, the focus of these experiments is to assess the performance of the cache replacement policies when some frequently requested documents are prevented from entering the large global cache at the server and when some large documents that contribute to high volumes of data transferred are prevented from entering the large global cache at the server. The three experiments are conducted for threshold-based policies, which include: (i) setting the maximum limit of document sizes at 100 kB (labeled as 0-100 kB); (ii) setting the minimum limit of document sizes at 0.5 kB (labeled 0.5 kB-Infty); and (iii) setting the minimum and maximum limits of document sizes at 0.5 kB and 100 kB (labeled 0.5-100 kB).

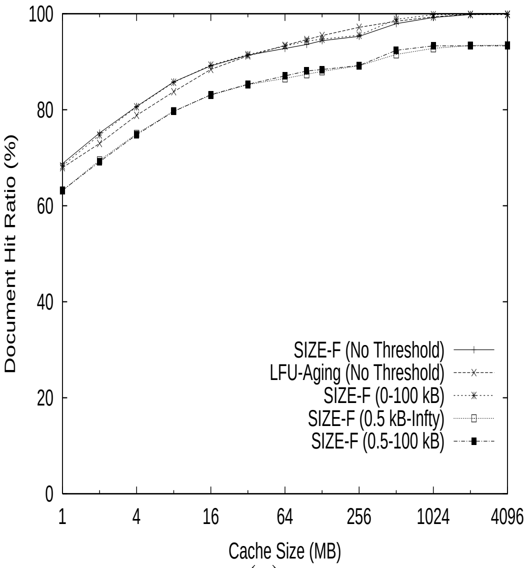
Figure 6.4 displays the results of the frequency-based policies (SIZE-F and LFU-Aging) with thresholds, while Figure 6.5 displays the results of the recency-based



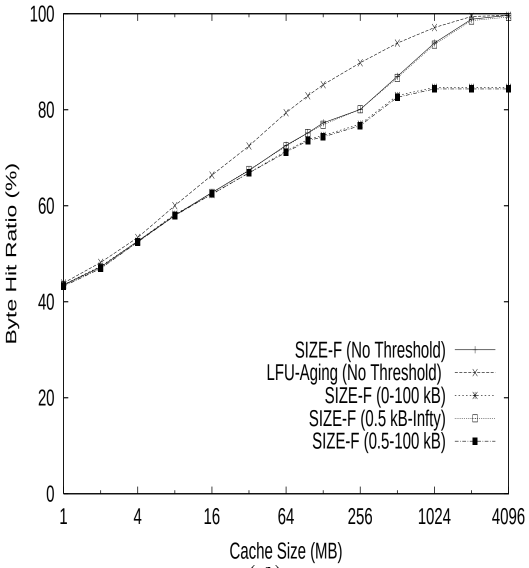
(a)



(b)



(c)



(d)

Figure 6.4: Effect of Frequency-Based Cache Replacement Policies with Document Size Thresholds on the Performance of a Web Server (Global Cache): (a)-(c) Document Hit Ratio; (b)-(d) Byte Hit Ratio

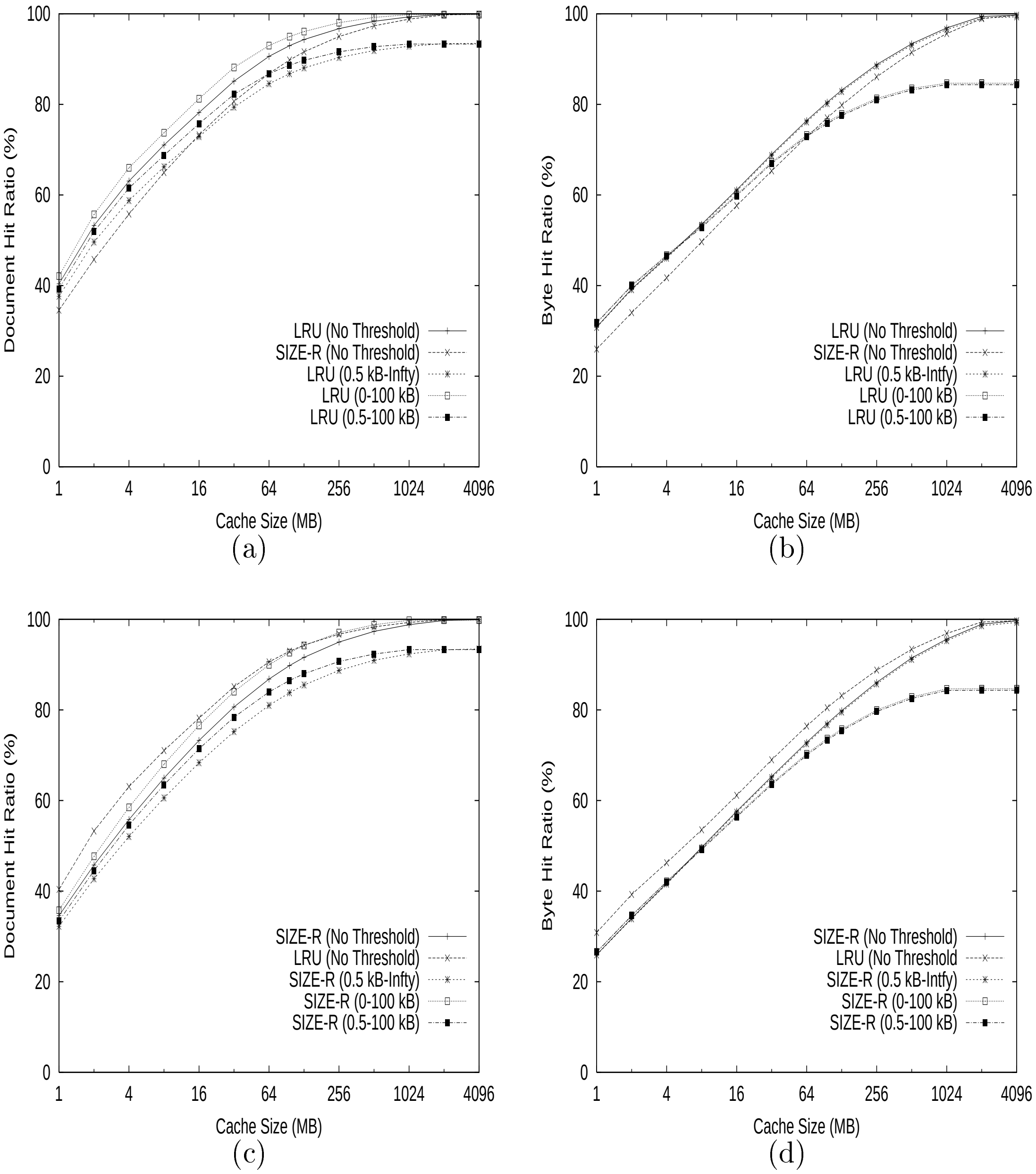


Figure 6.5: Effect of Recency-Based Cache Replacement Policies with Threshold Sizes on the Performance of a Web Server (Global Cache): (a)-(c) Document Hit Ratio; (b)-(d) Byte Hit Ratio

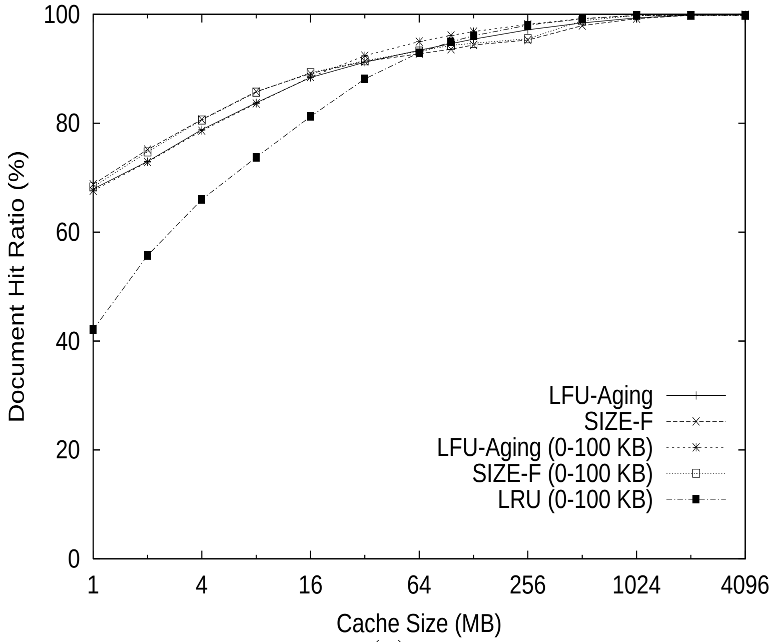
policies (SIZE-R and LRU) with thresholds. In both Figure 6.4 and Figure 6.5, the performance results of global caching for replacement policies without thresholds are presented for comparison. This means that there is provision for judging whether or not threshold-based policies are beneficial to global caching. The performance of LFU-Aging with thresholds is shown in Figure 6.4(a)-(b). Using the document hit ratio, SIZE-F without thresholds is the best when the cache size is less than 64 KB, and LFU-Aging with an upper limit threshold of 100 kB performs best when the cache size is greater than 64 MB. For byte hit ratio, LFU-Aging without thresholds and LFU-aging with a lower limit threshold of 0.5 kB are the best. Similarly, the performance of SIZE-F with thresholds is shown in Figures 6.4(c)-(d). The results indicate that SIZE-F with an upper limit threshold of 100 kB and SIZE-F without thresholds have the best document hit ratios. For byte hit ratio, LFU-Aging without thresholds performs best. The results obtained for frequency-based replacement policies with thresholds are consistent with the suggestion made earlier in Chapter 4 that if documents which are greater than 100 kB are prevented from entering the server cache, cache replacement policies are likely to perform better. A notable implication of these results is that cache policies with an upper document size threshold could improve performance by increasing document hit ratio, while cache policies with a lower document size threshold could provide a minimal amount of book-keeping even if such policies do not provide any performance improvement. Thus, threshold-based policies can be useful for either reducing the load on the disk system (by setting upper document size thresholds) or conserving the bandwidth of the disk system (by setting lower document size thresholds).

The performance of LRU with thresholds is shown in Figure 6.5(a)-(b). Using the performance results of global caching for recency-based policies (LRU and SIZE-R) without thresholds as the control case, LRU with thresholds significantly improves performance when compared to LRU without thresholds. In particular, LRU with an upper limit threshold of 100 kB has the best document hit ratio, while LRU with a lower limit threshold of 0.5 kB and LRU without thresholds have the best byte hit ratio. In the case of SIZE-R with thresholds as shown in Figure 6.5(c)-

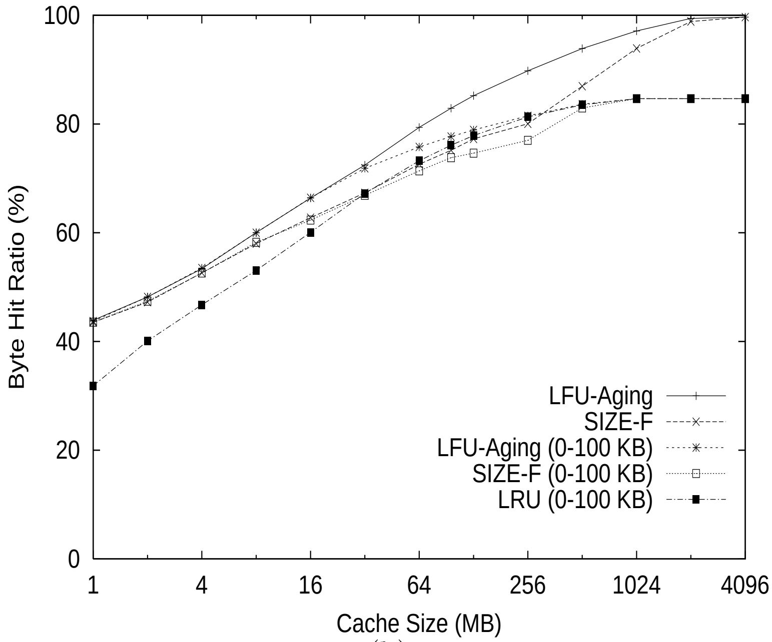
(d), LRU without thresholds is the best for both document hit ratio and byte hit ratio. The results simply indicate that LRU benefits from document size thresholds, while SIZE-F does not. The reason for this is that temporal locality in this workload is a characteristic of documents, regardless of the sizes. Another reason is that SIZE-F protects small documents, even when those documents are no longer requested by clients, and thus limits the likely performance gains of using document size thresholds.

Figure 6.6 summarizes the performance of cache replacement policies with thresholds for global cache at the server. The figure displays the performance results of frequency-based policies (SIZE-F and LFU-Aging) without thresholds, frequency-based policies (SIZE-F and LFU-Aging) with thresholds and recency-based policies with thresholds (LRU) for the purpose of comparison. These selected policies are evaluated only with an upper limit threshold of 100 kB. When the cache sizes are less than 64 MB, SIZE-F and LFU-Aging with or without thresholds perform nearly the same. For LRU with an upper limit threshold of 100 kB, a significant performance improvement is observed from Figure 6.6, although less than what frequency-based policies with or without thresholds can provide. At large cache sizes, greater than 64 MB, all the cache replacement policies shown in Figure 6.6 have a negligible difference in performance. The results clearly indicate that LRU with upper thresholds can improve performance, especially when the server cache is small.

In particular, LFU-Aging without thresholds has the best byte hit ratio, while LRU with an upper limit threshold of 100 kB has the worst byte hit ratio at cache sizes below 64 MB. For large cache sizes above 64 MB, LRU with an upper threshold of 100 kB produces almost the same byte hit ratio as LFU-Aging with an upper limit threshold of 100 kB. This observation implies that LRU with upper thresholds is tailored towards increasing the document hit ratio while sacrificing the byte hit ratio. A common implication of this behaviour is that small documents accessed at the server exhibit stronger temporal locality than large documents. Since cache replacement policies with thresholds yield at least the same level of performance as those policies without thresholds, managing global cache with thresholds can



(a)



(b)

Figure 6.6: Effect of Replacement Policies with Document Size Thresholds on Performance (Global Cache): (a) Document Hit Ratio; (b) Byte Hit Ratio

provide a minimal amount of book-keeping operations at the server.

6.3.2 Partitioned Caching

Partitioned caching is an alternative to global caching. The possible benefits are studied by allocating request streams from clients to separate partitions, based first on the size of Web documents requested by clients and then by the client IP address the first time it appears at the server. In investigating the effect of partitioned caches on server performance, three important decisions are required: (i) the number of partitions at the server; (ii) the size of the partitions; and (iii) the policies for document placement and replacement within the individual partitions.

6.3.2.1 Partitioning on the Basis of Document Size

In the first approach, partitioned caching is studied with the use of document size for allocating client request streams to partitions. Document size thresholds are obtained using statistical measures (such as median document size and mean document size) and empirical observations based on document referencing behaviour in Chapter 4. With the use of these appropriate choices, two simulation experiments are considered in this section.

The first experiment considers with two equally-sized partitions. Each of the four replacement policies (LFU-Aging, SIZE-F, LRU and SIZE-R) is combined with statistical and empirical placement policies based on the size of documents accessed by clients. The statistical placement policy is implemented with two equally-sized partitions by placing documents that are less than or equal to the mean document size (28 kB) in the first partition, while documents larger than 28 kB are placed in the second partition. For the empirical placement policy, documents less than or equal 40 kB are placed in the first partition, while documents larger than 40 kB are placed in the second partition. The reason for choosing a document size threshold of 40 kB is that most documents which were responsible for a high number of references and a volume of data transferred in this workload are within this threshold size as observed in Chapter 4 and explained in Section 6.2.3.3. Recall that the same replacement policy is used to manage each of the partitions in these experiments.

The second experiment investigates the effect of three equally-sized partitions. Combinations of the four replacement policies (LFU-Aging, SIZE-F, LRU and SIZE-R) and statistical and empirical placement policies are tested. The statistical placement policy uses the mean document size (28 kB) and median document size (0.9 kB), while the empirical placement policy uses document size thresholds of 1 kB and 40 kB, respectively. These threshold sizes were based on document referencing behaviour observed in Chapter 4 and explained in Section 6.2.3.3. The statistical placement policy is implemented by placing documents smaller than or equal to the median size (0.9 kB) in the first partition. Documents larger than the median document size (0.9 kB) but less than or equal to the mean document size (28 kB) are placed in the second partition. Documents larger than the mean document size (28 kB) are placed in the third partition. For the empirical placement policy, documents smaller than or equal to 1 kB are placed in the first partition. Documents larger than 1 kB but smaller than or equal to 40 kB are placed in the second partition. Documents larger than 40 kB are placed in the third partition.

Figure 6.7 shows the results from the first experiments using two equally-sized partitions. The figure is labeled such that the performance results of replacement policies with partitioned caches can be clearly identified from replacement policies without partitioned caches. For instance, a label with the prefix "PART" identifies a replacement policy that manages each of the partitions. If a postfix "*" is added to the label, it implies that documents are placed in partitions based on the empirical placement scheme; otherwise, the statistical placement scheme is used. The performance results of the cache replacement policies with global cache represent the control case. Figure 6.7 (a)-(b) shows that frequency-based policies (SIZE-F and LFU-Aging) without partitions are better than frequency-based policies (SIZE-F and LFU-Aging) with two equally-sized partitions. Similarly, Figure 6.7 (c)-(d) shows that recency-based policies without partitions are better than recency-based policies with partitions. This implies that two equally-sized partitions do not im-

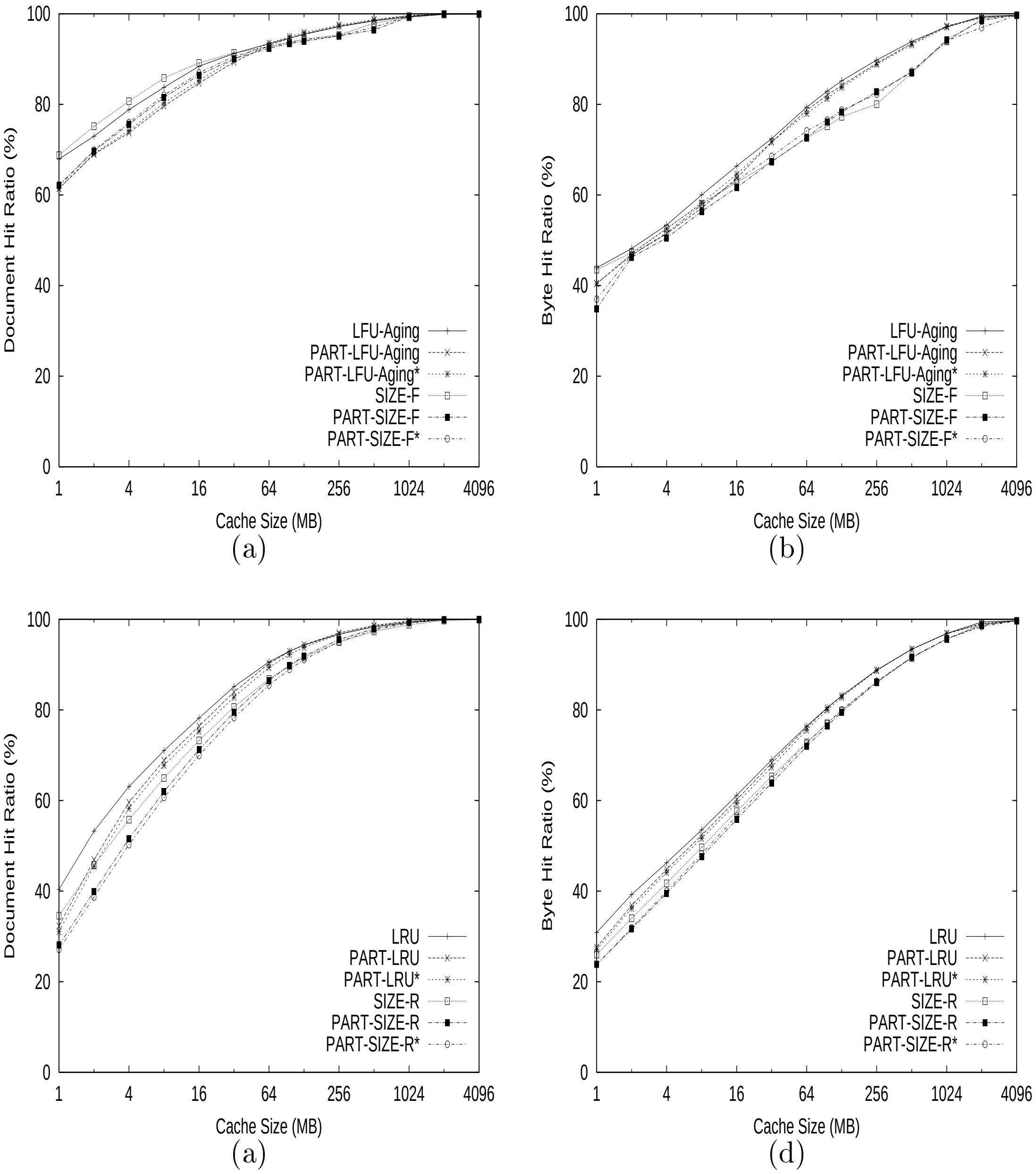


Figure 6.7: Effect of Two Equally-Sized Partitions: (a)-(b) Performance of Frequency-Based Replacement Policies; (c)-(d) Performance of Recency-Based Replacement Policies

prove performance when compared to global caching. The possible reason for this can be attributed to the choice of document size thresholds, in which small documents that are most frequently requested are placed in the first cache partition. This choice leads to an allocation in which small documents are replaced in the first partition in order to provide cache space for incoming documents, while large documents in the second partition are frequently replaced due to the small cache size.

Figure 6.8 shows the results from the second experiment using three equally-sized partitions. The labels on this figure are interpreted in the same manner as in Figure 6.7. The performance results of cache replacement policies with a single global cache are the control case for this experiment. Figure 6.8(a)-(b) shows that SIZE-F with three equally-sized partitions performs better than all the other frequency-based policies with or without partitions. In general, the frequency-based policies with three equally-sized paritions are observed to be better than frequency-based policies with global cache. With partitioned caching at the server, SIZE-F and LFU-Aging are the best in terms of document hit ratio and byte hit ratio, respectively. Recall that these replacement policies are also the best under global caching, both in terms of document hit ratio and byte hit ratio. The results from this experiment also demonstrate that when cache sizes are below 32 MB, the performance of SIZE-F with global caching is not different from that of the frequency-based policies with partitioned caches. Figure 6.8(c)-(d) indicates that there is no performance improvement in the use of the recency-based policies with three equally-sized partitions when compared to the recency-based policies with global caching. This observation implies that partitioned caching would not benefit recency-based policies at the server. The temporal locality in document referencing behaviour that is associated with small documents at the server is speculated to be the reason for these performance results. This means that the potential of recency-based policies can be weakened by an increase in the number of cache partitions because of poor exploitation of the temporal locality inherent in the server workload.

The performance improvement of the frequency-based policies with three equally-

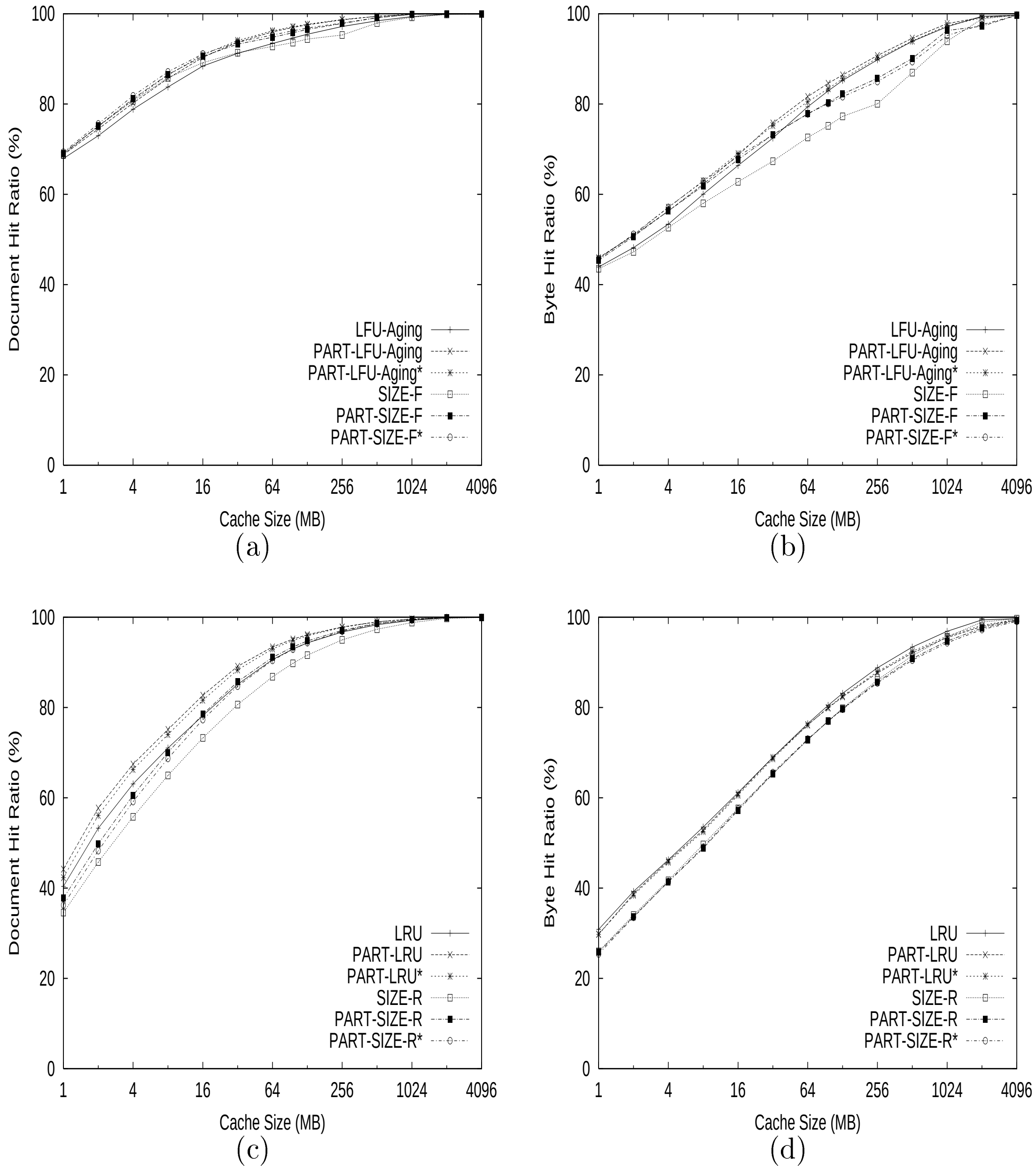


Figure 6.8: Effect of Three Equally-Sized Partitions: (a)-(b) Performance of Frequency-Based Replacement Policies; (c)-(d) Performance of Recency-Based Replacement Policies

sized partitions compared to a global cache is due to the document placement policies used. The results suggest that a partitioned cache configuration can be beneficial at the server if the document size thresholds are properly estimated using either statistical measures or empirical observation of referencing behaviour. Once the document sizes are accurately estimated, it should be possible to allocate documents that are frequently accessed and documents that are responsible for a high volume of data transferred to different partitions. The ability of document size placement policies to identify these types of documents is a key to performance. Moreover, no significant difference in performance is seen between document size placement policies that used statistical measures or empirical observations of document referencing behaviour. In practice, the statistical or empirical estimation of document size(s) for partitioning may be difficult to obtain, but on-line analysis of Web server traffic may be helpful in this regard.

The implication of these results is that frequency-based replacement policies can improve performance if the server cache is adequately partitioned with an appropriate document size placement policy. The results obtained for the cache replacement policies with partitioned caches in this study support the conclusion reached in [86] that partitioned caches at the server can improve performance.

6.3.2.2 Partitioning on a Per-Client Basis

The effect on server performance of partitioning on a per-client basis is studied. The experiments are conducted using various partition size combinations. Documents requested by individual clients are managed in partitions based on a round-robin allocation of the clients' first request at the server. Subsequent document requests from clients are directed to the same partitions where the first requests from these clients are serviced. This requires a client information table that matches individual clients to previously assigned partitions. The additional cost of maintaining such information is not considered in this analysis. Using appropriate choices for setting up partitioned cache experiments as outlined in Section 6.3.2, two experiments are conducted for determining the effectiveness of this approach to managing the server resources.

The first experiment considers two-equally sized partitions. Client requests are allocated to one of the two partitions in a round-robin manner using the client IP address. Once a client is allocated to a partition, all subsequent requests are directed there. This scheme is known as per-client placement policy. Each of the four replacement policies (SIZE-F, LFU-Aging, LRU, and SIZE-R) is used to manage the partitions and Figure 6.9 shows the performance results. The figure is labeled such that the performance results of the replacement policies that manage partitioned caches based on a per-client placement policy are clearly identified from replacement policies that manage global cache. For instance, a label with the prefix "PART" identifies a replacement policy that manages each of the cache partitions with a per-client placement policy. With reference to document hit ratio, both SIZE-F with two-equally-sized partitions and per-client placement policy, and SIZE-F with global cache are observed to be the best. SIZE-F with partitions and per-client placement policy performs slightly better than SIZE-F with global cache when the cache size is between 16 MB and 128 MB. However, in terms of byte hit ratio, LFU-Aging with two equally-sized partitions and per-client placement policy, and LFU-Aging with global cache perform the best. SIZE-R with two equally-sized partitions and per-client placement policy has the lowest performance.

The second experiment considers three equally-sized partitions. The same per-client placement policy as described for the first experiment is used to place documents referenced by individual clients in the server workload to each of the three partitions. Figure 6.10 shows the performance results of this second experiment when each of the four replacement policies (SIZE-F, LFU-Aging, LRU and SIZE-R) is used to manage the partitions. The labels on this figure are interpreted in the same manner as Figure 6.9. With reference to both document hit ratio and byte hit ratio, there is no performance improvement in increasing the number of cache partitions to three when other factors are constant. The speculated reason for having the same performance results for both experiments is that almost all the document requests from individual clients at the server are sent to an extremely small set of documents

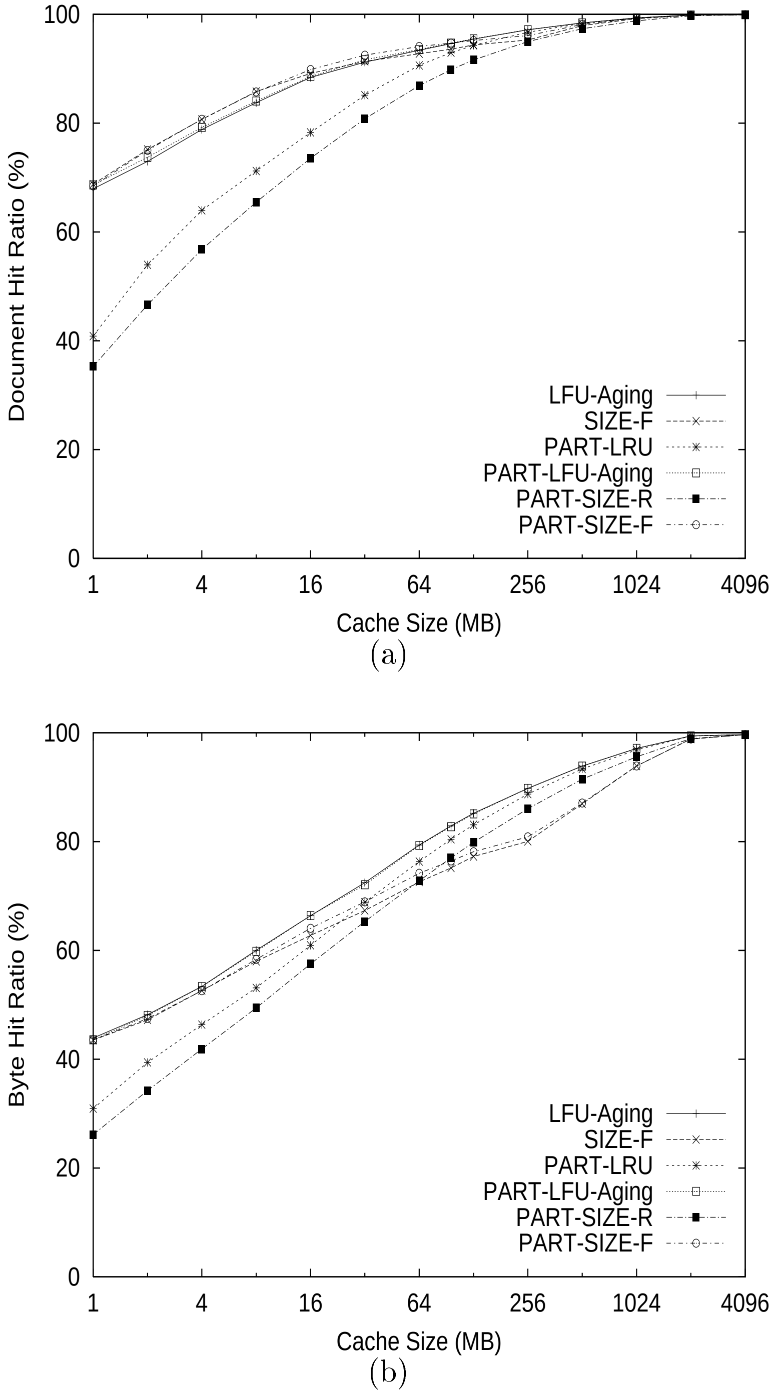
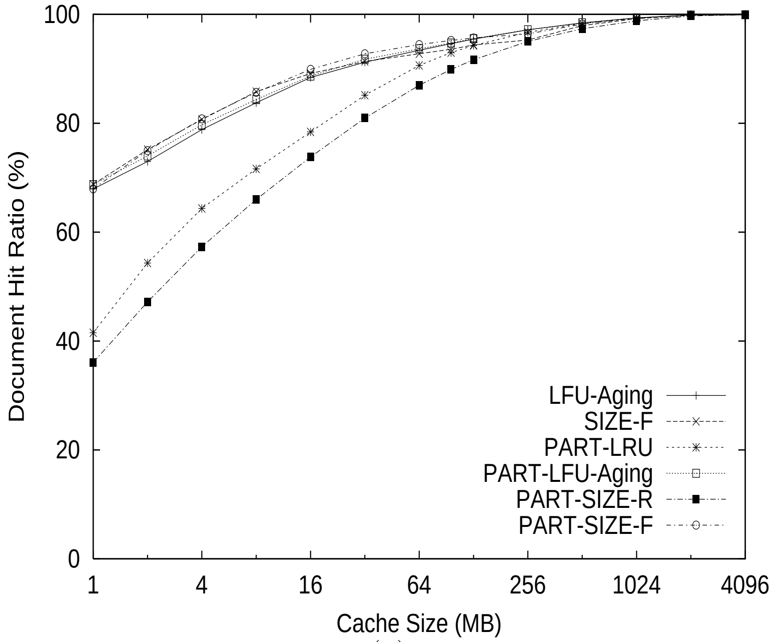
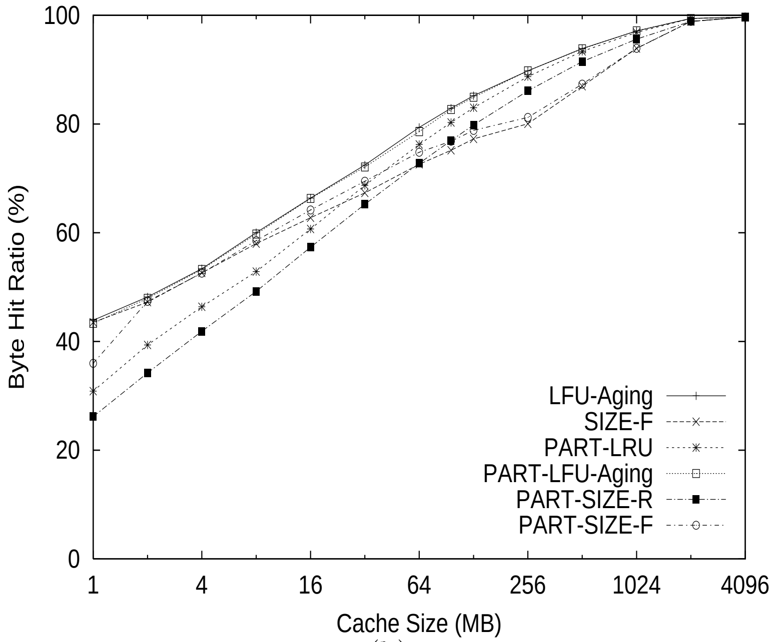


Figure 6.9: Effect of Client-based Partitioning with Two Equally-Sized Partitions: (a) Document Hit Ratio; (b) Byte Hit Ratio



(a)



(b)

Figure 6.10: Effect of Client-Based Partitioning with Three Equal-Sized Partitions: (a) Document Hit Ratio; (b) Byte Hit Ratio

at the server. This means that if the document reference patterns of clients are managed in partitioned caches, this set of frequently requested documents is likely to be replicated in every partition, regardless of the number of partitions. The results of these two experiments, however, demonstrate that client-based partitioning can be useful at the server. Moreover, it is believed that if individual information about clients accessing the server can be obtained, several alternative per-client placement policies can be investigated to understand further the effect of partitioned caches.

6.3.3 Management on the Basis of Client Type

The final experiment is designed to assess how the behavioural properties of individual client reference patterns affect performance under different replacement policies. In order to evaluate this effect, the request streams of selected clients are tested with four replacement policies (SIZE-F, LFU-Aging, LRU, and SIZE-R). In setting up the experiments, LFU-Aging and SIZE-F use the aging parameters set at $A_{max} = 30$ and $M_{refs} = \infty$. These aging values are chosen on the basis of the range of aging parameters reported in [7]. Since this performance analysis is ultimately to understand the relationship between the characteristics of client reference patterns and the performance of cache replacement policies, the warm-up period discussed in Section 6.2.3.5 is not taken into consideration. This implies that first time document misses as a result of a cold cache are included in the performance results.

Figures 6.11 and 6.12 indicate that SIZE-F is the best for the type of client reference pattern exhibited by Client 2597, in terms of both document hit ratio and byte hit ratio. This is because SIZE-F maximizes a given cache size by not evicting small documents that are frequently requested at the expense of large documents that are rarely requested by clients. This means that the number of documents to be stored in the cache can be increased since Client 2597 is a non-human client, whose document reference pattern is uniform. For Client 485 as an example of human client, the performance results show that SIZE-F has the best document hit ratio at small cache sizes and LFU-Aging has the best document hit ratio at large cache sizes. But in terms of byte hit ratio, LFU-Aging produces the best. The outcome of this performance assessment is similar to global caching as shown in Figure 6.2. This means Client 485 exhibits some dominant properties that are similar to dominant properties of the aggregate server workload. In the case of other human clients (Clients 22 and 374), SIZE-R has the best document hit ratio for the type of reference pattern exhibited by Client 22 and SIZE-F has the best document hit ratio for the type of reference pattern exhibited by Client 374.

Although, the difference in performance between cache replacement policies for Clients 374 and 22 is negligible, SIZE-R and LRU perform best in terms of the byte hit ratio for these clients. The performance results obtained from managing the request stream of selected clients suggest that a client-based resource management strategy that managed individual client's request streams separately would not necessarily yield appreciable increase in server performance. However, the results indicate that a client-based resource management strategy could be beneficial at the server if request streams of individual clients are managed separately by policies that take account of their inherent properties. In other words, the use of a replacement policy for global caching may not be beneficial to most clients accessing the server whose request streams may possess different properties based on their type (human or non-human). These differences in the types of clients cause some clients to experience better response times, while others do not.

One obvious advantage to managing the server resources on the basis of client type is the possibility of achieving Pareto-efficient resource management[1]. The implementation of this client-based strategy would require the use of an adaptive policy for matching individual client's request streams based on their types to suitable management policies. Since SIZE-F yields the best performance for a supposedly non-human client (Client 2597), while the performance of LFU-Aging, SIZE-F, LRU and SIZE-R varies slightly from one another for supposedly human clients (Clients 22, 375 and 485), the use of an adaptive policy based on the nature of client's re-

[1]Check Appendix A for the definition of the term.

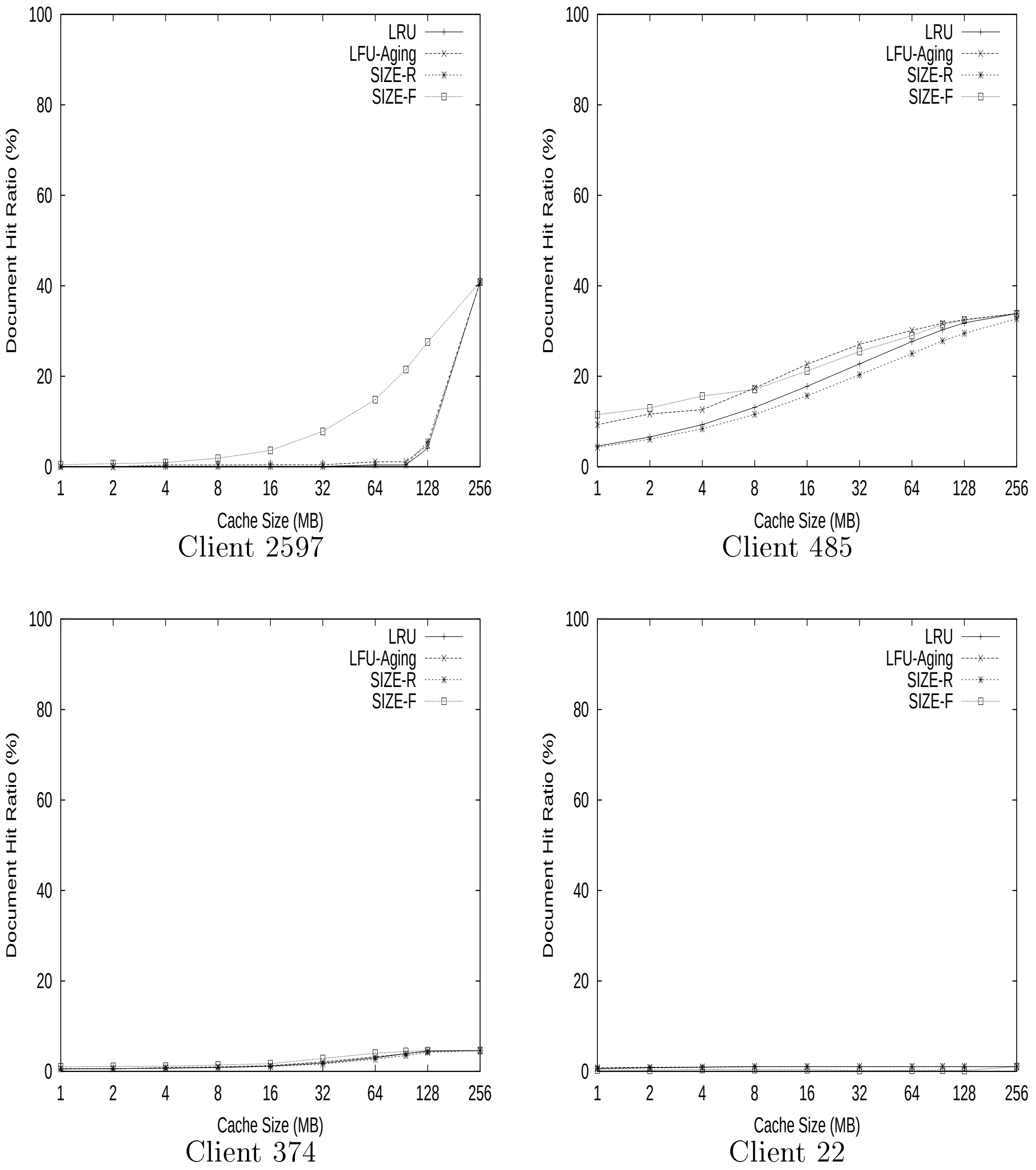


Figure 6.11: Document Hit Ratios of Frequency-Based and Recency-Based Replacement Policies Using the Reference Pattern of Selected Clients

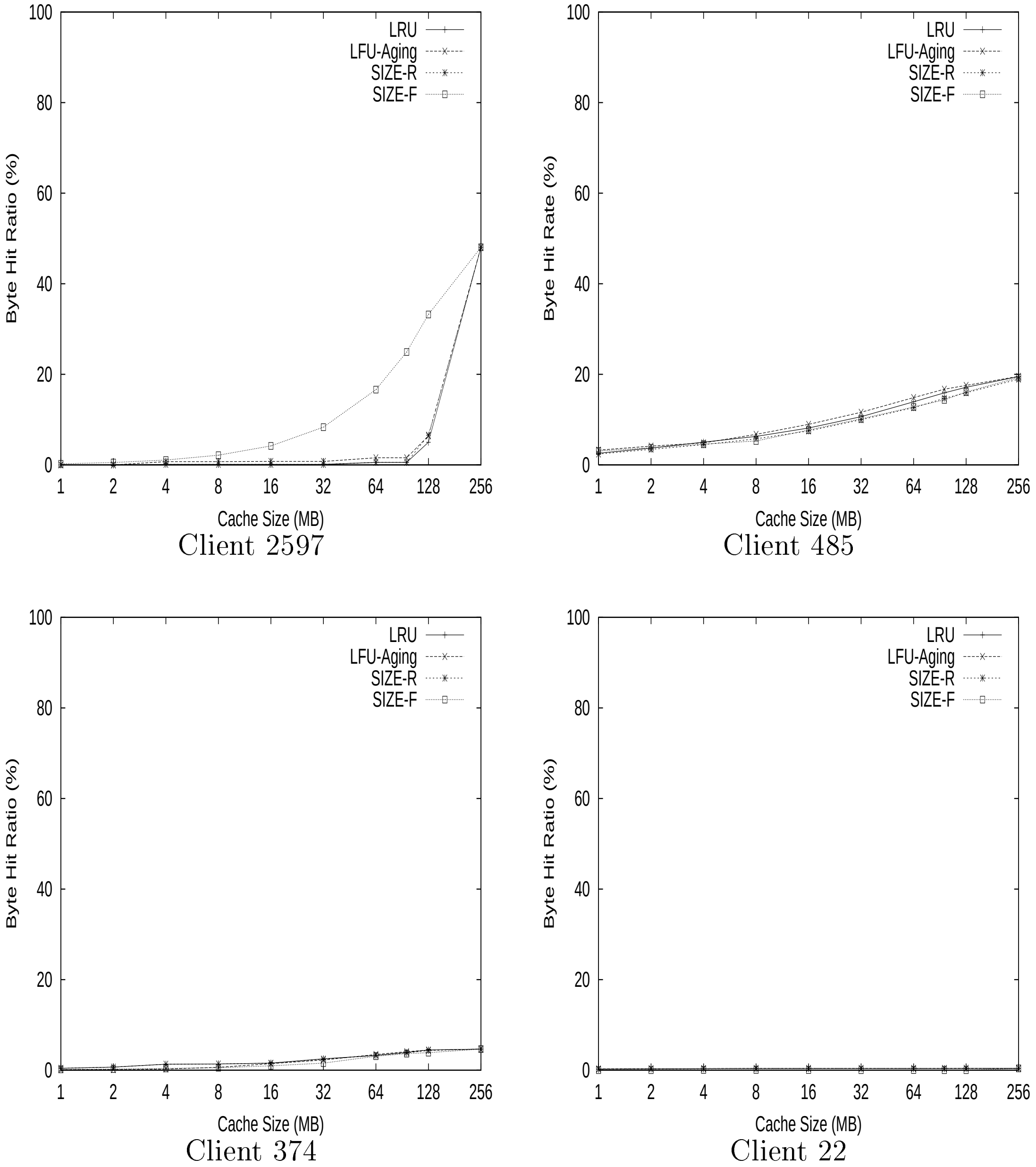


Figure 6.12: Byte Hit Ratios of Frequency-Based and Recency-Based Replacement Policies Using the Reference Pattern of Selected Clients

quest streams (or client types) seen at the server could be effective. Thus, there is a need to obtain client information for identifying the types of clients based on the nature of their reference patterns. Since there is no information within the access log collected to know individual client types (at least whether the client is non-human or human client), it is difficult to investigate such a Pareto-efficient strategy with the use of an adaptive policy for matching the client types to partitioned caches.

6.4 Summary

The chapter has presented cache management strategies that are capable of making informed decisions based on document referencing behaviour of Web clients at each level of the server access hierarchy. Two important cache configurations, global caching and partitioned caching, are studied along with placement policies using document size and client IP address. The rationale for this study is to understand the impact of these factors on Web server performance. In addition, selected client request streams which have been identified into various types (human and non-human) are managed separately with different cache replacement policies to demonstrate the value of using client information in managing server resources.

For global caching with unrestricted document size, the results show that a SIZE replacement policy (with a frequency-based aging policy) performs well at the server. But at large cache sizes (above 256 MB), there is no significant improvement in performance between recency-based replacement policies and frequency-based replacement policies studied. In the case of global caching with document size thresholds for admitting documents, there is no performance improvement when compared to global caching without thresholds. A recency-based policy (LRU) improves performance when global caching and document size thresholds are combined. Since the best performance in both cases is the same, global caching with thresholds is considered to be effective for managing server resources in view of minimal amount of computations if an aging replacement policy is used. In general, experimental results from global caching consistently show SIZE-F and LFU-Aging producing the best document hit ratio and byte hit ratio, respectively.

In investigating partitioned caching, two approaches were used. The first is partitioning on the basis of document size using statistical measures and empirical observations for document placement. The second is partitioning on a per-client basis using client IP address for document placement. The first approach yields some performance improvements over global caching, while the second approach yields no significant performance improvement over global caching. Recency-based policies seem to perform poorly with partitioned caching because temporal locality in the workload is weakened by allocating request streams into partitions.

Management on the basis of client type shows that there exists a relationship between characteristics of client reference patterns and the performance of replacement policies. The implication of this relationship is that a client-based resource management approach at the server can be effective if individual client's request streams are managed separately by policies that take account of their inherent properties rather than just relying only on client IP addresses. The performance results demonstrate the potential for using client information for effective management of server resources.

Chapter 7

Summary, Conclusions and Future Research

This thesis has introduced the concept of a Web server access hierarchy by characterizing the server workload at various levels of access aggregation, including document requests from multiple clients, document requests from single clients, and document requests from sessions within individual clients. The concept of access hierarchy provides additional insight into possible ways of managing the server resources effectively. This chapter concludes the thesis with a summary of how the fundamental research questions identified in Chapter 1 are finally resolved. The contributions of the research are discussed, and suggestions are given for future research.

7.1 Research Summary

This research can be divided into two parts. The first part deals with the characterization of the Web server access hierarchy. The focus of such a characterization is to understand how the behavioural characteristics of Web server workloads have been affected by various factors such as new usage patterns, proxies, upstream caching and interleaving of client streams. Chapters 4 and 5 are devoted to this part of the research. The second part of the research evaluates how the behavioural characteristics obtained from the first part can be used to manage the resources at the server effectively. Chapter 6 discussed this part. A seven-day server workload with about 80 million requests from clients was used throughout the research.

The workload characterization of the Web server access hierarchy deals with: (i) characteristics of the aggregate service demand from multiple clients; (ii) characteristics of the service demand from individual clients; and (iii) characteristics of the service demand in sessions from individual clients. The first characterization provides an opportunity to assess whether or not the aggregate server workload characteristics have changed over time, and if so, the factors responsible for the change. The results from this characterization as discussed in Chapter 4 show that some of these aggregate server characteristics have changed, while others have not. In particular, the document popularity of the aggregate server workload considered does not conform with Zipf's Law because of the extreme popularity of a small set of documents that were responsible for almost all the requests that the server received. Strong temporal locality is observed in the aggregate server workload.

The probability density function indicates that documents between 1-40 kB were most frequently requested and were responsible for most of the data transferred. Furthermore, a correlation measure shows that any preference for small documents over large documents is at most weak, while large documents weakly influence the volume of data transferred over the network. The performance implications of these characteristics, in general, favour the use of caching at the server. Using the correlation measure proposed by Barford *et al.* [15] and the graphical method used by Arlitt [7], there is a tradeoff in cache size between cache policies that optimize request throughput and policies that optimize data throughput at the server. However, cache policies that optimize request throughput by caching (small) documents that are most frequently requested are cost effective for resource management of a busy server. The need for Web page designers to keep the size of documents at the server to be less than 100 kB is also emphasized.

The second characterization is an attempt to understand the nature of individual client reference patterns. It is hypothesized that information about clients accessing the server may be useful in the management of the server resources. The results from this characterization show that a few clients are responsible for most of the requests at the server. Four "heavy-access" clients are selected from the aggregate reference stream and characterized accordingly. It is observed that there is a dramatic difference between the behavioural characteristics exhibited by "human" clients and "non-human" clients such as proxies, firewalls, or Web crawlers. Among supposedly human clients analyzed, the degree of upstream caching effects as explained in Section 2.3 is speculated to influence how the characteristics of these clients differ. Mixing the traffic from these two types of clients may affect the effective management of the server resources.

The characteristics of the reference patterns of supposedly human clients favour the use of caching strategies while non-human clients do not. The implication of this observation is that client information and cache management policies that are amenable to the characteristics of human clients should be key componenets in client-based resource management. However, the effectiveness of a caching strategy on individual clients would depend on the degree of upstream caching experienced.

The third characterization is an attempt to understand the nature of reference patterns within sessions of single clients. The main focus of this characterization is to examine the effectiveness of HTTP persistent connections at the server and to investigate phase-transition in client referencing behaviour. The results show that a fixed timeout value of 100 seconds or more offers the best utilization of HTTP persistent connections at the server. In particular, the request streams of human clients do not utilize persistent connections at the server effectively when the timeout value is small (i.e., below 100 seconds) while non-human clients do when the timeout value is between 1-10 seconds. If the server memory is a bottleneck, an adaptive timeout policy with short timeout value is proposed to improve the utilization of HTTP persistent connections by human clients, while non-human clients would stay with the implementation of HTTP/1.1 using a short timeout value. The implementation of this strategy will depend on the identification of clients at the server as human or non-human.

There is overwhelming evidence that phase-transition behaviour is more prominent in the reference patterns of "heavy-access" Web clients at the server. However, phase-transition behaviour in the request stream of non-human clients is difficult to predict while those of human clients can be predicted. The implication of phase-transition behaviour is that an adaptive strategy based on prefetching or a phase-transition model could be used to manage server resources based on sessions within individual client request streams.

In summary, the behavioural properties that serve as the fundamental basis for using a caching strategy to improve performance are less strong as one moves from the first level of the Web server access hierarchy to the third level. This means that the third level of the Web server access hierarchy could be amenable to an adaptive or phase-transition strategy. The impact of upstream caching effects on individual client reference patterns is speculated to diminish caching properties as one moves to the third level of the server access hierarchy. Therefore, the Web server access hierarchy can be used as a deciding factor for identifying when a particular strategy is suitable for managing the server resources. The behavioural properties of the aggregate server workload from multiple clients favour a caching strategy for managing the server resources, while the behavioural properties of individual client workloads favour the use of a prefetching or phase-transition strategy.

The second part of the thesis seeks to demonstrate how the characteristics of the Web server access hierarchy can be used to manage Web server resources effectively. In achieving this objective, cache management strategies that are capable of making informed decisions on document referencing behaviour of Web clients at each level of the access hierarchy are studied using a trace-driven simulation approach.

Two important cache configurations, global caching and partitioned caching with various placement policies based on document size and client IP address, are considered. The effect on Web server performance of global caching with document size thresholds and of partitioning a cache on the basis of document size and client IP address are investigated. In addition, the relationship between characteristics of client reference patterns and the performance of cache replacement policies is analyzed. The experimental results from conducting the investigations are summarized under global caching, partitioned caching and management on the basis of client type.

With global caching, a SIZE replacement policy (with a frequency-based aging policy to break ties) provides the best cache size-to-performance ratio. However, as the cache size becomes larger, there is no significant improvement between recency-based policies and frequency-based policies for managing the global cache. If the mean document hit ratio and mean byte hit ratio are calculated for every size of the server cache are considered, LFU-Aging provides the best cache size-to-performance ratio. The reason for this is that SIZE-F gains more document hit ratio at the expense of byte hit ratio as the cache size becomes larger. Global caching with document size thresholds show no performance improvement over global caching without document size thresholds. However, the performance of a recency-based policy (LRU) improves significantly when global caching and document size thresholds are combined. Since the best performance in both global caching with or without thresholds is the same, it implies that the amount of computation for managing server resources can be reduced if global caching is combined with document size thresholds.

For partitioned caching with document size(s), the performance results with either statistical or empirical placement policies reveal some performance improvement over global caching. Partitioning on a per-client basis, however, shows no significant performance improvement over global caching. Management on the basis of client type shows a SIZE replacement policy (with frequency-based aging policy to break ties) to be the best for non-human clients, while human clients show different types of replacement policies as the best. The results further show that there is a relationship between the nature of client reference patterns and the performance of replacement policies. The implication of this is that a client-based resource management at the server can be effective if individual client request streams are managed separately by policies that take account of their inherent properties rather than just relying on client IP addresses. Therefore, there is an indication that the use of client information at the server can be an effective way of managing the resources at the server.

7.2 Research Contributions and Conclusions

The dramatic growth in the use of the Web has seen considerable research aimed at providing reliable, scalable and efficient service to the clients. Various related research works have suggested caching strategies at the client and in the network as a way to improve Web performance. This research focussed on how to manage the server resources. The results provide insight into the behavioural characteristics of reference patterns from multiple clients, individual clients and sessions within single clients. The primary contribution of the thesis is the introduction of the Web server access hierarchy concept as a way of approaching how to manage server resources effectively. Secondary contributions of the thesis include the investigation of the behavioural characteristics of clients at the server, the benefits of using client information in the management of server resources, and the performance evaluations of global caching and partitioned caching at the server. Some important conclusions from this research are the following.

1. Some behavioural characteristics of aggregate server workload have changed since earlier studies, while others have not. The changes are not due to the upstream caching effect, but rather to the peculiar nature of the workload studied where an extremely small set of documents is responsible for almost all the requests at the server.

2. Human and non-human clients are two major types of clients at the server and their behavioural characteristics are presented.

3. Document inter-reference times from clients can be modeled as independent and exponentially distributed at ten-minute and one-hour intervals.

4. The existence of a break in the distribution of inter-reference times as a result of Web crawlers or Web user referencing behaviour can invalidate the use of inter-reference times as a measure of persistence behaviour at the server.

5. A tradeoff in cache size exists between cache policies that optimize request

throughput and those that optimize data throughput at the server. If large cache size is placed at a busy server and cache policies that optimize data throughput is used to manage the cache, there is no need for determining which of the resources is the bottleneck since both CPU and disk system are managed effectively at the same time.

6. Since the server memory is considered as a bottleneck, a change in the current implementation of HTTP/1.1 persistent connections is suggested. In particular, a strategy based on an adaptive timeout policy with short timeout values for human clients and a fixed timeout policy with short timeout values for non-human clients is suggested.

7. For global caching, a SIZE replacement policy (with frequency-based aging policy for breaking ties) provides the best cache size-to-performance ratio, while LFU-Aging has the best combination of document and byte hit ratios.

8. Partitioning the cache on the basis of document size is more effective than partitioning the cache on the basis of client IP address. Moreover, size-based partitioning outperforms global caching.

9. A client-based resource management at the server could be effective if individual client's request streams are managed separately by policies that take account of their inherent properties.

7.3 Directions for Future Research

There are several future research directions that can be investigated. First, self-similarity across the levels of the Web server access hierarchy can be investigated. This is necessary in view of understanding the impact of upstream caching effects on self-similarity. In addition, evaluation of a new size-based replacement policy that uses the volume of bytes transferred to age documents in the cache can be carried out.

Second, for client-based management to be effective, it will be necessary to collect enough client information to identify the type of clients accessing the server. Such an extended access log could be used to perform simulation experiments to assess the value of client information in resource management. Adaptive and knowledge-based policies could be designed to use information on client behaviour for decision making. In particular, the separation of "human" and "non-human" clients could be useful since their characteristics are quite different.

Third, the evaluation of algorithms for adjusting document working-sets and prefetching documents ahead of their use needs to be investigated in order to take advantage of phase-transition behaviour in sessions within individual client request streams at the server.

Finally, the development of a synthetic workload generator based on the behavioural characteristics of clients identified at the server could be worthwhile to explore the possible performance gains in using a client-based resource management strategy. In addition, such synthetic generator can be fine-tuned to model the requests and bytes transferred of sessions within client request streams to be exponentially distributed. This kind of client request streams can be used to evaluate adaptive timeout policies with short timeout values for improving HTTP persistent connections of human clients at the server.

References

[1] T. F. Abdelzaher and N. Bhatti, "Web Content Adaptation to Improve Server Overload Behaviour," in *Proceedings of the Eighth International World Wide Web Conference*, Toronto, Ontario, May 1999.

[2] G. Abdulla, *Analysis and Modeling of World Wide Web Traffic*, Ph.D. Dissertation, Virginia Polytechnic Institute and State University, Blacksburg, Virginia, May 1998.

[3] G. Abdulla, B. Liu, and E. A. Fox, "Searching the World Wide Web: Implications from Studying Different User Behaviour," in *Proceedings of the WebNet 98 World Wide Web Conference of the World Wide Web, Internet and Intranet*, Orlando, Florida, November 1998.

[4] M. Abrams, C. R. Standridge, G. Abdulla, S. Williams, and E. A. Fox, "Caching Proxies: Limitations and Potentials," in *Proceedings of the Fourth International World Wide Web Conference*, Boston, Massachusetts, December 1995.

[5] V. Almeida, A. Bestavros, M. Crovella, and A. Oliveira, "Characterizing Reference Locality in World Wide Web," in *Proceedings of PDIS 1996, The IEEE Conference on Parallel and Distributed Information Systems*, Miami Beach, Florida, pp. 92-103, December 1996.

[6] Apache HTTP Server Project. Available at URL: http://www.apache.org/httpd.html.

[7] M. F. Arlitt, *A Performance Study of Internet Web Servers*, M.Sc. Thesis, Department of Computer Science, University of Saskatchewan, Saskatoon, Saskatchewan, 1996.

[8] M. F. Arlitt, "Characterizing Web User Sessions," *Performance Evaluation Review*, Vol. 28, No. 2, pp. 50-56, September 2000.

[9] M. F. Arlitt and C. L. Williamson, "Web Server Workload Characterization: The Search for Invariants," in *Proceedings of the 1996 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*, Philadelphia, Pennsylvania, pp. 126-137, May 1996.

[10] M. F. Arlitt and C. L. Williamson, "Internet Web Servers: Workload Characterization and Performance Implications," *IEEE/ACM Transactions on Networking*, Vol. 5, No. 5, pp. 631-645, October 1997.

[11] M. Arlitt and T. Jin, "Workload Characterization of the 1998 World Cup Web Site," Technical Report, Hewlett-Packard Laboratories, February 1999.

[12] M. Arlitt and T. Jin, "A Workload Characterization Study of the 1998 World Cup Web Site," *IEEE Networks*, Vol. 14, No. 3, pp. 30-37, May/June 2000.

[13] H. Balakrishnan, S. Seshan, M. Stemm, and R. H. Katz, "Analyzing Stability in Wide Area Network Performance," in *Proceedings of the 1997 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*, Seattle, Washington, pp. 2-12, June 1997.

[14] G. Banga and P. Druschel, "Measuring the Capacity of a Web Server Under Realistic Loads," *World Wide Web*, Vol. 2, No. 1-2, pp. 69-83, January 1999.

[15] P. Barford, A. Bestavros, A. Bradley, and M. Crovella, "Changes in Web Client Access Patterns: Characteristics and Caching Implications," *World Wide Web*, Vol. 2, No. 1, pp. 15-28, January 1999.

[16] P. Barford and M. Crovella, "Generating Representative Web Workloads for Network and Server Performance Evaluation," in *Proceedings of the 1998 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*, Madison, Wisconsin, pp. 151-160, July 1998.

[17] P. Barford and M. Crovella, "A Performance Evaluation of Hyper Text Transfer Protocols," in *Proceedings of the 1999 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*, Atlanta, Georgia, pp. 188-197, May 1999.

[18] H. Berghel, "The Client's Side of the World Wide Web," *Communications of the ACM*, Vol. 39, No. 1, pp. 30-40, January 1996.

[19] T. Berners-Lee, "The World Wide Web: Past, Present and Future". Available at URL: http://www.w3.org/People/Berners-Lee/1996/ppf.html

[20] T. Berners-Lee, R. Calliau, and B. Pollermann, "World Wide Web: The Information Universe," *Electronic Networking: Research, Applications and Policy*, Vol. 2, No. 1, Spring 1992.

[21] T. Berners-Lee, R. Cailliau, A. Luotonen, H. F. Nielsen, and A. Secret, "The World Wide Web," *Communications of the ACM*, Vol. 37, No. 8, pp. 76-82, August 1994.

[22] A. Bestarvros, "Speculative Data Dissemination and Service to Reduce Server Load, Network Traffic and Service Time in Distributed Information System,"

in *Proceedings of the 1996 International Conference on Data Engineering*, New Orleans, Lousiana, March 1996.

[23] A. Bestavros, R. L. Carter, M. E. Crovella, C. R. Cunha, A. Heddaya, and S. A. Mirdad, "Application-Level Document Caching in the Internet," in *Proceedings of the Second International Workshop on Services in Distributed and Networked Environments (SDNE' 95)*, Whistler, British Columbia, pp. 166-173, June 1995.

[24] A. Bestavros "Demand-Based Document Dissemination for the World-Wide Web," Technical Report, Department of Computer Science, Boston University, Boston, Massachussets, 1995.

[25] J. Bolot and P. Hoschka "Performance Engineering of the World Wide Web: Application to Dimensioning and Cache Design," in *Proceedings of the Fifth International World Wide Web Conference*, Paris, France, pp. 6-10, May 1996.

[26] H. Braun and K. C. Claffy, "Web Traffic Characterization: An Assessment of the Impact of Caching Documents from NCSA's Web Server," *Computer Networks and ISDN Systems*, Vol. 28, pp. 37-51, 1996.

[27] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker, "Web Caching and Zipf-Like Distributions: Evidence and Implications," in *Proceedings of the IEEE INFOCOM '99 Conference*, New York, New York, March 1999.

[28] L. Breslau, P. Cao, L. Fao, G. Phílliips, and S. Shenker "On the Implications of Zipf's Law for Web Caching, " in *Proceedings of the Third International World Wide Web Caching Workshop*, Trans-European Research and Education Networking Association, Manchester, United Kingdom, June 1998.

[29] R. Bunt, D. Eager, G. Oster, and C. Williamson, "Achieving Load Balance and Effective Caching in Clustered Web Servers," in *Proceedings of the Fourth International Web Caching Workshop*, San Diego, California, pp. 159-169, March/April 1999.

[30] L. D. Catledge, and J. E. Pitkow, "Characterizing Browsing Strategies in the World Wide Web," *Computer Networks and ISDN Systems*, Vol. 26, No. 6, pp. 1065-1073, 1995.

[31] A. Chankhunthod, P. B. Danzig, C. Neerdaels, M. F. Schwartz, and K. J. Worrell, "A Hierarchical Internet Object Cache," in *Proceedings of the 1996 USENIX Annual TECHNICAL Conference*, San Diego, California, January 1996.

[32] J. Cho, H. Garcia-Molina, and L. Page, "Efficient Crawling Through URL Ordering," in *Proceedings of 7th World Wide Web Conference*, 1998. Available at URL: http://dbpubs.stanford.edu/ pub/1998-51

[33] R. M. Clark, S. J. D. Cox, and G. M. Laslett, "Generalizations of Power-Law Distributions Applicable to Sampled Fault-Trace Lengths: Model Choice, Parameter Estimation and Caveats," *Geophysical Journal International*, No. 136, pp. 357-372, 1999.

[34] M. Crovella and A. Bestavros *Performance Characteristics of World Wide Web Information Systems*, ACM SIGMETRICS 1997 Tutorial, Seattle, Washington, June 1997.

[35] M. E. Crovella and A. Bestavros, "Self-Similarity in World Wide Web Traffic: Evidence and Possible Causes," in *Proceedings of the 1996 ACM SIGMETRICS International Conference on Measurement and Performance Modeling of Computer Systems*, Philadelphia, Pennsylvania, pp. 160-169, May 1996.

[36] C. R. Cunha and A. Bestavros, and M. E. Crovella, "Characteristic of World Wide Web Client-Based Traces," Technical Report TR-95-010, Department of Computer Science, Boston University, Boston, Massachussets, April 1995.

[37] R. D'Angostino and M. Stephens, Editors, *Goodness-of-Fit Techniques*, Marcel Dekker Inc., New York, New York, 1986.

[38] P. J. Denning and S. C. Schwartz, "Properties of the Working-Set Model," *Communications of the ACM*, Vol. 15, No. 3, pp. 191-198, March 1972.

[39] P. J. Denning, "On Modeling Program Behavior," in *Proceeding of the Spring Joint Computer Conference*, pp. 937-944, 1972.

[40] J. Dilley, R. Friedrich, T. Jin, and J. Rolia, "Web Server Measurement and Modeling Techniques," *Performance Evaluation*, 33, pp. 5-26, 1998.

[41] B. M. Duska, D. Marwood, and M. J. Feeley, "The Measured Characteristic s of World Wide Web Client Proxy Caches," in *Proceedings of the USENIX Symposium on Internet Technology and Systems*, Monterey, California, December 1997.

[42] L. Eggert and J. Heidemann, "Application-Level Differentiated Services for Web Servers," *World Wide Web*, Vol. 2, No. 3, pp. 133-142, July 1999.

[43] Ferrari, D. "Improving Locality by Critical Working Sets," *Communications of the ACM*, Vol. 17, No. 11, pp. 614-620, 1974.

[44] R. Fielding, J. Gettys, J. C. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee, "Hypertext Transfer Protocol - HTTP 1.1," Technical Report, HTTP Working Group INTERNET-DRAFT, Revision 6, May 1999.

[45] X. Gabaix, "Zipf's Law for Cities: An Explanation," *Quarterly Journal of Economics*, Vol. 114, pp. 739-767, August 1999.

[46] S. Glassman, "A Caching Relay for the World Wide Web," *Computer Networks and ISDN Systems*, Vol. 27, pp. 165-173, 1994.

[47] J. Gwertzman and M. Seltzer, "The Case for Geographical Push Cashing," in *Proceedings of the Fifth Workshop on Hot Topics in Operating Systems*, Orcas Island, Washington, pp. 51-55, May 1995.

[48] A. Heddaya, "Dynacache: Weaving Caching into the Internet," in *Proceedings of the Third International World Wide Web Caching Workshop*, Manchester, England, June 1998.

[49] B. Hermans, "Intelligent Software Agents on the Internet: An Inventory of Currently Offered Functionality in the Information Society and a Prediction of (near) Future Developments". Available at URL: http://www.firstmonday.dk/issues/issue2-3/ch_123/index.html

[50] J. H. Hine, C. E. Wills, A. Martel, and J. Sommers, "Combining Client Knowledge and Resource Dependencies for Improved World Wide Web Performance", in *Proceedings of the INET 1998 Conference*, Geneva, Switzerland, July 1998.

[51] A. Iyengar and J. Challenger, "Improving Web Server Performance by Caching Dynamic Data," in *Proceedings of the USENIX Symposium on Internet Technology and Systems*, Monterey, California, December 1997.

[52] A. Iyengar, E. MacNair, and T. Nguyen "Web Server Performance Under Heavy Loads," IBM Research Report RC 20976, Thomas J. Watson Research Center, Yorktown Heights, New York, September 1997.

[53] S. Jin and A. Bestavros, "Sources and Characteristics of Web Temporal Locality," in *Proceedings of the Eighth International Symposium on Medeling, Analysis and Simulation of Computer and Telecomminucation Systems (MASCOTS '00)*, San Francisco, California, August/September 2000.

[54] R. Karedla, J. S. Love, and B. G. Wherry, "Caching Strategies to Improve Disk System Performance", *IEEE Computer*, Vol. 27, No. 3, pp.38-46, March 1994.

[55] E. D. Katz, M. Butler, and R. McGrath "A Scalable HTTP Server," *Computer Networks and ISDN Systems*, Vol. 27, pp.155-64, 1994.

[56] M. G. Kienzle, J. A. Garay, and W. H. Tetzlaff, "Analysis of Page-Reference Strings of an Interactive System," *IBM Journal of Research and Development*, Vol. 32, No. 4, pp. 523-535, July 1988.

[57] B. Krishnamurthy, J. C. Mogul, and D. M. Kristol, "Key Differences Between HTTP/1.0 and HTTP/1.1," in *Proceedings of the Eighth International World Wide Web Conference*, Toronto, Ontario, May 1999.

[58] T. T. Kwan, R. E. McGrath, and D. A. Reed , "NCSA's World Wide Web Server: Design and Performance," *IEEE Computer*, Vol. 28, No. 11, pp. 68-74, November 1995.

[59] E. Levy-Abegnoli, A. Iyengar, J. Song, and D. Dias, "Design and Performance of a Web Server Accelerator," in *Proceedings of the IEEE INFOCOM '99 Conference*, New York, New York, March 1999.

[60] A. Luotonen and K. Altis, "World Wide Web Proxies," *Computer Networks and ISDN Systems*, Vol. 27, pp. 147-154, 1994.

[61] A. Mahanti, *Web Proxy Workload Characterization and Modeling*, M.Sc. Thesis, Department of Computer Science, University of Saskatchewan, Saskatoon, Saskatchewan, September 1999.

[62] A. Mahanti, D. Eager and C. Williamson, "Temporal Locality and Its Impact on Web Proxy Cache Performance," *Performance Evaluation*, Vol. 42, Nos. 2-3, October 2000.

[63] S. Manley and M. Seltzer, "Web Facts and Fantasy," in *Proceedings of the 1997 USENIX Symposium on Internet Technologies and Systems*, Monterey, California, December 1997.

[64] E. P. Markatos, "Main Memory Caching of Web Documents," *Computer Networks and ISDN Systems*, Vol. 28, No. 7-11, pp. 893-905, May 1996.

[65] J. C. Mogul, "The Case for Persistent-Connection HTTP," in *Proceedings of the 1995 ACM SIGCOMM Conference*, Cambridge, Massachussets, pp. 299-313, August 1995.

[66] J. C. Mogul "Network Behaviour of a Busy Web Server and Its Clients," WRL Research Report 95/4, Digital Western Research Laboratory, May 1995. Available at URL: http://www.research.digital.com/wrl/techreports

[67] C. D. Murta, V. A. F. Almeida, and W. Meira, "Analyzing Performance of Partitioned Caches for the World Wide Web," in *Proceedings of the Third International World Wide Web Caching Workshop*, Manchester, England, June 1998.

[68] E. Nahum, T. Barzilai, and D. Kandlur, "Performance Issues in World Wide Web Servers," *World Wide Web*, Vol. 2, No. 1-2, 1999.

[69] Netscape Communication Corporation. Available at URL: http://www.netscape.com/servers

[70] Network Wizards, "Internet Domain Survey", January 2000. Available at URL: http://www.isc.org/ds/WWW-200001/report.html

[71] V. N. Padmanabhan and J. C. Mogul, "Improving World Wide WWeb Latency," *in Electronic Proceedings of the Second International Word Wide Web Conference*, Chicago, IIIinois, October 1994.

[72] V. N. Padmanabhan and J. C. Mogul, "Using Predictive Prefetching to Improve World Wide Web Latency," *ACM Computer Communication Review*, Vol. 26, No. 3, pp. 22-36, July 1996.

[73] J. B. Peachey, R. B. Bunt, and C. J. Colbourn, "Bradford-Zipf Phenomena in Computer Systems", in *Proceedings of Canadian Information Processing Society 1982 National Conference*, Saskatoon, Saskatchewan, Canada, pp. 155-161, May 1982.

[74] V. S. Pai, M. Aron, G. Banga, M. Svendsen, P. Druschel, W. Zwaenepoel, and E. Nahum, "Locality-Aware Request Distribution in Cluster-Based Network Servers," in *Proceedings of the Eighth ACM Conference on Architectural Support for Programming Languages and Operating Systems*, San Jose, California, October 1998.

[75] T. Partl and A. Dingle "A Comparison of World Wide Web Caching Algorithm Efficiency," *ICM Workshop on Web Caching*, Warsaw, Poland, September 1996.

[76] V. Paxson, "Empirically Derived Analytic Models of Wide-Area TCP Connections," *IEEE/ACM Transactions on Networking*, Vol. 2, No. 4, pp. 316-336, August 1994.

[77] V. Paxson and S. Floyd, "Wide Area Traffic: The Failure of Poisson Modeling," in *Proceedings of the 1994 ACM SIGCOMM Conference*, London, England, pp. 257-268, August 1994.

[78] S. Putz, "Interactive Information Services Using World Wide Web Hypertext," *Computer Networks and ISDN Systems*, Vol. 27, pp. 273-280, 1994.

[79] A. Rousskov and V. Soloview, "A Performance Study of the Squild Proxy on HTTP/1.0," *World Wide Web*, Vol. 2, No. 1-2, pp. 47-67, January 1999.

[80] L. Slothouber, "A Model of Web Server Performance," 1996. Available at URL: http://louvx.biap.com/white papers/performance/overview.html.

[81] J. Spirn, "Distance String Models for Program Behavior," *IEEE Computer*, Vol. 9, No. 11, pp. 14-20, November 1976.

[82] A. S. Tanenbaum, *Computer Networks*, Third Edition, Prentice-Hall Inc., 1996.

[83] I. Tatarinov, A. Rousskov, and V. Soloview "Static Caching in Web Servers," in *Proceedings of the IEEE International Conference on Computer Communication and Networks*, Las Vegas, Nevada, September 1997.

[84] R. Turner and B. Strecker, "Use of the LRU Stack Depth Distribution for Simulation of Paging Behaviour," *Communications of the ACM*, Vol. 20, No. 11, pp. 795-798, November 1977.

[85] R. J. Vetter, C. Spell, and C. Ward, "Mosaic and the World Wide Web," *IEEE Computer*, Vol. 27, No. 10, pp. 49-57, October 1994.

[86] S. Williams, M. Abrams, C. Standridge, G. Abdulla, and E. Fox, "Removal Policies in Network Caches for World-Wide Web Documents," in *Proceedings of 1996 ACM SIGCOMM Conference*, Stanford, California, pp. 293-305, August 1996.

[87] K. Yap, "A Technical Overview of the New HTTP/1.1 Specification," in *Proceedings of the AusWeb97, The Third Australian World Wide Web Conference*, Australia, May 1997.

[88] P. S. Yu and E. MacNair, "Performance Study of a Collaborative Method for Hierarchical Caching in Proxy Servers," *Computer Networks and ISDN Systems*, Vol. 30, pp. 215-224, 1998.

[89] The Zeus World Wide Web Server. Available at URL: http://www.zeus.co.uk

[90] L. Zhang, S. Floyd, and V. Jacobson, "Adaptive Caching," in *Proceedings of the NLANR Web Cache Workshop*, National Laboratory for Applied Network Research, Boulder, Colorado, June 1997.

[91] G. K. Zipf, *Human Behaviour and the Principle of Least Effort*, Addison-Wesley, Cambridge, Massachusetts, 1949.

Appendix A

Glossary

1. Common Gateway Interface (CGI): a technology that allows a Web server to obtain data from (or send data to) databases, documents, and other programs, and present that data to clients via the Web.

2. Domain Name Service (DNS): a mechanism of translating names of host computers in order to facilitate communication between computers on the Internet.

3. File Transfer Protocol (FTP): an Internet application service that allows documents to be copied from one remote computer to another.

4. Human Client: a client that accesses the Web server on behalf of human users.

5. HyperText Markup Language (HTML): a language used for developing documents to be accessed by clients via the Web.

6. HyperText Transfer Protocol (HTTP): a reliable application layer protocol that runs on top of Transmission Control Protocol/Internet Protocol for retrieving documents on the Web.

7. Least Recently Used (LRU): a replacement policy that removes the least recently requested document from the cache.

8. Least Frequently Used (LFU): a replacement policy that removes the least frequently requested document from the cache. LFU-Aging is a variation of this type of replacement policy.

9. Local Area Network (LAN): a network that connects several computers that a located nearby (in the same room or building) for sharing files and devices such as printers.

10. Multipurpose Internet Mail Extension (MIME): an official Internet standard for encoding documents to be exchanged between the client and the server via the Web.

11. Non-Human Client: a client that accesses the Web server on behalf of non-human users such as search engines or Web crawlers.

12. Pareto-Efficient Resource Management: a technique of allocating resources such that no client accessing the server is worse off, but at least one client is made to experience better document response time.

13. Persistent-Connection HyperText Transfer Protocol (P-HTTP): a key feature of HTTP/1.1 that allows a Web client to use a single Transmission Control Protocol connection at the server to make multiple requests and receive multiple responses over a fixed duration of time.

14. Phase Transition: an overwhelming change in the consecutive reference patterns of sessions within the client traffic.

15. Space-Time-Working-Set (STWS): a replacement policy that excludes the document with the largest product of time since last access and byte size.

16. Transport Control Protocol/Internet Protocol (TCP/IP): a reliable protocol that allows computers of all sizes, from many different computer vendors, running totally different operating systems, to communicate with each other. TCP is a transport layer protocol that transmits the application layer data stream in small chunks, called packets. IP is an unreliable network layer protocol that routes the packets from one computer to another over the Internet.

17. Universal Resource Locator (URL): a string of characters that identifies a document viewable on the Web.

18. World Wide Web (WWW): a distributed information access and retrieval system that uses hypertext technology in a client-server manner on the Internet.

19. Web Page: an item retrieved by a Universal Resource Locator. It may consist of more than one type of documents.

20. Web Client: a program used to send and receive documents via the Web.

21. Web Proxy: an intermediary program that acts as a server and a client for the purpose of making requests on behalf of other clients. It is used for caching frequently requested documents at the servers. This term is often used to refer to a machine that runs the Web proxy program.

22. Web Crawler: a program that retrieves Web pages for search engines by running several processes at a time.

23. Web Server: a program that accepts and supplies document requests to clients. This term is often used to refer to a machine that runs the Web server program.

24. Web Server Access Hierarchy (WSAH): a three-tier level of Web server access aggregation that describes aggregate traffic from multiple clients, traffic from individual clients and traffic within sessions from individual clients.